Міністерство освіти та науки України
Національний університет водного господарства та природокористування
Кафедра автоматизації, електротехнічних та комп’ютерно-інтегрованих технологій

04-03-359М

МЕТОДИЧНІ ВКАЗІВКИ

до виконання лабораторних робіт з навчальної дисципліни
«Штучний інтелект в робототехніці»
для здобувачів вищої освіти першого (бакалаврського) рівня
за освітньо-професійною програмою
«Робототехніка та штучний інтелект»
спеціальності 151 «Автоматизація та комп’ютерно-інтегровані технології» денної та заочної форм навчання

Рекомендовано науково-методичною радою з якості ННІ АКОТ
Протокол № 10 від 20.09.2022 р.

Рівне – 2022

Методичні вказівки до виконання лабораторних робіт з навчальної дисципліни «Штучний інтелект в робототехніці» для здобувачів вищої освіти першого (бакалаврського) рівня за освітньо-професійною програмою «Робототехніка та штучний інтелект» спеціальності 151 «Автоматизація та комп’ютерно-інтегровані технології» денної та заочної форм навчання [Електронне видання] / Сафоник А. П. – Рівне : НУВГП, 2022. – 193 с.

Укладач: Сафоник А.П., д.т.н., професор, професор кафедри автоматизації, електротехнічних та комп‘ютерно-інтегрованих технологій.

Відповідальний за випуск: Древецький В. В., д.т.н., професор, завідувач кафедри автоматизації, електротехнічних та комп‘ютерно-інтегрованих технологій.

Керівник освітньої програми «Робототехніка та штучний інтелект»: Сафоник А. П., д.т.н., професор кафедри автоматизації, електротехнічних та комп’ютерно-інтегрованих технологій.

Зміст

Лабораторна робота №1. Задачі штучного інтелекту в електронній комерції. Обробка, аналіз даних та їх візуалізація

1.1. Мета роботи

Навчитися обробляти експериментальні дані та їх візуалізувати.

1.2. Теоретичні відомості

Аналіз даних визначається як процес очищення, перетворення та моделювання даних для виявлення корисної інформації для прийняття ділових рішень. Метою аналізу даних є отримання корисної інформації з даних та прийняття рішення на основі аналізу даних.

Простий приклад аналізу даних - це коли ми приймаємо якесь рішення в нашому повсякденному житті, думаючи про те, що сталося минулого разу, або що станеться, вибравши саме це рішення. Це не що інше, як аналіз нашого минулого чи майбутнього та прийняття рішень на основі них. Для цього ми збираємо спогади про своє минуле або мрії про наше майбутнє. Тож це не що інше, як аналіз даних. Зараз те саме, що робить аналітик для комерційних цілей, називається Аналіз даних.

Інструменти аналізу даних полегшують користувачам обробку даних і маніпулювання ними, аналіз взаємозв’язків та кореляцій між наборами даних, а також допомагає виявити закономірності та тенденції інтерпретації. Ось повний перелік інструментів, що використовуються для аналізу даних у дослідженнях.

Види аналізу даних: прийоми та методи

Існує кілька типів методів аналізу даних , які існують на основі бізнесу та технологій. Однак основними методами аналізу даних є:

- Аналіз тексту
- Статистичний аналіз
- Діагностичний аналіз
- Прогностичний аналіз
- Давній аналіз
- Аналіз тексту

Аналіз тексту також називають видобутком даних. Це один із методів аналізу даних, щоб виявити закономірність у великих наборах даних за допомогою баз даних або інструментів аналізу даних. Раніше він трансформував необроблені дані у ділову інформацію. Інструменти бізнес-аналітики присутні на ринку, який використовується для прийняття стратегічних бізнес-рішень. Загалом він пропонує спосіб вилучення та вивчення даних та виведення шаблонів і, нарешті, інтерпретацію даних.

Статистичний аналіз

Статистичний аналіз показує "Що сталося?" за допомогою минулих даних у формі інформаційних панелей. Статистичний аналіз включає збір, аналіз, інтерпретацію, представлення та моделювання даних. Він аналізує набір даних або зразок даних. Існує дві категорії цього типу аналізу - описовий аналіз та інференційний аналіз.

Описовий аналіз

аналізує повні дані або зразок узагальнених числових даних. Він показує середнє значення та відхилення для безперервних даних, тоді як відсоток та частота для категоріальних даних.

Інференційний аналіз

аналізує зразок із повних даних. У цьому типі аналізу ви можете знайти різні висновки з тих самих даних, вибравши різні вибірки.

Діагностичний аналіз

Діагностичний аналіз показує "Чому це сталося?" знаходячи причину на основі аналізу статистичного аналізу. Цей аналіз корисний для виявлення моделей поведінки даних. Якщо у вашому бізнес-процесі з'являється нова проблема, ви можете заглянути в цей Аналіз, щоб знайти подібні закономірності цієї проблеми. І у нього можуть бути шанси використовувати подібні рецепти для нових проблем.

Прогностичний аналіз

Прогнозивний аналіз показує, "що може статися", використовуючи попередні дані. Найпростіший приклад аналізу даних - це те, якби минулого року я купив дві сукні на основі своїх заощаджень, а якщо цього року моя зарплата зросла вдвічі, то я можу купити чотири сукні. Але, звичайно, це непросто, оскільки вам доводиться думати про інші обставини, такі як шанси на зростання ціни на одяг цього року, або, можливо, замість суконь ви хочете придбати новий велосипед, або вам потрібно купити будинок!

Отже, цей Аналіз робить прогнози щодо майбутніх результатів на основі поточних чи минулих даних. Прогнозування - це лише оцінка. Його точність залежить від того, скільки у вас детальної інформації і скільки ви в ній копаєте.

Давній аналіз

Аналіз приписів поєднує уявлення з усіх попередніх аналізів,

щоб визначити, які дії вжити для поточної проблеми чи рішення. Більшість компаній, що керуються даними, використовують рецептурний аналіз, оскільки прогнозного та описового аналізу недостатньо для підвищення продуктивності даних. Спираючись на поточні ситуації та проблеми, вони аналізують дані та приймають рішення.

Процес аналізу даних

Процес аналізу даних - це не що інше, як збір інформації за допомогою відповідної програми або інструменту, який дозволяє досліджувати дані та знаходити в них шаблон. Виходячи з цієї інформації та даних, ви можете приймати рішення або отримувати остаточні висновки.

Аналіз даних складається з наступних етапів:

- Збір вимог до даних
- Збір даних
- Очищення даних
- Аналіз даних
- Інтерпретація даних
- Візуалізація даних
- Збір вимог до даних

Перш за все, вам слід подумати, чому ви хочете зробити цей аналіз даних? Все, що вам потрібно, щоб з'ясувати мету або мету проведення аналізу даних. Ви повинні вирішити, який тип аналізу даних ви хотіли зробити! На цьому етапі ви повинні вирішити, що аналізувати і як це виміряти, ви повинні розуміти, для чого ви проводите розслідування та які заходи ви повинні використовувати для цього аналізу.

Збір даних

Після збору вимог ви отримаєте чітке уявлення про те, які речі ви повинні виміряти, і якими мають бути ваші висновки. Тепер настав час збирати ваші дані на основі вимог. Збираючи свої дані, пам'ятайте, що зібрані дані повинні бути оброблені

або організовані для аналізу. Оскільки ви збирали дані з різних джерел, вам доведеться вести журнал із датою збору та джерелом даних.

Очищення даних

Тепер будь-які дані, які збираються, можуть бути не корисними або несуттєвими для вашої мети Аналізу, отже, їх слід очистити. Дані, які збираються, можуть містити повторювані записи, пробіли або помилки. Дані повинні бути очищені без помилок. Цю фазу необхідно виконати перед Аналізом, оскільки на основі очищення даних результати Аналізу будуть ближчими до очікуваних результатів.

Аналіз даних

Після збору, очищення та обробки даних вони готові до аналізу. Під час маніпулювання даними ви можете виявити, що у вас є точна інформація, яка вам потрібна, або вам може знадобитися зібрати більше даних. На цьому етапі ви можете використовувати інструменти та програмне забезпечення для аналізу даних, які допоможуть вам зрозуміти, інтерпретувати та зробити висновки на основі вимог.

Інтерпретація даних

Проаналізувавши ваші дані, нарешті настав час інтерпретувати ваші результати. Ви можете обрати спосіб вираження або передачі аналізу даних, або просто використовувати слова, або таблицю чи діаграму. Потім використовуйте результати процесу аналізу даних, щоб визначити найкращий напрямок дій.

Візуалізація даних

Візуалізація даних дуже поширена у вашому повсякденному житті; вони часто виступають у вигляді діаграм та графіків. Іншими словами, дані відображаються графічно, щоб людському мозку було легше їх зрозуміти та обробити. Візуалізація даних часто використовується для виявлення невідомих фактів та тенденцій. Спостерігаючи за

взаємозв'язками та порівнюючи масиви даних, ви можете знайти спосіб знайти значущу інформацію.

1.3. Порядок виконання роботи

Розглянемо практичну задачу обробки та аналізу даних з подальшою їх візуалізацією на прикладі передбачення розміру чайових в кафе (ресторані).

Крок 1. Завантажуємо бібліотеки

Для коректної роботи з даними в *python* потрібно завантажити спеціальну бібліотеку ***pandas***, програмну бібліотеку мовою *python* для обробки та аналізу даних. Бібліотека ***seaborn*** використовується для побудови графіків та візуалізації даних

```
import pandas as pd
import seaborn as sns
```

Крок 2. Завантаження набору даних

Завантажуємо набір даних для передбачення розміру чайових (поле *tip*) у ресторані (залежно від розміру рахунку, статі клієнта, часу доби (день чи вечір), дня тижня, розміру замовлення і було зроблено замовлення в залі для курців чи ні).

```
df = sns.load_dataset("tips")
```

Крок 3. Перегляд завантажених даних

Подивимося на перші 5 рядків у цьому наборі даних

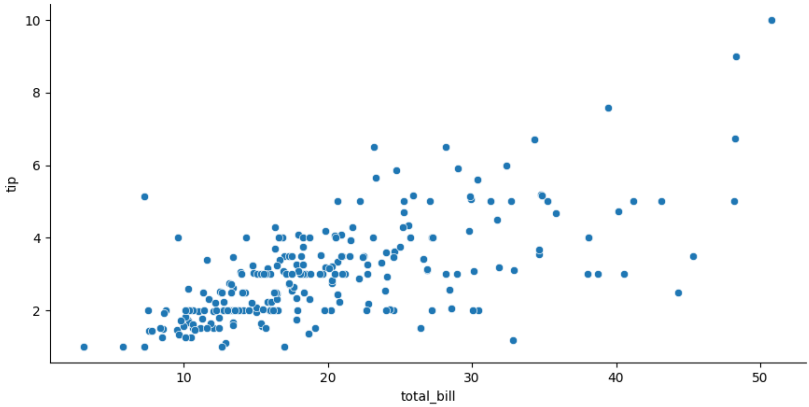
```
df.head()
```

```
   total_bill   tip     sex smoker  day    time  size
0       16.99  1.01  Female     No  Sun  Dinner     2
1       10.34  1.66    Male     No  Sun  Dinner     3
2       21.01  3.50    Male     No  Sun  Dinner     3
3       23.68  3.31    Male     No  Sun  Dinner     2
```

```
4        24.59  3.61  Female     No  Sun  Dinner     4
```

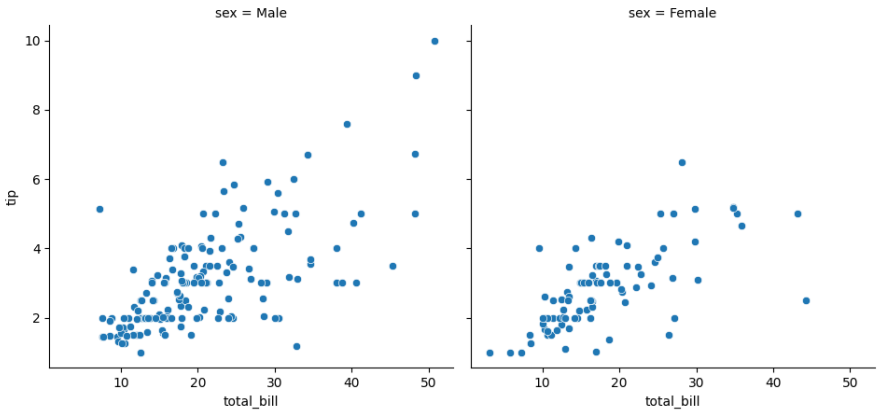
Перше завдання, яке можна вирішити за допомогою графіків – візуалізувати спільний розподіл двох величин, скажімо, залежність розміру чайових від загальної вартості замовлення:

```
sns.relplot(x='total_bill', y='tip', data=df)
```



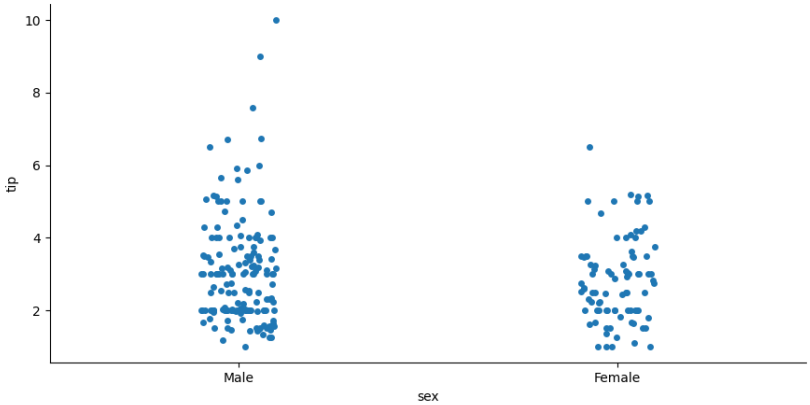
Чайові можуть дати як чоловіки, і жінки, тобто розмір чайових можна розділити на два підмножини:

```
sns.relplot(x='total_bill', y='tip', col='sex',
data=df);
```



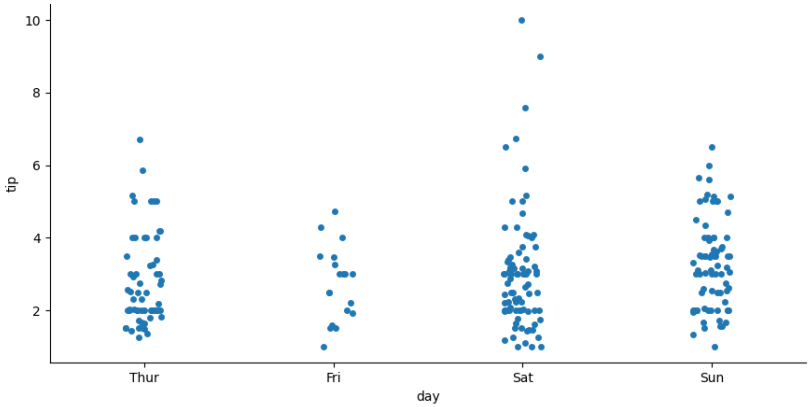
Як подивитися тільки на залежність розміру чайових від статі людини, яка сплатила замовлення? Без інформації про повну вартість досконалого замовлення? Це називається візуалізацією розкиду виміряних значень у кожній категорії:

```
sns.catplot(x='sex', y='tip', data=df);
```



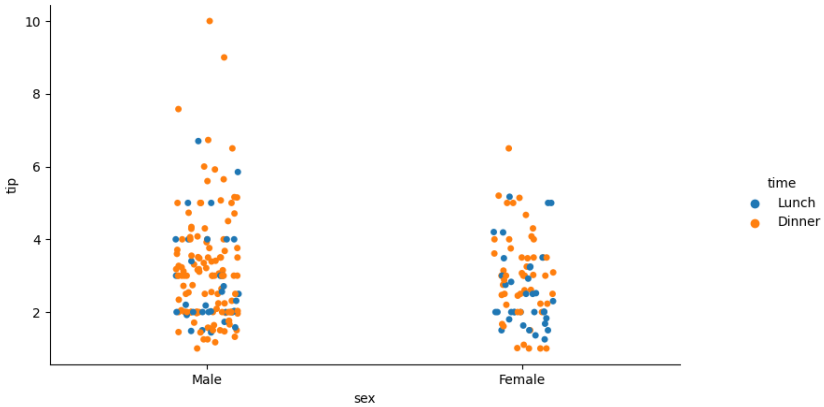
Аналогічно можна побудувати розподіл розміру чайових залежно від дня тижня:

```
sns.catplot(x='day', y='tip', data=df);
```



Можна подивитися, як на розмір чайових впливає не тільки стать людини, яка сплатила замовлення, але ще й час дня:

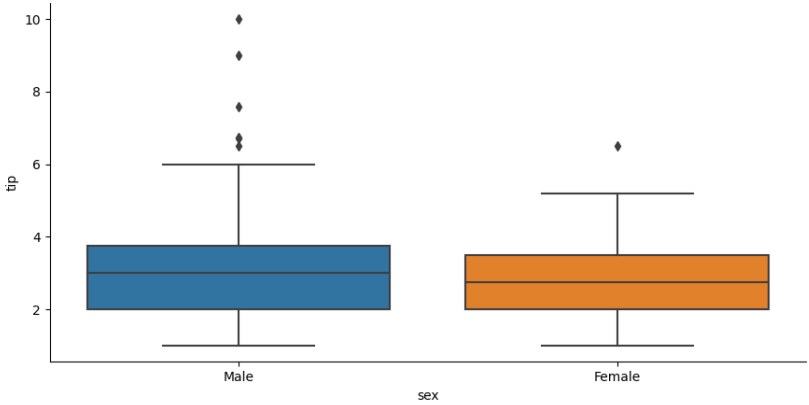
```
sns.catplot(x='sex', y='tip', hue='time', data=df);
```



Графік розкиду за категоріями зручний для невеликих наборів даних, оскільки зі збільшенням кількості точок вони однаково почнуть перекривати одне одного і зливатися. Щоб подолати ці труднощі, краще скористатися графіками, які самі містять деяку інформацію про розподіл усередині категорій. Один з таких графіків – це «скринька з вусами» або *boxplot*. Його можна

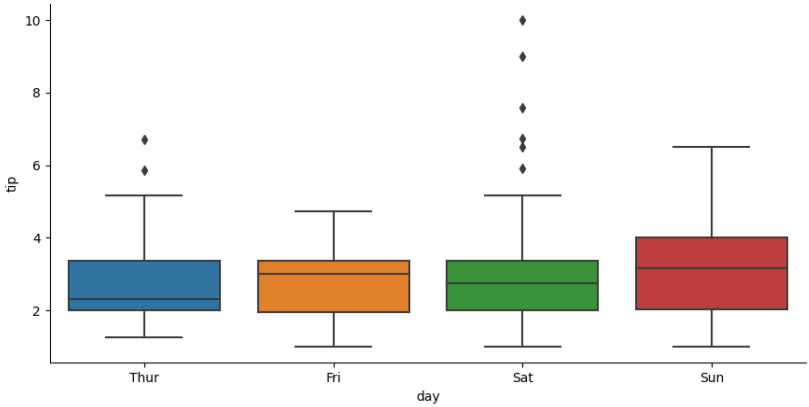
побудувати за допомогою тієї ж функції *catplot* з параметром *kind*, встановленим у *'box'*.

```
sns.catplot(x='sex', y='tip', kind='box', data=df);
```



Побудуємо тепер ящики з вусами по днях тижня.

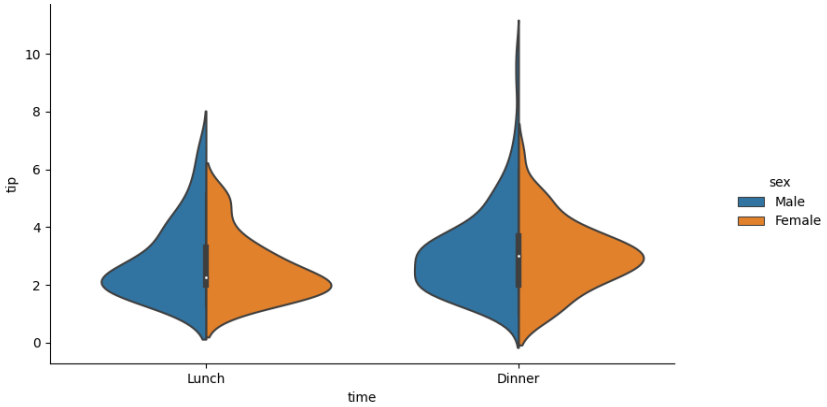
```
sns.catplot(x='day', y='tip', kind='box', data=df);
```



Ще один тип графіки, який дозволяє судити про розподіл значень – це *violinplot*. Широка і тонка чорна смуга всередині «віолончелі» відповідає «ящику» і «вусам» *boxplot*-а, а біла

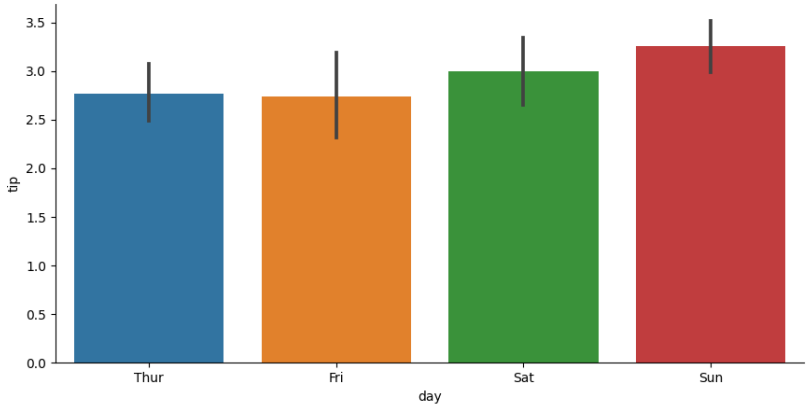
точка всередині – це медіана. Якщо дані складаються тільки з двох підмножин, то *violinplot* можна розділити навпіл для кожної з них за допомогою параметра *split*.

```
sns.catplot(x='time', y='tip', hue='sex',
kind='violin', split=True, data=df);
```



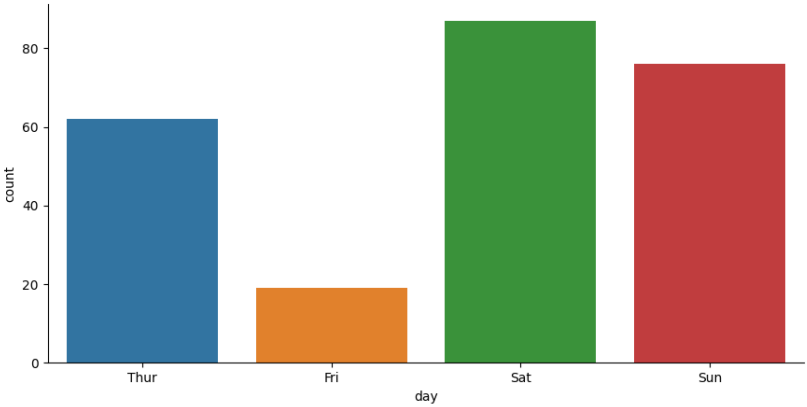
Стовпчасті діаграми теж можуть бути дуже зручними. У *Seaborn* функція *barplot*() за замовчуванням відображає середнє значення величини всередині кожної категорії та відображає її довірчий інтервал:

```
sns.catplot(x='day', y='tip', kind='bar', data=df);
```

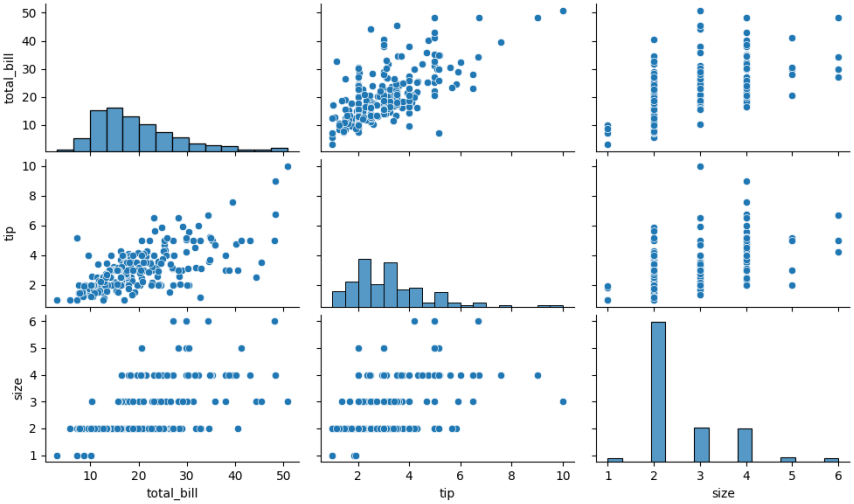


Щоб відобразити кількість спостережень всередині кожної категорії можна скористатися функцією *countplot*(), наприклад, ось так можна поглянути на кількість обслужених столиків по кожному дню:

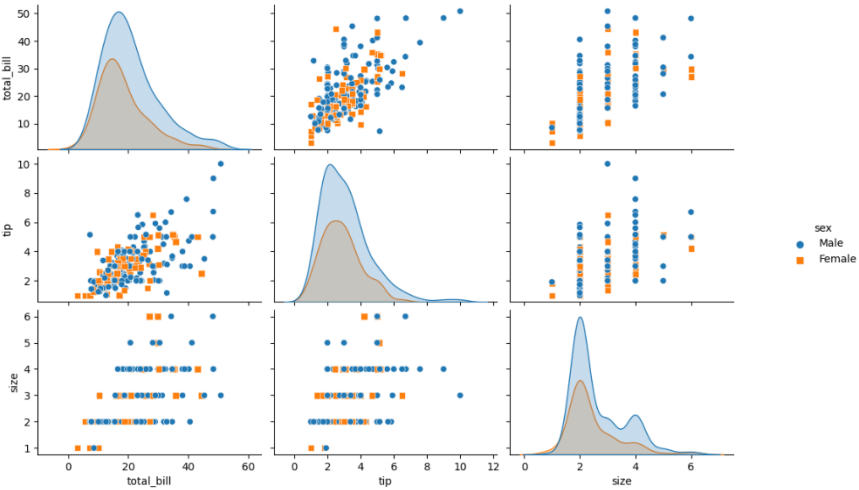
```
sns.catplot(x='day', kind='count', data=df);
```



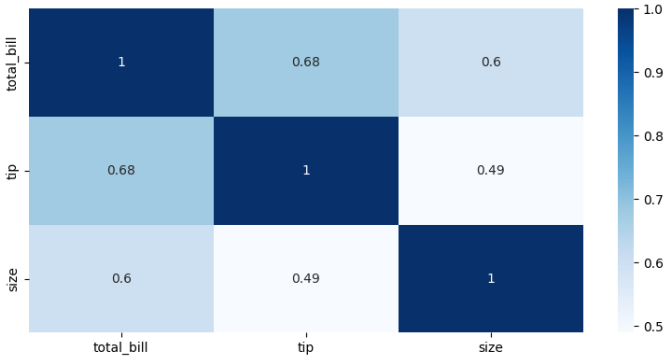

```
sns.pairplot(df)
```



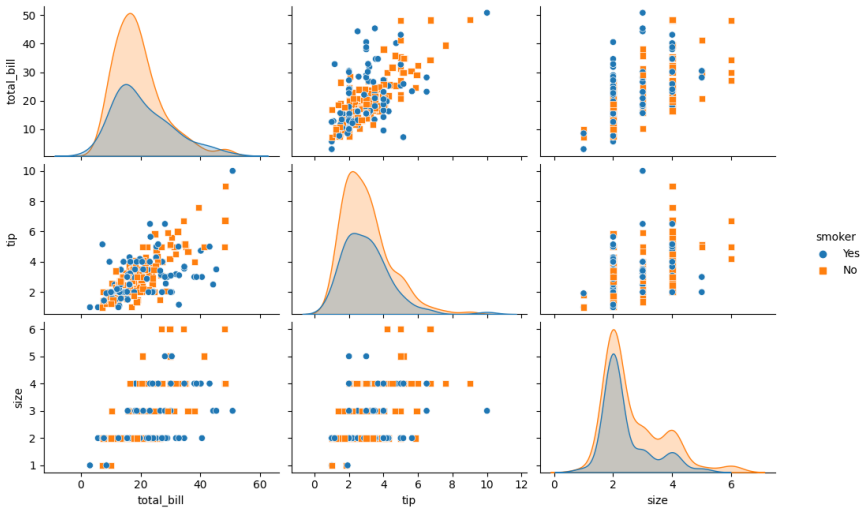

```
sns.pairplot(df ,hue ='sex', markers=["o", "s"])
```




```
sns.heatmap(df.corr(), annot=True, cmap = 'Blues')
```




```
sns.pairplot(df ,hue ='smoker', markers=["o", "s"])
```



Перевіряємо дані на наявність перепусток і дивимося типи змінних.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 244 entries, 0 to 243
```

```
Data columns (total 7 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   total_bill  244 non-null    float64
 1   tip         244 non-null    float64
 2   sex         244 non-null    category
 3   smoker      244 non-null    category
 4   day         244 non-null    category
 5   time        244 non-null    category
 6   size        244 non-null    int64
dtypes: category(4), float64(2), int64(1)
memory usage: 7.4 KB
```

Кодуємо категоріальні ознаки

Категоріальна ознака – це така ознака, яка може набувати одного значення з обмеженої кількості можливих. У наших даних 4 категоріальні ознаки, три з яких є бінарними. Нам необхідно закодувати їх, тобто перетворити на числові. Бінарні ознаки кодуються методом факторизації, який перетворює їх значення в 0 і 1. Ознака, що приймає більше 2 значень, закодуємо методом дамми-кодування. Розглянемо з прикладу. Нехай є категоріальна ознака *Category*, що приймає одне з чотирьох можливих значень [*Human', 'Penguin', 'Octopus', 'Alien'*]. Після застосування дамми-кодування ми отримаємо чотири нових ознаки (за кількістю можливих значень) *Category_Human*, *Category_Penguin*, *Category_Octopus*, *Category_Alien*. Для того рядка, у якої у вихідних даних стояла категорія *Human*, у стовпці *Category_Human* стоятиме 1, в інших стовпцях 0. Аналогічно іншого значення.

```
# вказуємо залежну (вихідну) змінну
y =df['tip']
# вказуємо незалежні (вхідні) змінні
X=df.drop(['tip'], axis=1)
# кодуємо бінарні ознаки нулями та одиницями
X['sex'] = pd.factorize(X['sex'])[0]
X['smoker'] = pd.factorize(X['smoker'])[0]
X['time'] = pd.factorize(X['time'])[0]
# кодуємо поле day методом даммі-кодування
```

```
X = pd.concat([X, pd.get_dummies(X['day'],
prefix="day")], axis=1)
# видаляємо старе поле day
X.drop(['day'], axis=1, inplace=True)
```

Дивимося, як змінилися дані після кодування.

```
X.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 244 entries, 0 to 243
Data columns (total 9 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   total_bill  244 non-null    float64
 1   sex         244 non-null    int64
 2   smoker      244 non-null    int64
 3   time        244 non-null    int64
 4   size        244 non-null    int64
 5   day_Thur    244 non-null    uint8
 6   day_Fri     244 non-null    uint8
 7   day_Sat     244 non-null    uint8
 8   day_Sun     244 non-null    uint8
dtypes: float64(1), int64(4), uint8(4)
memory usage: 10.6 KB
```

Бібліотека ***scikit-learn*** – де-факто найбільш популярний, різнобічний, добре документований інструмент, що постійно збагачується, для побудови моделей машинного навчання.

Виберемо кілька готових функцій.

```
from sklearn.model_selection import train_test_split
# Розбиття набору даних на навчальну та тестову частини
у співвідношенні 3:1.
X_train, X_test, y_train, y_test = train_test_split(X,
y, test_size=0.20, random_state=21)
```

DecisionTreeRegressor – процедура для побудови дерев рішень у задачах регресії. *DecisionTreeClassifier* – процедура для побудови дерев рішень у задачах класифікації.

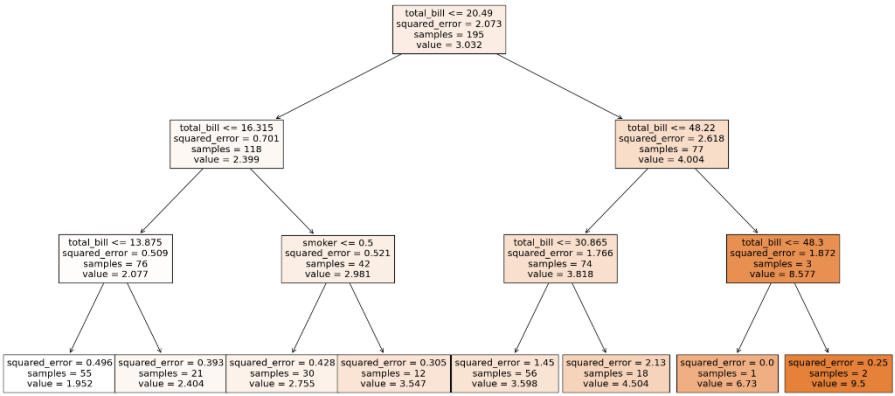
plot_tree – процедура для візуалізації дерев.

```
from sklearn.tree import DecisionTreeRegressor,
plot_tree
# Побудова дерева рішень. max_depth - максимальна
глибина дерева
dectree = DecisionTreeRegressor(max_depth=3,
random_state=21)
# навчаємо на навчальній вибірці
dectree.fit(X_train,y_train)
#визначаємо точність R^2 на тестовій вибірці
dectree.score(X_test,y_test)
```

```
0.4002336753388297
```

Малюємо дерево *matplotlib.pyplot* – базова бібліотека для візуалізації, що використовується тут для завдання загальних розмірів картинки. Процедура *plot_tree* безпосередньо малює дерево.

```
import matplotlib.pyplot as plt
plt.figure(figsize=((25,10)))
plot_tree(dectree,
          filled=True,
          feature_names=X.columns)
plt.show()
```



Ліва стрілка – так, права – ні. *Samples* – кількість прикладів з навчальної вибірки, що потрапили у цей вузол. *Value* – переказане значення чайових. *MSE* – величина середньоквадратичної помилки (середнє значення квадрата різниці між передбаченими чайовими та тими, що були насправді):

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

```
from sklearn.ensemble import RandomForestRegressor
rfc = RandomForestRegressor(n_estimators = 100,
max_depth=3, random_state=21)
rfc.fit(X_train, y_train)
rfc.score (X_test, y_test)
```

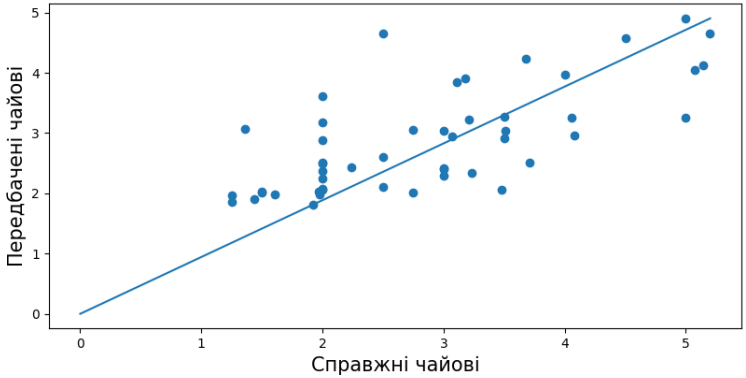
```
0.49200337705729547
```

```
from sklearn.metrics import mean_absolute_error
y_pred=rfc.predict(X_test)
mean_absolute_error(y_test,y_pred)
```

```
0.6065327286651406
```

```
plt.figure(figsize=(7, 7))
```

```
plt.scatter(y_test,y_pred) # малюємо точки, що
відповідають парам справжнє значення - прогноз
plt.plot([0, max(y_test)], [0, max(y_pred)]) # малюємо
пряму, на якій передбачення та справжні значення
збігаються
plt.xlabel('Справжні чайові', fontsize=15)
plt.ylabel('Передбачені чайові', fontsize=15);
plt.show()
```



1.4. Контрольні запитання

1. Опишіть функціонал бібліотек, що були використані в лабораторній роботі.
2. Які типи графіків були застосовані? Вкажіть їх особливі риси.
3. Що таке категоріальна ознака?
4. Вкажіть як саме будується дерево бібліотеки *matplotlib.pyplot*?
5. Напишіть поетапно, як саме виконується завдання прогнозування.

Лабораторна робота №2. Побудова моделі класифікації

2.1. Мета роботи

Побудова моделі класифікації співробітників: на вхід модель прийматиме дані про співробітника, а на виході вона повинна працювати у двох режимах:

- видавати ймовірність догляду для цього співробітника,
- видавати правильний з погляду моделі клас співробітника (не задоволений і збирається піти чи все влаштовує та залишається).

2.2. Теоретичні відомості

Ключ до успіху будь-якої організації – це залучення та утримання найкращих співробітників. Для HR-аналітиків одним із завдань є запобігання догляду персоналу. Для цього їм необхідно знати фактори, що збільшують залучення співробітників, і, навпаки, спонукають співробітників піти.

Постановка задачі аналізу даних

Навчати модель машинного навчання, яка передбачатиме відтік співробітників ми будемо за даними платформи *kaggle*.

https://www.kaggle.com/patelprashant/employee-attrition

Огляд доступних даних

У вибірці 3308 історичних спостережень та 20 змінних, одна з яких – цільова. Таким чином, про кожного з 3308 співробітників ми знаємо значення 20 їх характеристик (вік, стать, місце та область роботи, відстань від будинку тощо), у тому числі значення цільової змінної: пішов співробітник чи ні.

Доступні ознаки

Дані містять два типи змінних:

- Цільова: *Attrition*, пішов працівник чи ні
- Інші змінні: 19 змінних можуть використовуватися для прогнозу цільової змінної.

Ім'я стовпця	Значення	Ім'я стовпця	Значення
Age	Вік	*NumCompaniesWorked*	Кількість компаній, у яких працював співробітник
BusinessTravel	Частота відряджень	*PercentSalaryHike*	Відсоток підвищення з/п за час роботи
Department	Відділ	*StandardHours*	Стандартна тривалість робочого дня
DistanceFromHome	Відстань від будинку в км	*StockOptionLevel*	Рівень опціону на акції
Education	Рівень освіти	*TotalWorkingYears*	Загальний трудовий стаж
EducationField	Сфера освіти	*TrainingTimesLastYear*	Загальний час додаткового навчання
Gender	Стать	*YearsAtCompany*	Стаж роботи у цій компанії
JobRole	Посадова роль	*YearsSinceLastPromotion*	Кількість років з останнього підвищення
MaritalStatus	Сімейний стан	*YearsWithCurrManager*	Кількість років роботи з поточним менеджером
MonthlyIncome	Щомісячний дохід	*Attrition*	Цільова змінна: пішов працівник чи ні

План аналізу даних (*data mining*):

1. Завантажити дані для навчання
2. Обробити дані перед навчанням моделі
3. Навчити модель на навчальній вибірці
4. Завантажити та обробити дані для тестування
5. Провалідувати модель на тестовій вибірці

2.3. Порядок виконання роботи

1. Завантажити дані для навчання

Крок 1.1. Завантажуємо бібліотеки

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

from sklearn.metrics import roc_curve, auc # ROC-крива
from sklearn.metrics import confusion_matrix # матриця
невідповідностей
```

Будемо використовувати бібліотеку *MatplotLib*, яка чудово справляється з цим завданням. Модуль *seaborn* встановлює «приємні для очей» палітри і стилі для графіків.

```
import seaborn as sns
sns.set_style('whitegrid') #встановлення стилю
```

Деякі бібліотеки можуть виводити на екран попередження, які важливо враховувати у «бойових» завданнях. Але ми зазначимо ігнорувати їх.

```
import warnings
warnings.filterwarnings("ignore")
```

Крок 1.2. Завантажимо дані

Для виконання завдання ми будемо використовувати дані у форматі *.csv – comma separated values*, значення, розділені комами. Щоб завантажити такі дані, використовуватимемо функцію *read_csv()* з бібліотеки *pandas*.

```
training_data =
pd.read_csv("https://www.dropbox.com/s/jaea4smqjbmslsb/
training_data.csv?dl=1 ")
```

Подивимося на 5 випадково вибраних записів з навчального набору, для цього використовуватимемо функцію *sample()*. Параметр `random_state=123` фіксує «випадковість», тобто на

будь-якому комп'ютері метод *sample()* працюватиме однаково.

```
training_data.sample(5, random_state=123)
```

```
      Age     BusinessTravel  ... YearsWithCurrManager  Attrition
1930   45         Non-Travel  ...                    8          0
909    35      Travel_Rarely  ...                    2          1
222    30  Travel_Frequently  ...                    0          1
2307   33      Travel_Rarely  ...                    7          0
576    34      Travel_Rarely  ...                    3          0

[5 rows x 20 columns]
```

Крок 1.3. Подивимося загальну статистику за даними

Подивімося на технічні параметри завантажених даних для навчання. Для цього викличемо метод *describe()* для набору даних *training_data*.

Для зручності відображення ми транспонуємо результат.

```
training_data.describe().T
```

```
                          count          mean  ...      75%       max
Age                      3308.0     36.879383  ...     43.0      60.0
DistanceFromHome         3308.0      9.201935  ...     14.0      29.0
Education                3308.0      2.918380  ...      4.0       5.0
MonthlyIncome            3308.0  64594.903265  ...  83210.0  199990.0
NumCompaniesWorked       3293.0      2.703310  ...      4.0       9.0
PercentSalaryHike        3308.0     15.230048  ...     18.0      25.0
StandardHours            3308.0      8.000000  ...      8.0       8.0
StockOptionLevel         3308.0      0.804716  ...      1.0       3.0
TotalWorkingYears        3301.0     11.283248  ...     15.0      40.0
TrainingTimesLastYear    3308.0      2.801995  ...      3.0       6.0
YearsAtCompany           3308.0      6.998791  ...      9.0      40.0
YearsSinceLastPromotion  3308.0      2.187122  ...      3.0      15.0
YearsWithCurrManager     3308.0      4.118501  ...      7.0      17.0
Attrition                3308.0      0.160218  ...      0.0       1.0

[14 rows x 8 columns]
```

Звернімо увагу на загальні статистики показників у даних:

- *count* - кількість значень, які є пропущеними (*NaN*);
- *mean*, *std* - середнє та розкид даних у відповідному полі;
- Інші статистики - мінімальне і максимальне значення, і квантили.

З таких характеристик стовпців ми вже можемо отримати

деяку інформацію про дані:

- У стовпця *Attrition* середнє 0,16. Отже, у нашій вибірці лише 16% працівників пішли з компанії.
- У стовпця *MonthlyIncome* суттєва амплітуда значень: мінімальне значення доходу – 10090, максимальне – 199990.
- Стовпці *NumCompaniesWorked* і *TotalWorkingYears* мають пропуски.

2. Обробити дані перед навчанням моделі

Крок 2.1. Перевіряємо дані на наявність пропусків і типів змінних

Почнемо з перевірки загальної інформації про дані. Щоб це зробити, потрібно викликати у змінній *training_data* метод *info()*.

```
training_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3308 entries, 0 to 3307
Data columns (total 20 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   Age                      3308 non-null   int64
 1   BusinessTravel           3308 non-null   object
 2   Department               3308 non-null   object
 3   DistanceFromHome         3308 non-null   int64
 4   Education                3308 non-null   int64
 5   EducationField           3308 non-null   object
 6   Gender                   3308 non-null   object
 7   JobRole                  3308 non-null   object
 8   MaritalStatus            3308 non-null   object
 9   MonthlyIncome            3308 non-null   int64
 10  NumCompaniesWorked       3293 non-null   float64
 11  PercentSalaryHike        3308 non-null   int64
 12  StandardHours            3308 non-null   int64
 13  StockOptionLevel         3308 non-null   int64
 14  TotalWorkingYears        3301 non-null   float64
 15  TrainingTimesLastYear    3308 non-null   int64
 16  YearsAtCompany           3308 non-null   int64
 17  YearsSinceLastPromotion  3308 non-null   int64
 18  YearsWithCurrManager     3308 non-null   int64
 19  Attrition                3308 non-null   int64
dtypes: float64(2), int64(12), object(6)
memory usage: 517.0+ KB
```

Як ми вже бачили, пропусків у даних не так багато: тільки у

стовпців *NumCompaniesWorked* і *TotalWorkingYears*.

Крок 2.2. Заповнення пропусків

Розрахуємо середні значення ознак у навчальній вибірці, і заповнимо отриманими числами пропуски як і тестовому наборі даних, і у навчальній вибірці.

Для отримання середніх значень викличемо метод *mean()*. За умовчанням метод вважає середнім значення по стовпчикам. Після виконання комірки середні значення записані у змінній *train_mean*.

```
train_mean = training_data.mean()
train_mean
```

```
Age                          36.879383
DistanceFromHome              9.201935
Education                     2.918380
MonthlyIncome             64594.903265
NumCompaniesWorked            2.703310
PercentSalaryHike            15.230048
StandardHours                 8.000000
StockOptionLevel              0.804716
TotalWorkingYears            11.283248
TrainingTimesLastYear         2.801995
YearsAtCompany                6.998791
YearsSinceLastPromotion       2.187122
YearsWithCurrManager          4.118501
Attrition                     0.160218
dtype: float64
```

Пропуски даних можна заповнювати і різними методами:

- вибірковою статистикою (середнє, медіана);
- прогнозами регресії за відомими ознаками;
- випадковими значеннями.

Якщо відносно невелика частка спостереженню має пропуск, можна зовсім виключити неповні спостереження з пропущеними значеннями з вибірки.

Для заповнення середнім значенням передамо на вхід методу *fillna()* отриманий раніше набір середніх значень для кожного стовпця. Опція `inplace=True` каже, що ми запишемо зміни

прямо до існуючого масиву, а не створимо новий.

```
training_data.fillna(train_mean, inplace=True)
```

Переконаємося, що пробілів більше немає.

```
training_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3308 entries, 0 to 3307
Data columns (total 20 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   Age                      3308 non-null   int64
 1   BusinessTravel           3308 non-null   object
 2   Department               3308 non-null   object
 3   DistanceFromHome         3308 non-null   int64
 4   Education                3308 non-null   int64
 5   EducationField           3308 non-null   object
 6   Gender                   3308 non-null   object
 7   JobRole                  3308 non-null   object
 8   MaritalStatus            3308 non-null   object
 9   MonthlyIncome            3308 non-null   int64
 10  NumCompaniesWorked       3308 non-null   float64
 11  PercentSalaryHike        3308 non-null   int64
 12  StandardHours            3308 non-null   int64
 13  StockOptionLevel         3308 non-null   int64
 14  TotalWorkingYears        3308 non-null   float64
 15  TrainingTimesLastYear    3308 non-null   int64
 16  YearsAtCompany           3308 non-null   int64
 17  YearsSinceLastPromotion  3308 non-null   int64
 18  YearsWithCurrManager     3308 non-null   int64
 19  Attrition                3308 non-null   int64
dtypes: float64(2), int64(12), object(6)
memory usage: 517.0+ KB
```

Крок 2.3. Працюємо з цільовою змінною

Яка змінна цільова?

В даному випадку за умовою завдання ми повинні прогнозувати відхід співробітника, тому цільова змінна - це *Attrition*, факт наявності звільнення співробітника.

Нам потрібно виділити в окрему змінну у стовпець з нашої таблиці, який відповідає певній цільовій змінній. Для цього ми у таблиці *training_data* у квадратних дужках вказуємо ім'я потрібного. У нашому випадку це ім'я записано у змінній

target_variable_name.

```
y = training_data['Attrition']
```

Щоб порахувати кількість співробітників, які звільнилися (значення змінної *Attrition* дорівнює одиниці) та співробітників, які залишилися (значення *Attrition* дорівнює нулю), викличемо метод *value_counts()*.

```
y.value_counts()
```

```
0    2778
1     530
Name: Attrition, dtype: int64
```

Відокремимо вхідні змінні від вихідних (цільових), щоб можна було побудувати модель передбачення цільової змінної по вхідним даним. Для цього необхідно у змінної *training_data* викликати спосіб *drop()*.

```
X = training_data.drop('Attrition', axis=1)
```

Видно, що стовпця справді немає, а кількість рядків не змінилася.

Крок 3. Обробити текстові змінні

У даних 6 стовпців мають значення типу *object*. У нашому випадку це текстові ознаки. Щоб можна було подавати їх на вхід до алгоритму, нам необхідно закодувати їх.

Ми розглянемо два популярні методи кодування категоріальних ознак:

- метод *get_dummies()* із бібліотеки *pandas*
- метод *factorize()* з бібліотеки *pandas*

Для початку нам необхідно визначити які з ознак категоріальні:

```
text_features = ['BusinessTravel', 'Department',
'EducationField',  'JobRole', 'MaritalStatus']
```

Далі нам необхідно перетворити значення у вибраних колонках на числа, тому що дані алгоритми не вміють працювати з текстом.

Перший спосіб полягає в тому, щоб замінити ознаки типу:

	Відділ
Співробітник 1	Sales
Співробітник 2	RnD
Співробітник 3	HR

Ознаками типу:

	Відділ - Sales?	Відділ - Rnd?	Відділ - HR?
Співробітник 1	Так (1)	Ні (0)	Ні (0)
Співробітник 2	Ні (0)	Так (1)	Ні (0)
Співробітник 3	Ні (0)	Ні (0)	Так (1)

Таким чином, наші нові ознаки говорять, чи ставиться співробітник до певного відділу чи ні. За такого підходу кожному за категоріального ознаки з'являється стільки нових колонок, скільки є можливих категорій. Одна із колонок буде заповнена 1, а решта 0.

```
X_dummies = pd.get_dummies(data=X,
columns=text_features)
```

У функцію *get_dummies()* як параметри передаємо:

- *data* - вихідні дані

- *columns* - імена колонок, у яких перебувають кагегоріальні ознаки

```
X_dummies.head()
```

```
   Age  DistanceFromHome  ...  MaritalStatus_Married MaritalStatus_Single
0   51                 6  ...                      1                   0
1   31                10  ...                      0                   1
2   32                17  ...                      1                   0
3   38                 2  ...                      1                   0
4   32                10  ...                      0                   1

[5 rows x 38 columns]
```

Другий спосіб кодування категоріальних ознак полягає в тому, щоб просто привласнити кожній категорії унікальний номер, та замінити значення ознак на відповідні номери. Наприклад, замість таблиці:

	Відділ
Співробітник 1	Sales
Співробітник 2	RnD
Співробітник 3	RnD
Співробітник 4	Sales
Співробітник 5	HR
Співробітник 6	RnD

Отримуємо:

	Відділ
Співробітник 1	1
Співробітник 2	2
Співробітник 3	2
Співробітник 4	1
Співробітник 5	3
Співробітник 6	2

Тут усі значення *Sales* замінено на 1, *RnD* замінено на 2, *HR* замінено на 3.

```
X_dummies['Gender'] = pd.factorize(X['Gender'])[0]
X_dummies.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3308 entries, 0 to 3307
Data columns (total 38 columns):
 #   Column                              Non-Null Count  Dtype
---  ------                              --------------  -----
 0   Age                                 3308 non-null   int64
 1   DistanceFromHome                    3308 non-null   int64
 2   Education                           3308 non-null   int64
 3   Gender                              3308 non-null   int64
 4   MonthlyIncome                       3308 non-null   int64
 5   NumCompaniesWorked                  3308 non-null   float64
 6   PercentSalaryHike                   3308 non-null   int64
 7   StandardHours                       3308 non-null   int64
 8   StockOptionLevel                    3308 non-null   int64
 9   TotalWorkingYears                   3308 non-null   float64
 10  TrainingTimesLastYear               3308 non-null   int64
 11  YearsAtCompany                      3308 non-null   int64
 12  YearsSinceLastPromotion             3308 non-null   int64
 13  YearsWithCurrManager                3308 non-null   int64
 14  BusinessTravel_Non-Travel           3308 non-null   uint8
 15  BusinessTravel_Travel_Frequently    3308 non-null   uint8
 16  BusinessTravel_Travel_Rarely        3308 non-null   uint8
 17  Department_Human Resources          3308 non-null   uint8
 18  Department_Research & Development   3308 non-null   uint8
 19  Department_Sales                    3308 non-null   uint8
 20  EducationField_Human Resources      3308 non-null   uint8
 21  EducationField_Life Sciences        3308 non-null   uint8
 22  EducationField_Marketing            3308 non-null   uint8
 23  EducationField_Medical              3308 non-null   uint8
 24  EducationField_Other                3308 non-null   uint8
 25  EducationField_Technical Degree     3308 non-null   uint8
 26  JobRole_Healthcare Representative   3308 non-null   uint8
 27  JobRole_Human Resources             3308 non-null   uint8
 28  JobRole_Laboratory Technician       3308 non-null   uint8
 29  JobRole_Manager                     3308 non-null   uint8
 30  JobRole_Manufacturing Director      3308 non-null   uint8
 31  JobRole_Research Director           3308 non-null   uint8
 32  JobRole_Research Scientist          3308 non-null   uint8
 33  JobRole_Sales Executive             3308 non-null   uint8
 34  JobRole_Sales Representative        3308 non-null   uint8
 35  MaritalStatus_Divorced              3308 non-null   uint8
 36  MaritalStatus_Married               3308 non-null   uint8
 37  MaritalStatus_Single                3308 non-null   uint8
dtypes: float64(2), int64(12), uint8(24)
memory usage: 439.5 KB
```

4. Поділимо вибірку на навчальну та тестову

```
from sklearn.model_selection import train_test_split
# Розбиття набору даних на навчальну та тестову частини
у співвідношенні 3:1.
X_train, X_test, y_train, y_test =
train_test_split(X_dummies, y, test_size=0.25,
random_state=42)
```

5. Навчити модель на навчальній вибірці

Крок 5.1. Вибираємо метод, який використовуватимемо

- Градієнтний бустинг над вирішальними деревами *xgboost* *

У бібліотеці *xgboost*, програмній бібліотеці мовою *python*, реалізовано градієнтний бустинг над вирішальними деревами як завданням регресії, так завданням класифікації.

```
import xgboost as xgb
```

У моделі *xgboost* три типи параметрів: загальні параметри, параметри бустера та параметри завдання.

Загальні параметри відносяться до того, який бустер ми використовуємо: дерево або лінійна модель. Параметри бустера залежать від того, який бустер ви вибрали (максимальна глибина дерева, кількість дерев в ансамблі тощо)

```
xgboost_model = xgb.XGBClassifier(n_estimators=40,
max_depth=15)
```

Розглянемо найважливіші параметри:

- параметр *n_estimators* визначає, скільки дерев в ансамблі,
- у параметрі *max_depth* встановлюється, яка максимальна глибина дерева,

Крок 5.2. Навчити модель

```
xgboost_model.fit(X_train, y_train)
```

```
XGBClassifier(base_score=0.5, booster='gbtree', callbacks=None,
              colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1,
              early_stopping_rounds=None, enable_categorical=False,
              eval_metric=None, gamma=0, gpu_id=-1, grow_policy='depthwise',
              importance_type=None, interaction_constraints='',
              learning_rate=0.300000012, max_bin=256, max_cat_to_onehot=4,
              max_delta_step=0, max_depth=15, max_leaves=0, min_child_weight=1,
              missing=nan, monotone_constraints='()', n_estimators=40, n_jobs=0,
              num_parallel_tree=1, predictor='auto', random_state=0,
              reg_alpha=0, reg_lambda=1, ...)
```

6. Провалідувати модель на тестовій вибірці

```
test_predictions_xgboost =
xgboost_model.predict(X_test)
```

Крок 6.1. Точність прогнозу

```
xgboost_model.score(X_test, y_test)
```

```
0.966142684401451
```

Крок 6.2. Матриця помилок

Нагадаєм, *Confusion matrix* містить зведені показники якості роботи класифікатора. Рядки цієї таблиці відповідають фактичним класам тестового набору, а стовпці – міткам, передбаченим класифікатором.

```
from sklearn.metrics import confusion_matrix
```

Таблиця містить чотири зведені показники, кожен з яких відображає кількість об'єктів в одній із чотирьох категорій:

- істинно позитивний (*True positive*, *TP*) - об'єкт класу 1 був чітко позначений міткою 1;
- хибно позитивний (*False positive*, *FP*) - об'єкт фактично належить класу 0, але позначений міткою 1;
- істинно негативний (*True negative*, *TN*) - класифікатор чітко визначив, що об'єкт класу 0 належить класу 0;
- помилково негативний (*False negative*, *FN*) - класифікатор помітив об'єкт міткою 0, проте насправді об'єкт належить

класу 1.

Примітка: помилки False positive часто називають помилковою тривогою, а False negative – пропуском цілі.

```
                    Передбачено 0  Передбачено 1
        Фактично 0        TN             FP
        Фактично 1        FN             TP
```

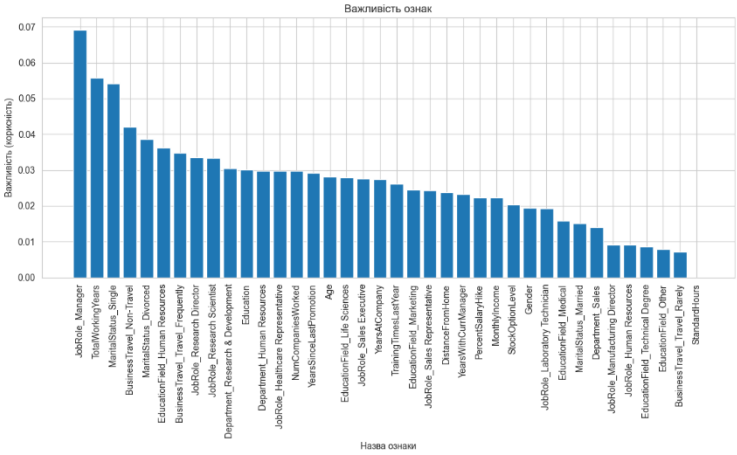
```
xgboost_confusion_matrix = confusion_matrix(y_test,
test_predictions_xgboost)
xgboost_confusion_matrix =
pd.DataFrame(xgboost_confusion_matrix)
xgboost_confusion_matrix
```

```
     0    1
0  676    2
1   26  123
```

*** Визначення важливості ознак**

Очевидно, що не всі зібрані ознаки будуть однаково корисними. Після навчання алгоритму ми можемо подивитися, які з ознак більше впливають на результат. Якщо в наборі даних виявляться даремні ознаки, їх можна вилучити, щоб зменшити час навчання. Подивитися на важливість ознак в алгоритмах бібліотеки *sklearn* можна за допомогою властивості `feature_importances_`.

```
importances = xgboost_model.feature_importances_
indices = np.argsort(importances)[::-1]

plt.figure(figsize=(10, 5))

plt.title('Важливість ознак')
plt.ylabel('Важливість (корисність)')
plt.xlabel('Назва ознаки')

plt.bar(X_test.columns[indices], importances[indices])
plt.xticks(rotation=90)
plt.show()
```



Градієнтний бустинг над вирішальними деревами *catboost* *.

```
pip install catboost
```

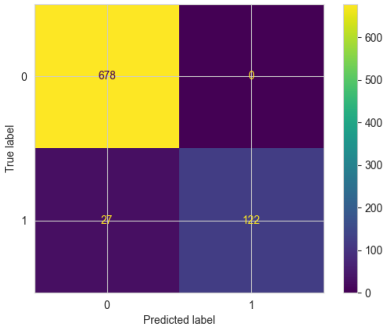
```
Collecting catboost
  Using cached catboost-1.1-cp310-none-win_amd64.whl (73.9 MB)
Requirement already satisfied: pandas>=0.24.0 in
c:\users\shich\appdata\local\programs\python\python310\lib\site-packages (from
catboost) (1.5.1)
Requirement already satisfied: plotly in
c:\users\shich\appdata\local\programs\python\python310\lib\site-packages (from
catboost) (5.10.0)
Requirement already satisfied: six in
c:\users\shich\appdata\local\programs\python\python310\lib\site-packages (from
catboost) (1.16.0)
Requirement already satisfied: numpy>=1.16.0 in
c:\users\shich\appdata\local\programs\python\python310\lib\site-packages (from
catboost) (1.22.4)
Requirement already satisfied: graphviz in
c:\users\shich\appdata\local\programs\python\python310\lib\site-packages (from
catboost) (0.20.1)
Requirement already satisfied: scipy in
c:\users\shich\appdata\local\programs\python\python310\lib\site-packages (from
catboost) (1.8.1)
Requirement already satisfied: matplotlib in
c:\users\shich\appdata\local\programs\python\python310\lib\site-packages (from
catboost) (3.6.1)
Requirement already satisfied: python-dateutil>=2.8.1 in
c:\users\shich\appdata\local\programs\python\python310\lib\site-packages (from
pandas>=0.24.0->catboost) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in
c:\users\shich\appdata\local\programs\python\python310\lib\site-packages (from
pandas>=0.24.0->catboost) (2022.5)
Requirement already satisfied: pillow>=6.2.0 in
c:\users\shich\appdata\local\programs\python\python310\lib\site-packages (from
matplotlib->catboost) (9.2.0)
Requirement already satisfied: packaging>=20.0 in
c:\users\shich\appdata\local\programs\python\python310\lib\site-packages (from
matplotlib->catboost) (21.3)
Requirement already satisfied: cycler>=0.10 in
c:\users\shich\appdata\local\programs\python\python310\lib\site-packages (from
matplotlib->catboost) (0.11.0)
Requirement already satisfied: kiwisolver>=1.0.1 in
c:\users\shich\appdata\local\programs\python\python310\lib\site-packages (from
matplotlib->catboost) (1.4.4)
Requirement already satisfied: pyparsing>=2.2.1 in
c:\users\shich\appdata\local\programs\python\python310\lib\site-packages (from
matplotlib->catboost) (3.0.9)
Requirement already satisfied: fonttools>=4.22.0 in
c:\users\shich\appdata\local\programs\python\python310\lib\site-packages (from
matplotlib->catboost) (4.37.4)
Requirement already satisfied: contourpy>=1.0.1 in
c:\users\shich\appdata\local\programs\python\python310\lib\site-packages (from
matplotlib->catboost) (1.0.5)
Requirement already satisfied: tenacity>=6.2.0 in
c:\users\shich\appdata\local\programs\python\python310\lib\site-packages (from plotly-
>catboost) (8.1.0)
Installing collected packages: catboost
Successfully installed catboost-1.1

[notice] A new release of pip available: 22.2.2 -> 22.3
[notice] To update, run: python.exe -m pip install --upgrade pip
```

```
from catboost import CatBoostClassifier

cat_model = CatBoostClassifier(
    random_seed = 42,
    logging_level = 'Silent')

X_train1, X_test1, y_train1, y_test1 =
train_test_split(X, y, test_size=0.25, random_state=42)
```

```
categorical_features_indices = np.where(X_train1.dtypes
!= np.float)[0]
cat_model.fit( X_train1, y_train1, cat_features =
categorical_features_indices)

cat_model.score(X_test1, y_test1)
```

```
0.9673518742442564
```

```
test_predictions_catboost = cat_model.predict(X_test1)

from sklearn.metrics import ConfusionMatrixDisplay
cm = confusion_matrix(y_test1,
test_predictions_catboost, labels=cat_model.classes_)
cm_display =
ConfusionMatrixDisplay(confusion_matrix=cm,
display_labels=cat_model.classes_)
cm_display.plot()
```



2.4. Контрольні запитання

1. Які ви знаєте методи відображення інформації з *.csv* файлу?
2. Яким чином на даній лекції було вирішено проблему заповнення пропусків?
3. Для відокремлення основної змінної з якою Вам треба працювати потрібно …
4. Як саме була вирішена проблема нездатності алгоритмів зчитувати текст в якості значень параметрів?
5. Опишіть процес визначення важливості ознак.

Лабораторна робота №3. Баєсовий аналіз у Python

3.1. Мета роботи

Вивчити принципи Баєсових методів. Навчитися використовувати їх у практичних завданнях. Виконати завдання оптимізації параметрів та порівняти результат Баєсового аналізу та методу найменших квадратів.

3.2. Теоретичні відомості

Введення в Баєсові методи

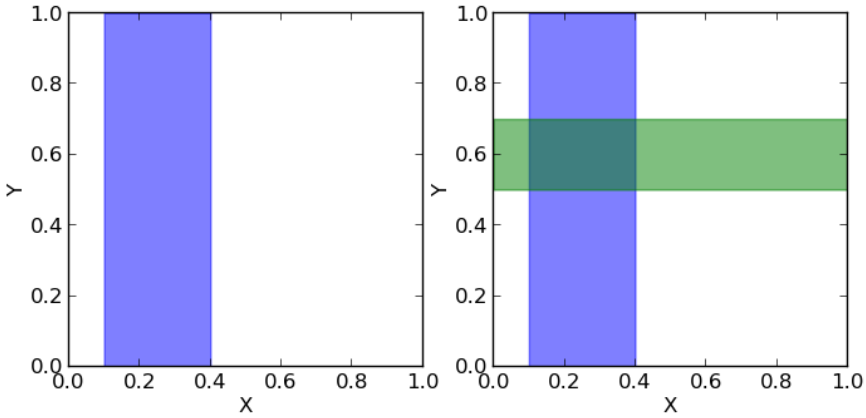
В даний час Баєсові методи набули досить широкого поширення і активно використовуються в різних галузях знань. Однак, на жаль, не так багато людей мають уявлення про те, що це таке і навіщо це потрібно.

Формула Баєса була опублікована у 1763 році через 2 роки після смерті її автора, Томаса Баєса. Однак, методи, що її використовують, набули дійсно широкого поширення лише до кінця XX століття. Це пояснюється тим, що розрахунки вимагають певних обчислювальних витрат, і вони стали можливі лише з розвитком інформаційних технологій.

Формула Баєса і весь наступний виклад вимагає розуміння ймовірності. На практиці ймовірність настання події є частота настання цієї події, тобто відношення кількості спостережень події до загальної кількості спостережень за великої (теоретично нескінченної) загальної кількості спостережень.

Розглянемо наступний експеримент: ми називаємо будь-яке число з відрізка $[0, 1]$ і стежимо за тим, що це число буде між, наприклад, 0.1 та 0.4. Як неважко здогадатися, ймовірність цієї події дорівнюватиме відношенню довжини відрізка $[0.1, 0.4]$ до загальної довжини відрізка $[0, 1]$ (іншими словами, відношення «кількості» можливих рівноймовірних значень до загальної «кількості» значень), тобто $(0.4 - 0.1) / (1 - 0) = 0.3$, тобто ймовірність попадання у відрізок $[0.1, 0.4]$ дорівнює 30%.

Тепер подивимося на квадрат $[0, 1] \times [0, 1]$.



Назвемо пари чисел (x, y), кожне з яких більше за нуль і менше одиниці. Імовірність того, що x (перше число) буде в межах відрізка $[0.1, 0.4]$ (показаний на першому малюнку як синя область, на даний момент для нас друге число y не важливо), дорівнює відношенню площі синьої області до площі всього квадрата, то є $(0.4 - 0.1) \cdot (1 - 0) \,/\, (1 \cdot 1) = 0.3$, тобто 30%. Таким чином, можна записати, що ймовірність того, що x належить відрізку $[0.1, 0.4]$ дорівнює $p(0.1 <= x <= 0.4) = 0.3$ або для стислості $p(X) = 0.3$.

Аналогічно, ймовірність того, що y знаходиться всередині відрізка $[0.5, 0.7]$ дорівнює відношенню площі зеленої області до площі всього квадрата $p(0.5 <= y = 0.7) = 0.2$ або для стислості $p(Y) = 0.2$.

Тепер подивимося, що можна дізнатися про значення одночасно x та y.

Якщо ми хочемо знати, яка ймовірність того, що одночасно x та y знаходяться у відповідних заданих відрізках, то нам потрібно порахувати відношення темної площі (перетину зеленої та синьої областей) до площі всього квадрата: $p(X, Y) = (0.4 - 0.1\,) \cdot (0.7 - 0.5) \,/\, (1 \cdot 1) = 0.06$.

А тепер припустимо ми хочемо знати, яка ймовірність того, що y знаходиться в інтервалі $[0.5, 0.7]$, якщо x вже знаходиться в інтервалі $[0.1, 0.4]$. Тобто фактично у нас є фільтр і коли ми називаємо пари (x, y), то ми відразу відкидає ті пари, які не задовольняють умову знаходження x у заданому інтервалі, а потім із відфільтрованих пар ми вважаємо ті, для яких y задовольняє нашу умову і вважаємо ймовірність як відношення кількості пар, для яких y лежить у згаданому вище відрізку до загальної кількості відфільтрованих пар (тобто для яких x лежить у відрізку $[0.1, 0.4]$). Ми можемо записати цю можливість як $p(Y|X)$. Очевидно, що ця ймовірність дорівнює відношенню площі темної області (перетин зеленої та синьої областей) до площі синьої області. Площа темної області дорівнює $(0.4 - 0.1) \cdot (0.7 - 0.5) = 0.06$, а площа синій $(0.4 - 0.1) \cdot (1 - 0) = 0.3$, тоді їх відношення дорівнює $0.06/0.3 = 0.2$. Іншими словами, ймовірність знаходження y на відрізку $[0.5, 0.7]$ при тому, що x належить відрізку $[0.1, 0.4]$ дорівнює $p(Y|X) = 0.2$.

Можна помітити, що з урахуванням всього вищесказаного та всіх наведених вище позначень ми можемо написати наступне вираження

$$p(Y|X) = p(X, Y) \,/\, p(X)$$

Коротко відтворимо всю попередню логіку тепер стосовно $p(X, Y)$: ми називаємо пари (x, y) і фільтруємо ті, для яких y лежить між 0.5 і 0.7, тоді ймовірність того, що x знаходиться у відрізку $[0.1, 0.4]$ за умови, що y належить відрізку $[0.5, 0.7]$ дорівнює відношенню площі темної області до зеленої площі:

$$p(X|Y) = p(X, Y) \,/\, p(Y)$$

У двох наведених вище формулах бачимо, що член p(X, Y) однаковий, і ми можемо його виключити:

$$p(X, Y) = p(X|Y) \times p(Y) = p(Y|X) \times p(X)$$

Ми можемо переписати останню рівність як

$$p(X, Y) = \frac{p(Y|X) \times p(X)}{p(Y)}$$

Це і є теорема Баєса.

Цікаво ще помітити, що $p(Y)$ це фактично $p(X,Y)$ при всіх значеннях X. Тобто, якщо ми візьмемо темну область і розтягнемо її так, що вона покриватиме всі значення X, вона точно повторюватиме зелену область , Отже, вона дорівнюватиме $p(Y)$. На мові математики це означатиме таке:

$$P(y) = \int p(X,Y)dX \text{ або } P(y) = \int p(Y|X)p(X)dX$$

Тоді ми можемо переписати формулу Баєса у такому вигляді:

$$p(X|\,Y) = \frac{p(Y|X) \times p(X)}{\int p(Y|X)p(X)dX}$$

Розглянемо наступний приклад. Візьмемо монетку та підкинемо її 3 рази. З однаковою ймовірністю ми можемо отримати такі результати (О – орел, Р – решка): ООО, ОРО, ООР, ОРР, РОО, РОР, РРО, РРР.

Ми можемо порахувати скільки орлів випало кожному разі і скільки у своїй було змін орел-решка, решка-орел:

Результат	**Орлів**	**Змін**	**Ймовірність**
ООО	3	0	1/8
ОРО	2	2	1/8
ООР	2	1	1/8
ОРР	1	1	1/8
РОО	2	1	1/8
РРО	1	1	1/8
РОР	1	2	1/8
РРР	0	0	1/8

Ми можемо розглядати кількість орлів та кількість змін як дві випадкові величини. Тоді таблиця ймовірностей матиме такий вигляд:

Кількість змін	**Орлів**			
	0	**1**	**2**	**3**
0	1/8	0	0	1/8
1	0	2/8	2/8	0
2	0	1/8	1/8	0

Тепер ми можемо побачити формулу Баєса у дії.

Але насамперед проведемо аналогію з квадратом, який ми розглядали раніше.

Можна помітити, що O є сумою третього стовпця («синя область» квадрата) і дорівнює сумі всіх значень осередків у цьому стовпці: $p(1E) = 2/8 + 1/8 = 3/8$

$p(1C)$ є сума третього рядка («зелена область» квадрата) і, аналогічно, дорівнює сумі всіх значень осередків у цьому рядку

$$p(1C) = 2/8 + 2/8 = 4/8$$

Імовірність того, що ми отримали одного орла і одну зміну дорівнює перетину цих областей (тобто значення в клітці перетину третього стовпця та третього рядка) $p(1\mathrm{C}, 1E) = 2/8$

Тоді, слідуючи формулам описаним вище, ми можемо порахувати можливість отримати одну зміну, якщо ми отримали одного орла в трьох кидках:

$$p(1C|1E) = p(1C, 1E) \,/\, p(1E) = (2/8) \,/\, (3/8) = 2/3$$

або можливість отримати одного орла, якщо ми отримали одну зміну:

$$p(1E|1C) = p(1C, 1E) \,/\, p(1C) = (2/8) \,/\, (4/8) = 1/2$$

Якщо ми порахуємо можливість отримати одну зміну за наявності одного орла $p(1E|1C)$ через формулу Баєса, то отримаємо:

$$p(1E|1C) = p(1C|1E) \cdot p(1E)/p(1C) = (2/3) \cdot (3/8)/(4/8) = 1/2$$

Але яке практичне значення має наведений вище приклад?

Справа в тому, що коли ми аналізуємо реальні дані, зазвичай нас цікавить якийсь параметр цих даних (наприклад, середня, дисперсія тощо). Тоді ми можемо провести наступну аналогію з вищенаведеною таблицею ймовірностей: нехай рядки будуть нашими експериментальними даними (позначимо їх $Data$), а

стовпці — можливими значеннями параметра цих даних, що цікавить нас (позначимо його θ). Тоді нас цікавить можливість отримати певне значення параметра на основі наявних даних $p(\theta|Data)$.

Ми можемо застосувати формулу Баєса і записати таке:

$$p(\theta|Data) = \frac{p(Data|\theta) \times p(\theta)}{p(Data)}$$

А згадавши формулу з інтегралом, можна записати таке:

$$p(\theta|Data) = \frac{p(Data|\theta) \times p(\theta)}{\int p(Data|\theta) \times p(\theta) d\theta}$$

Тобто фактично як результат нашого аналізу ми маємо ймовірність як функцію параметра. Тепер ми можемо, наприклад, максимізувати цю функцію і знайти найбільш ймовірне значення параметра, порахувати дисперсію і середнє значення параметра, порахувати межі відрізка, всередині якого параметр, що цікавить нас, лежить з ймовірністю 95% і ін.

Імовірність $p(\theta|Data)$ називають апостеріорною ймовірністю. І для того, щоб порахувати її, нам треба мати $p(Data|\theta)$ – функцію правдоподібності та $p(\theta)$ – апріорну ймовірність.

Функція правдоподібності визначається нашою моделлю. Тобто ми створюємо модель збору даних, яка залежить від параметра, що нас цікавить. Наприклад, ми хочемо інтерполювати дані за допомогою прямої $y = ax + b$ (таким чином ми припускаємо, що всі дані мають лінійну залежність з накладеним на неї гаусовим шумом з відомою дисперсією). Тоді a і b — це наші параметри, і ми хочемо дізнатися про їх найбільш ймовірні значення, а функція правдоподібності — гаус із середнім, заданим рівнянням прямою, і дисперсією.

Апріорна ймовірність включає інформацію, яку ми знаємо до проведення аналізу. Наприклад, ми точно знаємо, що пряма повинна мати позитивний нахил, або, що значення в точці перетину з віссю x має бути позитивним — усе це і не лише ми можемо інкорпорувати у наш аналіз.

Як можна помітити, знаменник дробу є інтегралом (або у

випадку, коли параметри можуть приймати лише певні дискретні значення сумою) чисельника по всіх можливих значеннях параметра. Практично це означає, що знаменник є константою і служить для того, щоб нормалізувати апостеріорну ймовірність (тобто, щоб інтеграл апостеріорної ймовірності дорівнював одиниці).

3.3. Порядок виконання роботи

Найбільш популярним засобом для Баєсового аналізу є мова R з пакетами $JAGS$ та/або $BUGS$. Для рядового користувача розбіжність у пакетах полягає у крос-платформенності $JAGS$ (наявність конфлікту між $Ubuntu$ та $BUGS$), а також у тому, що у $JAGS$ можна створювати свої функції та розподіли (втім, необхідність у цьому у рядового користувача виникає нечасто, якщо взагалі виникає).

Однак в даній лабораторній річ буде йти про альтернативу $R - Python$ з модулем $pymc$.

Завдання буде полягати у знаходженні коефіцієнтів лінійної залежності даних. Подібне завдання оптимізації параметрів досить часто зустрічається в різних галузях знань. Наприкінці ми зможемо порівняти результат Баєсового аналізу та методу найменших квадратів.

Формулювання завдання

Ми маємо дані, отримані в ході гіпотетичного експерименту, які лінійно залежать від деякої величини x. Дані надходять із шумом, дисперсія якого невідома. Необхідно визначити коефіцієнти лінійної залежності.

Вирішення задачі

Спочатку ми імпортуємо модулі, які ми встановили і які нам знадобляться:

```
import numpy
import pymc
```

Потім треба отримати гіпотетичні лінійні дані.

І тому визначаємо, скільки точок хочемо мати (в даному випадку 20), вони рівномірно розподілені на відрізку $[0, 10]$, і задаємо реальні коефіцієнти лінійної залежності. Далі ми накладаємо на дані гаусів шум:

```
#Generate data with noise
number_points     = 20
true_coefficients = [10.4, 5.5]
x                          = numpy.linspace(0, 10,
number_points)
noise                   = numpy.random.normal(size =
number_points)
data                      = true_coefficients[0]*x +
true_coefficients[1] + noise
```

Отже, ми маємо дані, і тепер треба подумати, як проводити аналіз.

По-перше, ми знаємо (або припускаємо), що наш шум гаусів, отже, функція правдоподібності у нас буде гаусова. У неї є два параметри: середнє значення та дисперсія. Так як середнє шуму дорівнює нулю, то середнє для функції правдоподібності задаватиметься значенням моделі (а модель у нас лінійна, тому там два параметри). Тоді як дисперсія нам невідома, отже вона буде ще одним параметром.

В результаті у нас є три параметри та гаусова функція правдоподібності.

Ми нічого не знаємо про значення параметрів, тому спочатку припустимо їх рівномірно розподіленими з довільними кордонами (ці межі можна відсувати скільки завгодно далеко).

При заданні апріорного розподілу, ми повинні вказати мітку, за якою ми дізнаватимемося про апостеріорні значення параметрів (перший аргумент), а також вказати межі розподілу (другий і третій аргументи). Всі перелічені аргументи є обов'язковими (ще є додаткові аргументи, опис яких можна знайти в документації).

```
sigma = pymc.Uniform('sigma', 0., 100.)
a     = pymc.Uniform('a', 0., 20.)
b     = pymc.Uniform('b', 0., 20.)
```

Тепер треба поставити модель. У $pymc$ існує два найбільш часто використовувані класи: детерміністичний і стохастичний. Якщо при заданих вхідних даних можна однозначно визначити значення, яке модель повертає, це детерміністична модель. У нашому випадку при заданих коефіцієнтах лінійної залежності для будь-якої точки ми можемо однозначно визначити результат, відповідно це детерміністична модель:

```
@pymc.deterministic(plot=False)
def linear_fit(a=a, b=b, x=x):
      return a*x + b
```

І нарешті задаємо функцію правдоподібності, в якій середнє це значення моделі, $sigma$ параметр з заданим апріорним розподілом, а $data$ це наші експериментальні дані:

```
y = pymc.Normal('y', mu=linear_fit, tau=1.0/sigma**2,
value=data, observed=True)
```

Принцип роботи $pymc$

З точки зору математики, нам потрібно вирішити таке завдання:

```
p (a, b, sigma | Data) = p (Data | a, b, sigma) * p
(a, b, sigma) / p (Data)
```

Так як a, b і $sigma$ незалежні, ми можемо переписати рівняння в такому вигляді:

```
p(a, b, sigma | Data) = p(Data | a, b, sigma) * p (a)
* p (b) * p (sigma) / p (Data)
```

На папері завдання виглядає дуже просто, але коли його

вирішують чисельно, то виникають труднощі.

$p(Data)$ – це константа.

$p(Data \mid a, b, sigma)$ нам безперечно задана (тобто при відомих a, b і $sigma$ ми можемо однозначно розрахувати ймовірності для наших наявних даних) а ось замість $p(a)$, $p(b)$ і $p(sigma)$ у нас, по суті, є лише генератори псевдовипадкових величин, розподілених за заданим нами законом.

Як із усього цього отримати апостеріорний розподіл? Генерувати (робити вибірку) a, b і $sigma$, та потім рахувати $p(Data \mid a, b, sigma)$. В результаті у нас вийде ланцюжок значень, який є вибіркою з апостеріорного розподілу. Але тут виникає питання, як ми можемо зробити цю вибірку правильно. Якщо наш апостеріорний розподіл має кілька мод (пагорбів), то як можна згенерувати вибірку, що покриває всі моди. Тобто завдання в тому, як ефективно зробити вибірку, яка б покривала весь розподіл за найменшу кількість ітерацій. Для цього є кілька алгоритмів, найбільш використовуваний з яких є $MCMC$ ($Markov\ chain\ Monte\ Carlo$). Ланцюг Маркова - це така послідовність випадкових подій, в якій кожен елемент залежить від попереднього, але не залежить від попереднього. Я не описуватиму сам алгоритм (це може бути темою окремого посту), але тільки зазначу, що $pymc$ реалізує цей алгоритм і як результат дає ланцюг Маркова, що є вибіркою з апостеріорного розподілу. Взагалі кажучи, якщо ми не хочемо, щоб ланцюг був марківським, то нам просто треба його «утончити», тобто. брати, наприклад, кожен другий елемент.

Отже, ми створюємо другий файл, назвемо його $run\_model.py$, в якому генеруватимемо ланцюг Маркова. Файли $model.py$ і $run\_model.py$ повинні бути в одній папці, інакше до файлу $run\_model.py$ потрібно додати код:

```
from sys import path
path.append("шлях/до/папки/з/файлом/model.py/")
```

Спочатку ми імпортуємо деякі модулі, які нам знадобляться:

```
from numpy import polyfit
from matplotlib.pyplot import figure, plot, show, legend
import pymc
import model
```

$polyfit$ реалізує метод найменших квадратів – з ним ми порівняємо результати Баєсового аналізу.

$figure$, $plot$, $show$, $legend$ потрібні для того, щоб побудувати підсумковий графік.

$model$ – це, власне, наша модель.

Потім ми створюємо об'єкт $MCMC$ і запускаємо вибірку:

```
#Define MCMC:
D = pymc.MCMC(model, db = 'pickle')
#Sample MCMC: 10000 iterations, burn-in period is 1000
D.sample(iter = 10000, burn = 1000)
```

$D.sample$ приймає два аргументи – кількість ітерацій та $burn - in$ («період розігріву»). Період розігріву - це кількість перших ітерацій, що скидаються. Справа в тому, що $MCMC$ спочатку залежить від стартової точки, тому нам необхідно відрізати цей період залежності. На цьому наш аналіз закінчено.

Тепер у нас є об'єкт D, в якому знаходиться вибірка, і який має різні методи (функції), що дозволяють розрахувати параметри цієї вибірки (середнє найбільш ймовірне значення, дисперсію тощо).

Для того, щоб порівняти результати, ми робимо аналіз методом найменших квадратів:

```
#compute chi-squared fitting for comparison:
chisq_result = polyfit(model.x, model.data, 1)
```

Тепер друкуємо всі результати:

```
#print the results:
```

```
  print "\n\nResult of chi-square result: a= %f, b= %f" % (chisq_result[0], chisq_result[1])
  print "\nResult of Bayesian analysis: a= %f, b= %f" % (D.a.value, D.b.value)
  print "\nThe real coefficients are:   a= %f, b= %f\n" %(model.true_coefficients[0], model.true_coefficients[1])
```

Будуємо стандартні для pymc графіки:

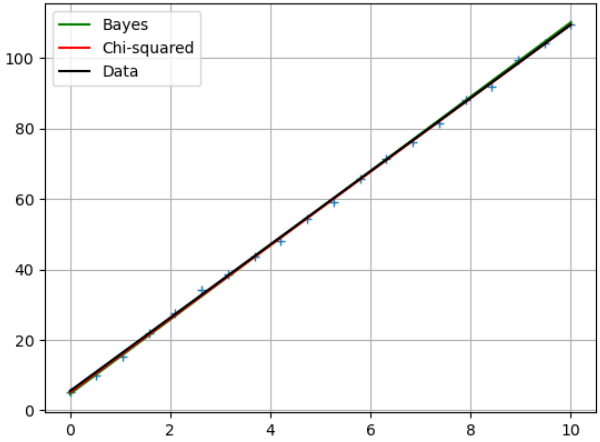
```
  #plot graphs from MCMC:
  pymc.Matplot.plot(D)
```

І, нарешті, будуємо наш підсумковий графік:

```
  #plot noised data, true line and two fitted lines (bayes and chi-squared):
  figure()
  plot(model.x, model.data, marker='+', linestyle='')
  plot(model.x, D.a.value * model.x + D.b.value, color='g', label='Bayes')
  plot(model.x, chisq_result[0] * model.x + chisq_result[1], color='r', label='Chi-squared')
  plot(model.x, model.true_coefficients[0] * model.x + model.true_coefficients[1], color='k', label='Data')
  legend()
  show()
```

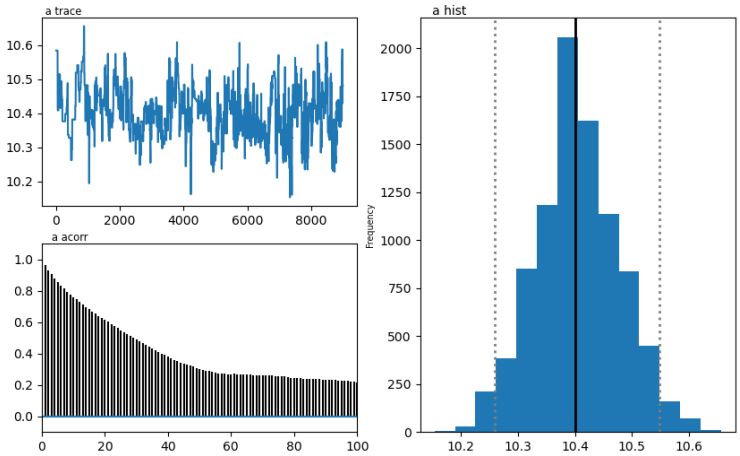
У терміналі ми бачимо таку відповідь:

```
Result of chi-square result: a = 10.369861, b = 5.469129

Result of Bayesian analysis: a = 10.424226, b = 5.008487

The real coefficients are:   a = 10.400000, b = 5.500000
```

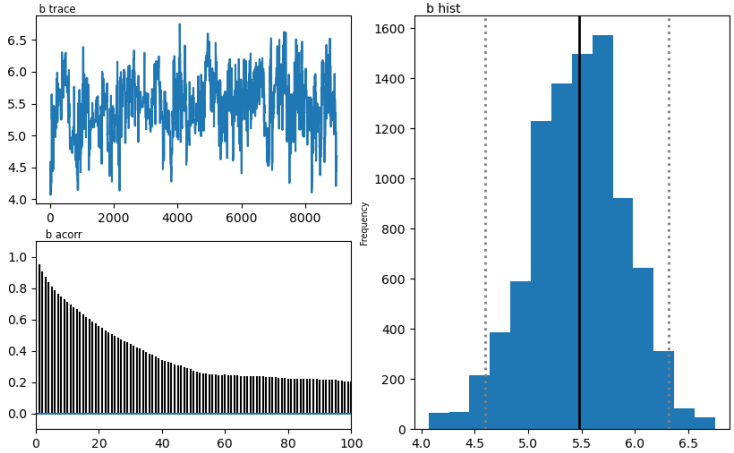


Зауважу, що оскільки ми маємо справу з випадковим процесом, ті значення, які ви побачите у себе, можуть відрізнятися від вищенаведених (крім останнього рядка).

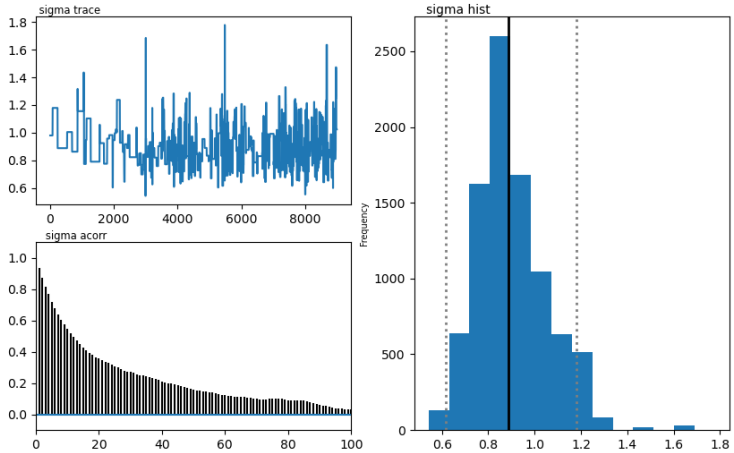
У папці з файлом $run\_model.py$ ми побачимо такі графіки. Для параметра a:



Для параметра b:



Для параметра $sigma$:



Справа бачимо гістограму апостеріорного розподілу, а дві картинки зліва відносяться до ланцюга Маркова.

На них я зараз не звертатиму уваги. Скажу лише, що нижній – це графік автокореляції. Він дає уявлення про збіжність $MCMC$.

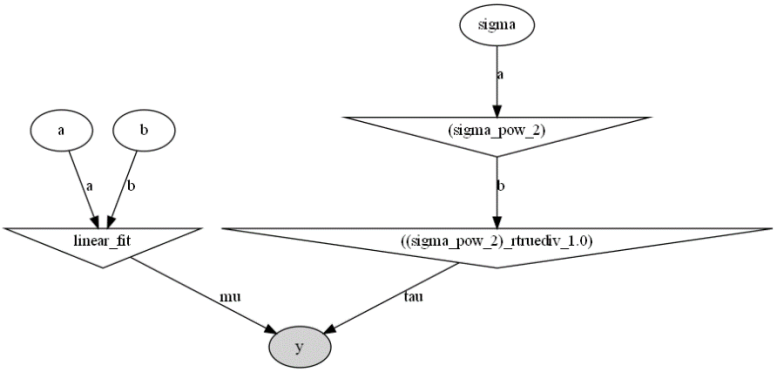
А верхній графік – слід вибірки тобто, як відбувалася вибірка

з часом. Середнє цього сліду є середня вибірка (порівняйте вертикальну вісь на цьому графіку з горизонтальною віссю на гістограмі праворуч).

Якщо поставити модуль $pydot$ і в файл $run\_model.py$ включити наступний рядок:

```
pymc.graph.dag(D).write_png('dag.png')
```

Він створить у папці з файлом $run\_model.py$ наступний малюнок:



Це прямий ациклічний граф, який представляє нашу модель. Білі еліпси показують стохастичні вузли (це a, b і $sigma$), трикутники детерміністичні вузли, а затемнений еліпс включає наші псевдоекспериментальні дані.

Тобто ми бачимо, що значення a і b надходять у нашу модель ($linear\_fit$), яка сама по собі є детерміністським вузлом, а потім надходять у функцію правдоподібності y. $Sigma$ спочатку задається стохастичним вузлом, але оскільки параметром функції правдоподібності є не $sigma$, а $tau = 1/sigma^2$, то стохастичне значення $sigma$ спочатку зводиться в квадрат (верхній трикутник праворуч), і потім обчислюємо tau, яка надходить у функцію правдоподібності, як і наші згенеровані дані. Цей граф корисний як для пояснення моделі, так і для самостійної перевірки логіки моделі.

3.4. Контрольні запитання

1. На якій теорії побудовані принципи Баєсових методів? Запишіть теорему Баєса.

2. Що потрібно аби порахувати імовірність $p(\theta|Data)$? Вкажіть від чого залежить функція правдоподібності?

3. Який принцип роботи бібліотеки $pymc$?

4. Які два класи $pymc$ є найбільш використовувані? Опишіть їх.

5. Опишіть елементи і їх значення ациклічного графа.

Лабораторна робота №4. Обробка природної мови (NLP) у Python з кодом. Кластеризація текстів. Тематичне моделювання)

4.1. Мета роботи

Вивчити способи реалізації кластеризації текстів та завдання тематичного моделювання. Використання методів, що використовують дані методи обробки мови. Візуалізація результатів інтерактивним інструментом *pyLDAvis*.

4.2. Теоретичні відомості

1. Кластеризація текстів.

На минулій лекції було роз'яснено проблеми класифікації, яка є формою контрольованого навчання. Однак існує і кластеризація, яка є формою навчання без учителя.

Кластеризація тексту — це завдання угруповання набору текстів без міток таким чином, щоб тексти в одній групі (кластери) були більш схожі один на одного, ніж на тексти в інших кластерах.

Багато алгоритмів кластеризації доступні в *Scikit-Learn* та інших бібліотеках, але, можливо, найпростішим для розуміння є алгоритм, відомий як кластеризація *k*-середніх, який реалізований у *sklearn.cluster.KMeans*.

k-середніх алгоритм пошуку для заздалегідь певної кількості кластерів в межах немічених багатовимірного масиву даних. Використовується проста концепція того, як виглядає оптимальна кластеризація:

«Центр кластеру» — це середнє арифметичне всіх точок, що належать кластеру.

Кожна точка знаходиться ближче до свого центру кластера, ніж до інших центрів кластера.

Ці два припущення є основою моделі *k*-середніх.

2. Тематичне моделювання.

Завдання тематичного моделювання: захоплювати семантичну інформацію за межами окремих слів; виявляти приховані теми чи теми в документах;

відповідно анотувати документи; використовувати анотації для управління, узагальнення, пошуку та рекомендування вмісту.

У машинному навчанні та обробки природної мови, тематична модель являє собою тип статистичної моделі для виявлення абстрактних «тем», що зустрічаються в колекції документів. Тематичне моделювання — це часто використовуваний інструмент інтелектуального аналізу тексту для виявлення прихованих семантичних структур в тілі тексту та дозволяє нам ефективно аналізувати великі обсяги текстів, об'єднуючи документи в кластери.

Великий обсяг текстових даних практично не помічений, що означає, що ми не зможемо застосувати наші попередні підходи до контрольованого навчання, тому що ці моделі машинного навчання насправді будуть залежати від історичних даних. У вас не буде зручною мітки, прикріпленою до текстового набору даних, наприклад позитивною або негативною. Замість цього у вас може бути безліч ярликів, таких як різні категорії, які публікуються в газетній статті. Отже нам належить спробувати виявити ці ярлики за допомогою тематичного моделювання.

Еволюція моделей від мішків слів до прихованих тем:

Модель	Рік	Опис
Векторна модель	1975	Представлення колеції документів векторами одного спільного для всієї колекції векторного простору
Латентно-семантичний аналіз	1988	Дозволяє аналізувати взаємозв'язок між набором ддокументів і термінами , які в них зустрічаються, шляхом створення набору понять.
Ймовірний латентно-семантичний аналіз	1999	Заснований на змішаному розкладанні, отриманому з моделі прихованих класів
Прихований розподіл Діріхле	2003	Додає генеративний процес для документів: трирівнева ієрархія, байєсівська модель

Прихований розподіл Діріхле (*LDA – Latent Dirichlet*

Allocation)— найпопулярніший метод моделювання тем, тож будемо використовувати його. *LDA* — це метод матричної факторизації. У векторному просторі будь-який корпус (збори документів) може бути представлений як матриця документ-термін.

Тут слід мати на увазі дуже важливу ідею: насправді дуже складно оцінити неконтрольоване навчання, ефективність моделі навчання, тому що ми насправді не знали правильну тему або правильну відповідь. Все, що ми знаємо, це те, що документи, згруповані разом, поділяють якісь схожі тематичні ідеї. Користувач повинен визначити, що насправді являють ці теми.

4.3. Порядок виконання роботи

1. Кластеризація текстів.

Нам потрібно отримати необроблений текст заголовків статей.

```
df = pd.read_csv('https://raw.githubusercontent.com/olegdubetcky/Text-Classification-with-ML-Project/main/news.csv', encoding='utf8')
```

Виконаємо попередню обробку тексту для поліпшення роботи алгоритму. Тож, виконаємо очищення від стоп-слів та приведемо слова в нормальну форму. Це знижує рівень шуму, щоб алгоритм не надавав ваги, скажімо, використовувати минулий час замість справжнього або щоб іменники у множині не зважали окремою частиною словника замість іменника в однині.

```
nltk.download('stopwords')
nltk.download('punkt')
url = 'https://raw.githubusercontent.com/olegdubetcky/Ukrainian-Stopwords/main/ukrainian'
r = requests.get(url)
```

```
with
open(nltk.data.path[0]+'/corpora/stopwords/ukrainian',
'wb') as f:
    f.write(r.content)

stopwords = stopwords.words("ukrainian")
```

Створимо два словникових списка: один з основами і один тільки з токенізацією.

```
totalvocab_normalized = []
totalvocab_tokenized = []
stop_words = frozenset(stopwords +
list(string.punctuation))
for index, row in df.iterrows():
    sentences =
nltk.sent_tokenize(row['title'].lower())
    for sentence in sentences:
        words = nltk.word_tokenize(sentence)
        without_stop_words = [word for word in words if
not word in stop_words]

        normal_words = []
        for token in without_stop_words:
            p = morph.parse(token)[0]
            normal_words.append(p.normal_form)
        n_title = ' '.join(normal_words)
        totalvocab_tokenized.extend(without_stop_words)
        totalvocab_normalized.extend(normal_words)
        df.loc[index, 'normalized'] = n_title
```

Зі словникових списків ми можемо зробити таблицю пошуку основ для повного слова. Однак, оскільки основи можуть ставитися до декількох слів, ця таблиця пошуку поверне тільки перше слово.

Тепер можемо реалізувати алгоритм *TF-IDF Vectorizer*.

```
vectorizer = TfidfVectorizer()
vec_matrix = vectorizer.fit_transform(df['normalized'])
```

В результаті виходить матриця розміром (3293, 6013) близько 6 000 слів.

Тепер ми можемо приступити до кластеризації.

Складно відповісти, скільки кластерів призначити даним. За допомогою *KMeans* дані занадто сильно згруповані разом для ієрархічної кластеризації або будь-якого алгоритму, який самостійно знаходить кластери. Тож оберемо метод оцінки кількості кластерів, які з'являться у наборі даних.

```
from sklearn.cluster import KMeans
import math
n_clusters = int(math.sqrt(df.shape[0] / 2) * 1.5)

km = KMeans(n_clusters=n_clusters)
km.fit(vec_matrix)
```

Тепер можна зв'язати кожну статтю з відповідним кластером.

```
clusters = km.labels_.tolist()

df['cluster'] = clusters
df.head()
```

Нарешті, можемо роздрукувати кластери і назви пов'язаних з ними статей.

```
# сортувати центри кластера за близькістю до центроїда
order_centroids = km.cluster_centers_.argsort()[:, ::-
1]

for i in range(n_clusters):
    print("Кластер %d слова:" % i, end='')

    for ind in order_centroids[i, :6]:
        print(' %s' % terms[ind], end=',')

    print("Кластер %d назви:" % i, end='')
```

```
    for title in df[df['cluster'] ==
i]['title'].values.tolist():
        print(' - %s' % title)
```

Як бачите, деякі з них краще за інших, але, грунтуючись виключно на назвах, K-Means, схоже, розділили статті на відносно послідовні кластери.

```
Топ слова у кластері:

Кластер 0 слова: львов, демонструвати, мер, виставка, додатковий, ковід,

Кластер 0 назви:
 - У Львові ще одна лікарня прийматиме хворих на ковід та опрацьовано
можливість страхувати медиків у разі інфікування
 - За сьогоднішній день у Львові провакцинували 640 людей
 - Влада Львова проситиме про обмеження інспекційних перевірок
підприємств
 - У Львові розгортають додаткові місця у всіх лікарнях, де приймають
хворих на Covid-19
 - У Львові до 30 листопада суботу й неділю оголошено робочими днями, -
міськрада


Кластер 1 слова: сша, байден, трамп, інавгурація, імпічмент, вибороти,

Кластер 1 назви:
 - США допоможуть Україні у підписанні контрактів на отримання вакцини
від корнавірусу
 - Влада зробить усе можливе, щоб притягнути до відповідальності винних у
втручанні у вибори в США, - Єрмак
 - Трамп ветував військовий бюджет США на 2021 рік
 - "46-й президент США": спецмарафон на "5 наналі" до інавгурації Джо
Байдена
 - Адміністрація Байдена добиватиметься екстрадиції Ассанжа
Кластер 2 слова: виявити, доба, випадок, новий, україна, 19,

Кластер 2 назви:
 - У Києві за добу виявили 422 випадки коронавірусу
 - В Україні за добу виявлено 5 062 нові випадки коронавірусу
 - За добу в Україні виявлено рекордні 7053 нових випадків COVID-19
 - За добу в Україні виявлено 5181 новий випадок COVID-19
 - За добу виявлені рекордні 7 517 нових випадків COVID-19, одужали 2 680
людей
```

Код до 1-ої частини:

```
import pandas as pd
```

```
df = pd.read_csv('https://raw.githubusercontent.com/olegdubetcky/Text-Classification-with-ML-Project/main/news.csv', encoding='utf8')

import requests
import nltk
nltk.download('stopwords')
nltk.download('punkt')
import string
from nltk.corpus import stopwords

url = 'https://raw.githubusercontent.com/olegdubetcky/Ukrainian-Stopwords/main/ukrainian'
r = requests.get(url)
with open(nltk.data.path[0]+'/corpora/stopwords/ukrainian', 'wb') as f:
    f.write(r.content)
# Retrieve HTTP meta-data
print(r.status_code)
print(r.headers['content-type'])
print(r.encoding)

import string
from nltk.corpus import stopwords
stopwords = stopwords.words("ukrainian")

!pip install git+https://github.com/kmike/pymorphy2.git
!pip install -U pymorphy2-dicts-uk

import pymorphy2
morph = pymorphy2.MorphAnalyzer(lang='uk')

totalvocab_normalized = []
totalvocab_tokenized = []
stop_words = frozenset(stopwords+list(string.punctuation))
for index, row in df.iterrows():
  sentences = nltk.sent_tokenize(row['title'].lower())
  for sentence in sentences:
    words = nltk.word_tokenize(sentence)
      without_stop_words = [word for word in words if not word in stop_words]

    normal_words=[]
    for token in without_stop_words:
      p = morph.parse(token)[0]
      normal_words.append(p.normal_form)
    n_title = ' '.join(normal_words)
    totalvocab_tokenized.extend(without_stop_words)
    totalvocab_normalized.extend(normal_words)
    df.loc[index, 'normalized'] =n_title

vocab_frame = pd.DataFrame({'слова': totalvocab_tokenized}, index = totalvocab_normalized)
vocab_frame.head()
```

```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer()
vec_matrix = vectorizer.fit_transform(df['normalized'])

print(vec_matrix.shape)
terms = vectorizer.get_feature_names()

from sklearn.cluster import KMeans
import math
n_clusters = int(math.sqrt(df.shape[0] / 2) * 1.5)

km = KMeans(n_clusters=n_clusters)
km.fit(vec_matrix)

clusters = km.labels_.tolist()

df['cluster'] = clusters
df.head()

print("Топ слова у кластері:")
print()

#сортувати центри кластера за близькістю до центроїда
order_centroids = km.cluster_centers_.argsort()[:, ::-1]

for i in range(n_clusters):
    print("Кластер %d слова:" % i, end='')

    for ind in order_centroids[i, :6]:
        print(' %s' % terms[ind], end=',')

    print()
    print()

    print("Кластер %d назви:" % i, end='')
    print()
    for title in df[df['cluster'] == i]['title'].values.tolist():
        print(' - %s' % title)
    print()
    print()

print()
print()
```

2. Тематичне моделювання.

Встановимо параметри для векторізатора: *max_df* = .2 означає ігнорувати терміни, що зустрічаються більш ніж у 20% документів; *max_df* = 3 означає ігнорувати терміни, що зустрічаються більш ніж у 3 документах; *max_features* — це

розмір випадкових підмножин функцій, які слід враховувати при поділі вузла. Таким чином, це по суті позбавлення від термінів, які дійсно використовуються в багатьох документах. Так що зазвичай непогано викинути кілька дійсно загальних слів, а також викидати випадкові слова, можливо, навіть помилки або орфографічні помилки.

```
vectorizer = TfidfVectorizer(max_df=.2, min_df=3,
max_features=2000)
doc_train_matrix =
vectorizer.fit_transform(train_docs['normalized'])
words = vectorizer.get_feature_names()
doc_train_matrix

topic_labels = ['Тема {}'.format(i) for i in range(1,
9)]

lda = LatentDirichletAllocation(n_components=8,
n_jobs=-1, max_iter=500,
learning_method='batch', evaluate_every=5,
                                verbose=1,
random_state=42)

topics_count = lda.components_
```

Отже, є два основних припущення, які ми збираємося зробити, щоб насправді застосувати *LDA* для тематичного моделювання. По-перше, в документах схожої тематики використовуються схожі групи слів. І це досить розумне припущення, тому що в основному йдеться, що якщо у вас є різні документи охоплюють схожу тему, наприклад, набір документів по темі бізнесу або економіки, які в кінцевому підсумку вони повинні використовувати схожі слова, такі як гроші, ціна, ринкові акції … Ще одне припущення, яке ми збираємося зробити, полягає в тому, що приховані теми можна знайти, скориставшись пошуком роботи по групах слів, які часто зустрічаються разом в документах по всьому корпусу.І ми дійсно можемо думати про ці дві припущеннях математично.

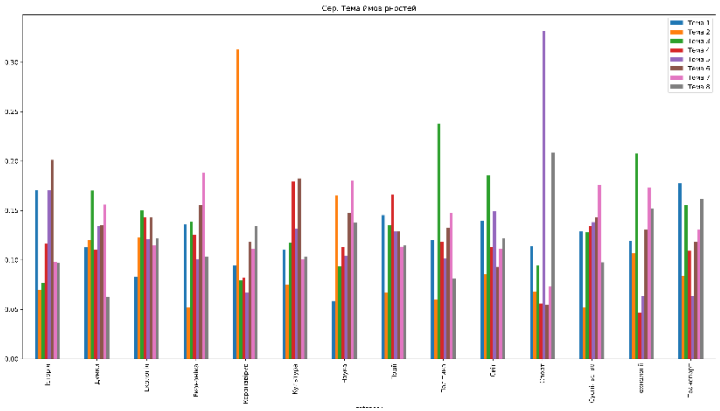
Ми можемо змоделювати ці припущення наступним чином. Ми можемо сказати, що документи — це імовірнісні розподілу за деякими прихованим темам, а потім самі теми є імовірнісні розподілу слів. Ми можемо уявити, що будь-який конкретний документ матиме розподіл ймовірностей по заданій кількості прихованих тим, тому припустимо, що ми вирішили, що існує вісім прихованих тим в різних документах, тоді будь-який конкретний документ буде мати можливість приналежності до кожної теми.

Можемо поглянути на найпопулярніші слова у темах:

```
n_words = 12
top_words = {}
for topic, words_ in topics.items():
    top_words[topic] =
words_.nlargest(n_words).index.tolist()
```

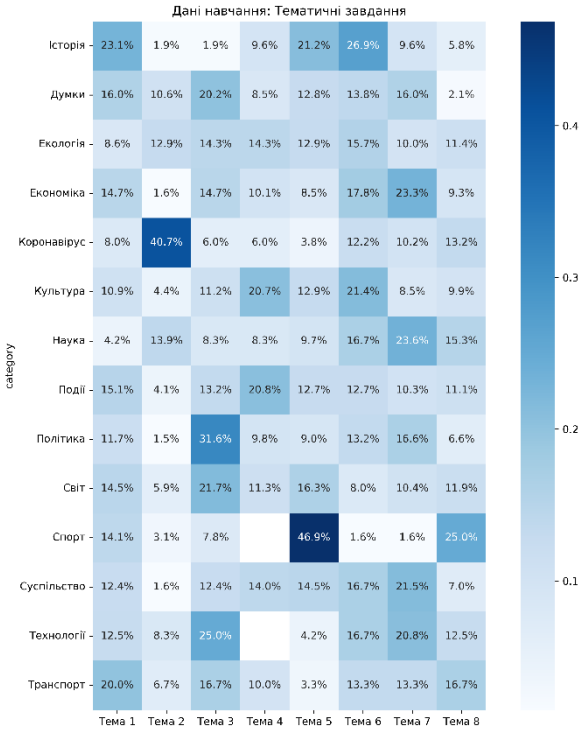
	Тема 1	Тема 2	Тема 3	...	Тема 6	Тема 7	Тема 8
0	день	covid	сша	...	український	створити	карантинний
1	іран	19	та	...	львівський	кількість	червоний
2	північний	випадок	байден	...	зеленський	планувати	зона
3	вихідний	доба	проти	...	міський	світ	область
4	карантин	новий	зеленський	...	представити	мільйон	зон
5	літак	коронавірус	санкція	...	посол	пандемія	обмеження
6	локдаун	виявити	закликати	...	знайти	почати	карантин
7	мау	зафіксувати	рад	...	вакцина	туреччина	європа
8	мзс	вакцина	фото	...	голова	15	відкрити
9	працювати	кий	який	...	та	зеленський	та
10	музей	вакцинація	трамп	...	допомога	коронавірус	єс
11	проєкт	хворий	закон	...	національний	скасувати	квітень

Тепер можна моглянути як перетинаються теми і категорії:

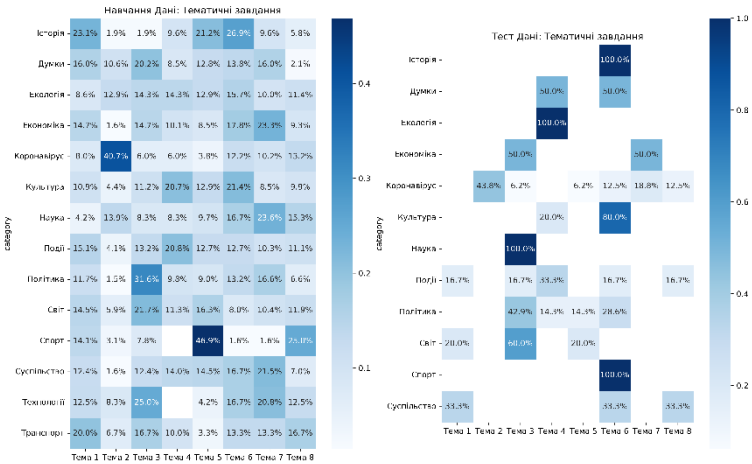


Ясно що категорії «Коронавірус», «Наука», «Політика», «Спорт» та «Транспорт» чітко відрізняються.

Більшість понять тем має відносно чіткий зв’язок з якоюсь іншою темою.



Отже, давайте переглянемо оцінку з тестовим набором. При використанні інформації про гіпотезу у тестовому наборі, а також вибір з цієї теми забезпечує найвищу якість. Таким чином, ми можемо призначити критику цієї теми, а потім подивитися, який вихідний текст.



Ось ознаки того, що ми бачимо зовсім іншу тему. Давайте подивимося на них в дії, вибравши кілька випадків.

```
train_eval =
pd.DataFrame(data=lda.transform(doc_train_matrix),
                              columns=topic_labels,
                              index=train_docs.category)
test_eval =
pd.DataFrame(data=lda.transform(doc_test_matrix),
                            columns=topic_labels,
                            index=test_docs.category)
# Перегляньте неправильно класифіковані статті
test_assignments =
test_eval.groupby(level='category').idxmax(
    axis=1).reset_index(-1,
drop=True).to_frame('predicted').reset_index()
test_assignments['title'] = test_docs.title.values
```

```
    category predicted                                          title
0    Історія    Тема 6  Байден прокоментував голосування у справі про ...
1      Думки    Тема 6  Україна безкоштовно отримає 16 мільйонів доз в...
2      Думки    Тема 4  Україна розпочне щеплення вакцииною Pfizer, на...
3   Екологія    Тема 4  На Львівщині впроваджують проєкт з аеромедично...
4  Економіка    Тема 7  Колишнього голову правління &quot;ПриватБанку&...
```

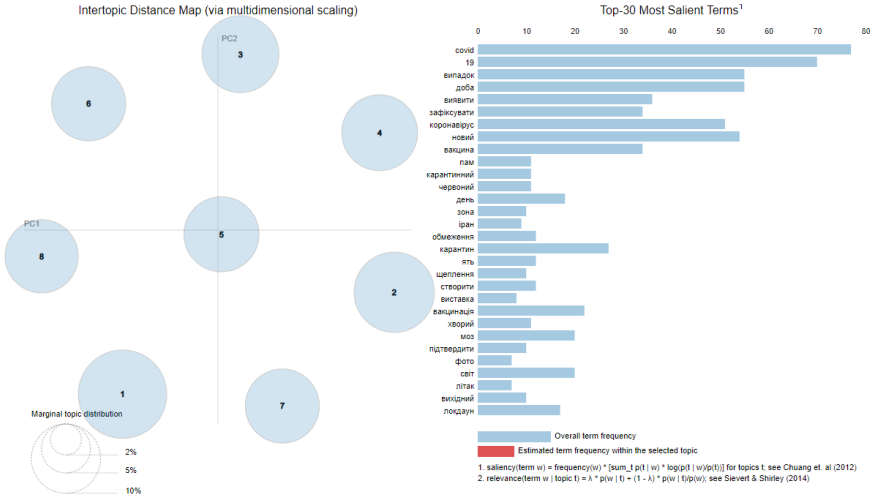
```
misclassified =
test_assignments[(test_assignments.category == 'Світ')
& (
        test_assignments.predicted == 'Тема 3')]
misclassified.title
```

```
41    В Асканії-Новій може загинути увесь "молодняк"...
42    Переклад книги В’ятровича про польсько-українс...
43    Окупанти передадуть полонених українців через ...
Name: title, dtype: object
['В Асканії-Новій може загинути увесь "молодняк" сірих журавлів', 'Переклад книги
В’ятровича про польсько-українську війну видали в Канаді', 'Окупанти передадуть полонених
українців через Мінськ, - Денісова']
```

Оцінка та візуалізація результатів

pyLDAvis — інтерактивний інструмент, який дозволяє досліджувати значення об’єктів і взаємозв’язків. Так ви зможете візуалізувати великі робочі відносини, а також зрозуміти, як теми пов’язані з кожним з них.

```
lda_viz = pyLDAvis.sklearn.prepare(lda,
doc_train_matrix, vectorizer, mds='tsne')
pyLDAvis.display(lda_viz)
```



Код до 2-ої частини:

```
import pandas as pd
df = pd.read_csv('https://raw.githubusercontent.com/olegdubetcky/Text-Classification-with-ML-Project/main/news.csv', encoding='utf8')

import requests
import nltk
nltk.download('stopwords')
nltk.download('punkt')
import string
from nltk.corpus import stopwords

url = 'https://raw.githubusercontent.com/olegdubetcky/Ukrainian-Stopwords/main/ukrainian'
r = requests.get(url)
with open(nltk.data.path[0]+'/corpora/stopwords/ukrainian', 'wb') as f:
    f.write(r.content)
# Retrieve HTTP meta-data
print(r.status_code)
print(r.headers['content-type'])
print(r.encoding)

import string
from nltk.corpus import stopwords
stopwords = stopwords.words("ukrainian")

!pip install git+https://github.com/kmike/pymorphy2.git
!pip install -U pymorphy2-dicts-uk

import pymorphy2
morph = pymorphy2.MorphAnalyzer(lang='uk')
```

```
    stop_words = frozenset(stopwords+list(string.punctuation)+[u'quot'])
    for index, row in df.iterrows():
      sentences = nltk.sent_tokenize(row['title'].lower())
      for sentence in sentences:
        words = nltk.word_tokenize(sentence)
          without_stop_words = [word for word in words if not word in
stop_words]

        normal_words=[]
        for token in without_stop_words:
          p = morph.parse(token)[0]
          normal_words.append(p.normal_form)
        n_title = ' '.join(normal_words)
        df.loc[index, 'normalized'] =n_title

    #розділити дані на зразки для навчання та тестування
    from sklearn.model_selection import train_test_split
    train_docs, test_docs = train_test_split(df, stratify=df.category,
test_size=50, random_state=42)

    train_docs.shape, test_docs.shape

    import pandas as pd
    pd.Series(test_docs.category).value_counts()

    from sklearn.feature_extraction.text import TfidfVectorizer
    vectorizer = TfidfVectorizer(max_df=.2, min_df=3, max_features=2000)
    doc_train_matrix = vectorizer.fit_transform(train_docs['normalized'])
    words = vectorizer.get_feature_names()
    doc_train_matrix

    doc_test_matrix = vectorizer.transform(test_docs['normalized'])
    doc_test_matrix

    topic_labels = ['Тема {}'.format(i) for i in range(1, 9)]

    from sklearn.decomposition import LatentDirichletAllocation
    lda = LatentDirichletAllocation(n_components=8, n_jobs=-1, max_iter=500,
                                                    learning_method='batch',
evaluate_every=5,
                                verbose=1, random_state=42)
    lda.fit(doc_train_matrix)

    import joblib
    joblib.dump(lda, 'lda_500_iter.pkl')
    lda = joblib.load('lda_500_iter.pkl')

    #Досліджуйте теми та розподіл слів
    topics_count = lda.components_
    print(topics_count.shape)
    topics_count[:8]
```

```
topics_prob = topics_count / topics_count.sum(axis=1).reshape(-1, 1)
topics = pd.DataFrame(topics_prob.T,
                      index=words,
                      columns=topic_labels)
topics.head()

from wordcloud import WordCloud
import matplotlib.pyplot as plt
w = WordCloud(background_color='white',max_font_size = 50)
fig, axes = plt.subplots(ncols=2, nrows=4, figsize=(12, 12))
axes = axes.flatten()
for t, (topic, freq) in enumerate(topics.items()):
    w.generate_from_frequencies(freq.to_dict())
    axes[t].imshow(w, interpolation='bilinear')
    axes[t].set_title(topic, fontsize=12)
    axes[t].axis('off')
axes[5].set_visible(False)
plt.tight_layout()

# всі слова мають позитивну ймовірність для всіх тем
topics[topics.gt(0).all(1)].shape[0] == topics.shape[0]

n_words = 12
top_words = {}
for topic, words_ in topics.items():
    top_words[topic] = words_.nlargest(n_words).index.tolist()
pd.DataFrame(top_words)

import seaborn as sns
sns.heatmap(topics, cmap='Blues')

train_preds = lda.transform(doc_train_matrix)
train_preds.shape

train_eval = pd.DataFrame(train_preds, columns=topic_labels,
index=train_docs.category)
train_eval.head()

train_eval.groupby(level='category').mean().plot.bar(title='Сер. Тема
ймовірностей', figsize=(20,10));

df = train_eval.groupby(level='category').idxmax(
    axis=1).reset_index(-1, drop=True)
plt.figure(figsize=(12, 12))
sns.heatmap(df.groupby(level='category').value_counts(normalize=True)
                    .unstack(-1), annot=True, fmt='.1%', cmap='Blues',
square=True)
plt.title('Дані навчання: Тематичні завдання')

train_eval = pd.DataFrame(data=lda.transform(doc_train_matrix),
                          columns=topic_labels,
                          index=train_docs.category)
test_eval = pd.DataFrame(data=lda.transform(doc_test_matrix),
```

```
                                columns=topic_labels,
                                index=test_docs.category)

    import seaborn as sns

    fig, axes = plt.subplots(ncols=2, figsize=(16,10))
    source = ['Навчання', 'Тест']
    for i, df in enumerate([train_eval, test_eval]):
        df = df.groupby(level='category').idxmax(
        axis=1).reset_index(-1, drop=True)

sns.heatmap(df.groupby(level='category').value_counts(normalize=True)
                        .unstack(-1),  annot=True,  fmt='.1%',  cmap='Blues',
square=True, ax=axes[i])
        axes[i].set_title('{} Дані: Тематичні завдання'.format(source[i]));

    #Перегляньте неправильно класифіковані статті
    test_assignments = test_eval.groupby(level='category').idxmax(
                                                        axis=1).reset_index(-1,
drop=True).to_frame('predicted').reset_index()
    test_assignments['title'] = test_docs.title.values
    test_assignments.head()

    misclassified = test_assignments[(test_assignments.category == 'Світ') &
(
        test_assignments.predicted == 'Тема 3')]
    misclassified.title

    misclassified.title.tolist()

    !pip install pyLDAvis
    import pyLDAvis.sklearn

    lda_viz  =  pyLDAvis.sklearn.prepare(lda,  doc_train_matrix,  vectorizer,
mds='tsne')
    pyLDAvis.display(lda_viz)
```

4.4. Контрольні запитання

1. Дайте визначення термінів кластеризації тексту та тематичного моделювання.

2. На якому алгоритмі була реалізована кластеризація тексту в даній лекції? Опишіть його основу моделі.

3. Що таке прихований розподіл Діріхле? Опишіть принцип його роботи.

4. Який інтерактивний інструмент дозволяє наглядно продемонструвати результати для подальшої оцінки?

5. Назвіть основні припущення, що були допущені для застосування *LDA* в тематичному моделюванні.

Лабораторна робота №5. Обробка природної мови (NLP) у Python з кодом. Розпізнавання іменованих сутностей. Вкладання слів та семантична подібність)

5.1. Мета роботи

Розробити задачу для виконання завдання розпізнавання іменованих сутностей за допомогою *WebAnno*. Вивчити методи вкладання слів та семантичної подібності у *Python*.

5.2. Теоретичні відомості

1. Розпізнавання іменованих сутностей

Це завдання видобування інформації, яка спрямована на пошук і класифікацію іменованих сутностей, згаданих в неструктурованому тексті, по заздалегідь певних категорій, таким як імена людей, організації, місця розташування, медичні коди, вираження часу, кількості, грошові значення, відсотки та інше.

Для цього ви можете використовувати готову попередньо навчену модель *NER* за допомогою бібліотеки з відкритим вихідним кодом, таку як Spacy або Stanford CoreNLP.

Тепер, якщо ви гадаєте, що попередньо навчені моделі *NER* не дають результату відповідно до ваших очікувань, які ви шукаєте (приклад: тварина, об’єкт), недоступні в попередньо навченої моделі *NER*, тоді ви можете навчити свою власну кастомну модель.

Для навчання кастомної моделі *NER* у вас має бути величезна кількість анотованих даних. Для цього ви повинні використовувати який-небудь інструмент анотації, наприклад:

- Brat rapid annotation tool
- GATE

- WebAnno

WebAnno є прийнятним для мене, тому що постачається як файл *jar*, що означає, що вам не потрібно його встановлювати. Може використовуватися для складного проекту — кілька користувачів можуть одночасно працювати в одному проекті. Та найголовніше — простота використання.

2. Вкладання слів та семантична подібність

При обробці природної мови вкладання слів використовується для подання слів для аналізу тексту у формі вектора, який виконує кодування значення слова таким чином, щоб слова, які знаходяться ближче у цьому векторному просторі, мали аналогічні у сенсі.

Вкладання слів — це вивчене уявлення тексту, в якому слова, що мають однакове значення, мають аналогічне подання. Іншими словами, він представляє слова в системі координат, де пов'язані слова, засновані на сукупності відносин, розташовані ближче одне до одного. Саме такий підхід до подання слів і документів можна вважати одним з ключових досягнень глибокого навчання в вирішенні складних проблем обробки природної мови.

Мета полягає в тому, щоб закодувати (нормовані) слова у вектор, який існує на певній позиції в «просторі слів». По суті, ми представляємо словниковий запас у векторному просторі. В математичних принципах косинусної та евклідової відстані починається магія.

Word2Vec — один із найпопулярніших методів вивчення вкладання слів з використанням неглибокої нейронної мережі. Його розробив Томаш Міколов у 2013 році в *Google*.

Щоб використовувати вбудовування слів, у вас є два основних варіанти:

Використовуйте попередньо навчені моделі, які ви можете завантажити онлайн (найпростіший варіант)

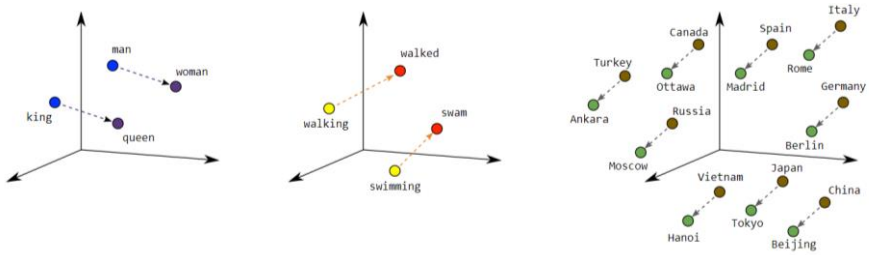
Навчайте призначені для користувача моделі, використовуючи свої власні дані і алгоритм Word2Vec (або інший).

Тут згадуються дві бібліотеки обробки природної мови (NLP) Python:

Gensim — це бібліотека моделювання тем для Python, яка забезпечує доступ до Word2Vec і інших алгоритмів вбудовування слів для навчання, а також дозволяє завантажувати попередньо навчені вбудовування слів, які ви можете завантажити з Інтернету. Наприклад проекти групи lang-uk Word embeddings (*Word2Vec*, *GloVe*, *LexVec*).

Семантична подібність визначається шляхом порівняння векторів слів для «вбудовування слів», багатовимірних представлень значення слова.

Положення (відстань та напрямок) у векторному просторі може кодувати семантику в хорошому вбудовуванні. Наприклад, такі візуалізації реальних вкладень показують геометричні відносини, які фіксують семантичні відносини, такі як відносини між країною та її столицею:



Male-Female Verb Tense Country-Capital

Такий осмислений простір дає вашій системі машинного навчання можливість виявляти закономірності, які можуть

допомогти у вирішенні завдання навчання.

5.3. Порядок виконання роботи

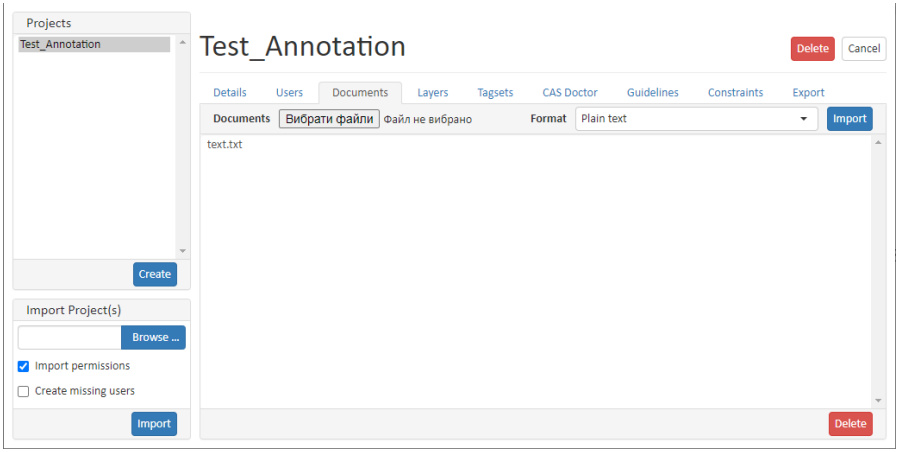
1. Розпізнавання іменованих сутностей

Завантажте webanno по посиланню нижче:
WebAnno 3.6.7
Щоб запустити це веб-додаток, в консолі використовуйте наступну команду:

```
java -jar webanno-standalone-3.6.7.jar
```

WebAnno — це веб-додаток, в браузері відкрийте нову вкладку http://localhost:8080/login.html

Створіть новий проект, на сторінці налаштувань проектів, у вкладці «*Projects*» натисніть «*Create*». Напишіть яку-небудь назву проекту. (Приклад: «*Test_Annotation*») та оберіть «*Project type*» як *annotation*.



Після визначення деталей проекту з’являться кілька вкладок, таких як Користувачі, Документи, Шари, Набори тегів та ін.

Звідти виберіть вкладку Документи і зробіть наступне: у списку виберіть формат «*Plain text*», завантажте текстовий файл текстового документа, для якого ми будемо готувати навчальні дані та натисніть «*Import*». Зразок тексту, який я використовував у вхідному текстовому файлі для підготовки даних навчання, наведено нижче:

Командувач Сил спеціальних операцій США у Європі Девід Тейбор відвідав 73-й морський центр спеціального призначення імені кошового отамана Антіна Головатого.

Про візит американського армійця повідомила пресслужба військової частини в понеділок, 3 травня.

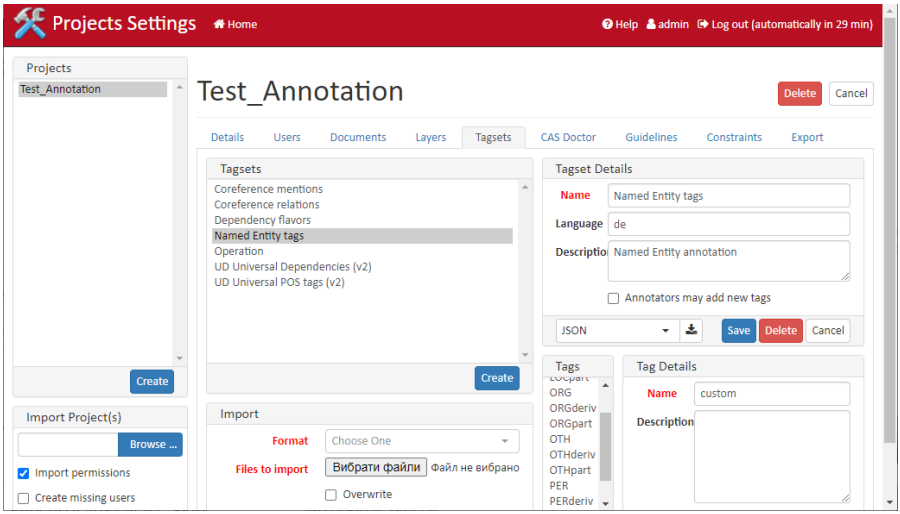
Зазначається, що мета візиту — побачити та оцінити рівень військового співробітництва, підготовки та взаємодії між морськими підрозділами спеціальних операцій України та Сполучених Штатів Америки.

Разом із командувачем ССО ЗСУ генерал-майором Григорієм Галаганом вони оглянули навчально-тренувальну базу центру, обговорили нові можливості для навчання операторів та покращення взаємодії у сфері виконання завдань підрозділами.

«Навчання, які ми проводимо, мають важливе значення для підтримки взаємодії з нашими союзниками, партнерами та друзями. Цей навчальний центр надає нам можливість тренуватися разом і вчитися один у одного в стратегічно важливому регіоні», – цитує Тейбора пресслужба 73-го морського центру.

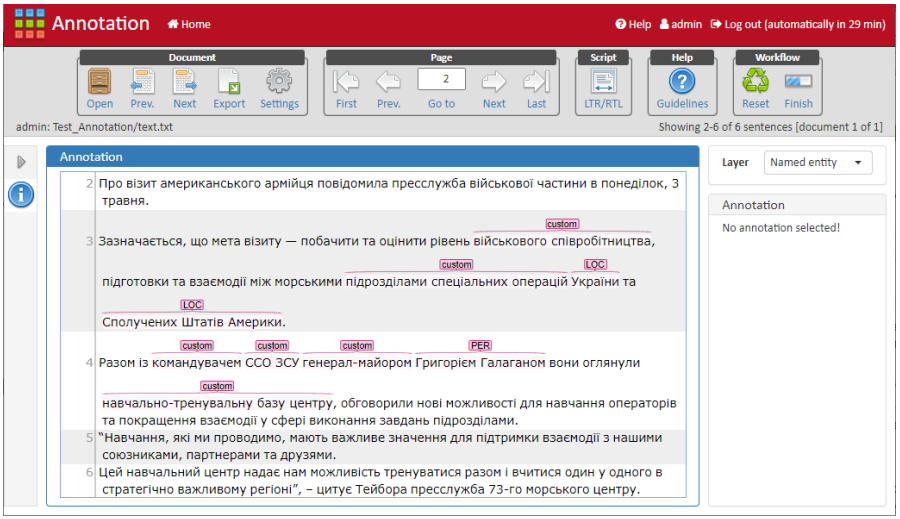
Отже, тепер давайте подивимось, як створити нову сутність. Для цього виконайте наступні кроки:

1. Перейдіть на вкладку «*Tagsets*».
2. Виберіть *Named Entity* зі списку наборів тегів.
3. Для створення нового тегу, наприклад «*custom*», натисніть «*Create*» в розділі «*Tags*».



Таким же чином ви можете створити свою кастомну сутність.

Тепер в меню проекту виберіть Анотація. З'явиться нове спливаюче вікно, виберіть документ, який ви хочете анотувати звідти.



Як тільки ви закінчите з анотацією, натисніть «*Export*», виберіть «*WebAnno TSV v3.2*» у спливаючому вікні та експорту його.

Тепер давайте почнемо кодування для створення остаточних форматованих користувальницьких навчальних даних у форматі *Spacy* для навчання користувальницької моделі розпізнавання іменованих сутностей (*NER*) з використанням *Spacy*.

```
# Підготовка тренувальних даних в форматі Spacy
TRAIN_DATA = []
ent_list = []
   from web_anno_tsv import open_web_anno_tsv

   tsv = './text.tsv'

   with open_web_anno_tsv(tsv) as f:
     for i, sentence in enumerate(f):
         #print(f"Sentence {i}:", sentence.text)
         ent_list_sen = []
         for j, annotation in
enumerate(sentence.annotations):
                 ent_list_sen.append((annotation.start,anno
         tation.stop,annotation.label))
            ent_list.append(ent_list_sen)
         ent_dic = {}
         ent_dic['entities'] = ent_list[-1]
         # Підготуйте підсумкові дані навчання
         TRAIN_DATA.append([sentence.text,ent_dic])
```

```
[['Командувач Сил спеціальних операцій США у Європі Девід Тейбор відвідав 73-й морський
центр спеціального призначення імені кошового отамана Антіна Головатого.', {'entities':
[(0, 10, 'custom'), (11, 35, 'custom'), (36, 39, 'LOC'), (42, 48, 'LOC'), (49, 61, 'PER'),
(71, 157, 'custom')]}], ['Про візит американського армійця повідомила пресслужба
військової частини в понеділок, 3 травня.', {'entities': []}], ['Зазначається, що мета
візиту – побачити та оцінити рівень військового співробітництва, підготовки та взаємодії
між морськими підрозділами спеціальних операцій України та Сполучених Штатів Америки.',
{'entities': [(58, 85, 'custom'), (125, 158, 'custom'), (159, 166, 'LOC'), (170, 195,
'LOC')]}], ['Разом із командувачем ССО ЗСУ генерал-майором Григорієм Галаганом вони
оглянули навчально-тренувальну базу центру, обговорили нові можливості для навчання
операторів та покращення взаємодії у сфері виконання завдань підрозділами.', {'entities':
[(9, 21, 'custom'), (22, 29, 'custom'), (30, 45, 'custom'), (46, 65, 'PER'), (80, 113,
'custom')]}], ['“Навчання, які ми проводимо, мають важливе значення для підтримки
взаємодії з нашими союзниками, партнерами та друзями.', {'entities': []}], ['Цей
навчальний центр надає нам можливість тренуватися разом і вчитися один у одного в
стратегічно важливому регіоні”, – цитує Тейбора пресслужба 73-го морського центру.',
{'entities': []}]]
```

Тепер давайте спробуємо навчити нову свіжу модель *NER*, використовуючи підготовлені призначені для користувача дані *NER*. Визначте змінні, необхідні для обробки навчальної моделі.

```
model = None
model_dir=Path("model_ner")
n_iter=100if model is not None:
    nlp = spacy.load(model)
    print("Loaded model '%s'" % model)
else:
    nlp = spacy.blank('uk')
    print("Created blank 'uk' model")#Потім завантажте
порожню модель для процесу, що виконує дію NER, і
налаштуйте конвеєр тільки з NER за допомогою функції
create_pipe.if 'ner' not in nlp.pipe_names:
    ner = nlp.create_pipe('ner')
    nlp.add_pipe(ner, last=True)
else:
    ner = nlp.get_pipe('ner')for _, annotations in
TRAIN_DATA:
    for ent in annotations.get('entities'):
        ner.add_label(ent[2])other_pipes = [pipe for pipe
in nlp.pipe_names if pipe != 'ner']
with nlp.disable_pipes(*other_pipes):  # only train NER
    optimizer = nlp.begin_training()
    for itn in range(n_iter):
        random.shuffle(TRAIN_DATA)
        losses = {}
        for text, annotations in tqdm(TRAIN_DATA):
            nlp.update(
                [text],
                [annotations],
                drop=0.5,
                sgd=optimizer,
                losses=losses)
        print(losses)
```

Щоб протестувати навчену модель,

```
for text, _ in TRAIN_DATA:
        doc = nlp(text)
        print("Entities", [(ent.text, ent.label_) for
```

```
ent in doc.ents])
            print("Tokens", [(t.text, t.ent_type_,
t.ent_iob) for t in doc])
```

```
Entities [('військового співробітництва', 'custom'), ('підрозділами спеціальних операцій', 'custom'),
('України', 'LOC'), ('Сполучених Штатів Америки', 'LOC')]
Tokens [('Зазначається', '', 2), (',', '', 2), ('що', '', 2), ('мета', '', 2), ('візиту', '', 2), ('–', '',
2), ('побачити', '', 2), ('та', '', 2), ('оцінити', '', 2), ('рівень', '', 2), ('військового', 'custom', 3),
('співробітництва', 'custom', 1), (',', '', 2), ('підготовки', '', 2), ('та', '', 2), ('взаємодії', '', 2),
('між', '', 2), ('морськими', '', 2), ('підрозділами', 'custom', 3), ('спеціальних', 'custom', 1),
('операцій', 'custom', 1), ('України', 'LOC', 3), ('та', '', 2), ('Сполучених', 'LOC', 3), ('Штатів', 'LOC',
1), ('Америки', 'LOC', 1), ('.', '', 2)]
Entities [('Командувач', 'custom'), ('Сил спеціальних операцій', 'custom'), ('США', 'LOC'), ('Європі',
'LOC'), ('Девід Тейбор', 'PER'), ('73-й морський центр спеціального призначення імені кошового отамана
Антіна Головатого.', 'custom')]
Tokens [('Командувач', 'custom', 3), ('Сил', 'custom', 3), ('спеціальних', 'custom', 1), ('операцій',
'custom', 1), ('США', 'LOC', 3), ('у', '', 2), ('Європі', 'LOC', 3), ('Девід', 'PER', 3), ('Тейбор', 'PER',
1), ('відвідав', '', 2), ('73-й', 'custom', 3), ('морський', 'custom', 1), ('центр', 'custom', 1),
('спеціального', 'custom', 1), ('призначення', 'custom', 1), ('імені', 'custom', 1), ('кошового', 'custom',
1), ('отамана', 'custom', 1), ('Антіна', 'custom', 1), ('Головатого', 'custom', 1), ('.', 'custom', 1)]
Entities [('командувачем', 'custom'), ('ССО ЗСУ', 'custom'), ('генерал-майором', 'custom'), ('Григорієм
Галаганом', 'PER'), ('навчально-тренувальну базу центру', 'custom')]
Tokens [('Разом', '', 2), ('із', '', 2), ('командувачем', 'custom', 3), ('ССО', 'custom', 3), ('ЗСУ',
'custom', 1), ('генерал', 'custom', 3), ('-', 'custom', 1), ('майором', 'custom', 1), ('Григорієм', 'PER',
3), ('Галаганом', 'PER', 1), ('вони', '', 2), ('оглянули', '', 2), ('навчально', 'custom', 3), ('-',
'custom', 1), ('тренувальну', 'custom', 1), ('базу', 'custom', 1), ('центру', 'custom', 1), (',', '', 2),
('обговорили', '', 2), ('нові', '', 2), ('можливості', '', 2), ('для', '', 2), ('навчання', '', 2),
('операторів', '', 2), ('та', '', 2), ('покращення', '', 2), ('взаємодії', '', 2), ('у', '', 2), ('сфері',
'', 2), ('виконання', '', 2), ('завдань', '', 2), ('підрозділами', '', 2), ('.', '', 2)]
Entities []
Tokens [('“', '', 2), ('Навчання', '', 2), (',', '', 2), ('які', '', 2), ('ми', '', 2), ('проводимо', '',
2), (',', '', 2), ('мають', '', 2), ('важливе', '', 2), ('значення', '', 2), ('для', '', 2), ('підтримки',
'', 2), ('взаємодії', '', 2), ('з', '', 2), ('нашими', '', 2), ('союзниками', '', 2), (',', '', 2),
('партнерами', '', 2), ('та', '', 2), ('друзями', '', 2), ('.', '', 2)]
Entities []
Tokens [('Цей', '', 2), ('навчальний', '', 2), ('центр', '', 2), ('надає', '', 2), ('нам', '', 2),
('можливість', '', 2), ('тренуватися', '', 2), ('разом', '', 2), ('і', '', 2), ('вчитися', '', 2), ('один',
'', 2), ('у', '', 2), ('одного', '', 2), ('в', '', 2), ('стратегічно', '', 2), ('важливому', '', 2),
('регіоні', '', 2), ('”', '', 2), (',', '', 2), ('–', '', 2), ('цитує', '', 2), ('Тейбора', '', 2),
```

Нарешті, збережіть модель на свій шлях, який зберігається в змінної *model_dir*.

```
if model_dir is not None:
    model_dir = Path(model_dir)
    if not model_dir.exists():
        model_dir.mkdir()
    nlp.to_disk(model_dir)
    print("Saved model to", model_dir)model =
spacy.load(model_dir)
```

2. Вкладання слів та семантична подібність

Давайте встановимо модель української мови *Spacy*. Для роботи з векторами потрібно використовувати велику модель.

```
!python -m spacy download uk_core_news_lg
import spacy
nlp = spacy.load("uk_core_news_lg")doc1 = nlp("Українські військові відбили атаки окупантів у районі восьми населених пунктів – Генштаб")for token in doc1:
  print(token.text, token.has_vector)
```

```
Українські True
військові  True
відбили    True
атаки      True
окупантів  True
у          True
районі     True
восьми     True
населених  True
пунктів    True
–          True
Генштаб    True
```

У прикладі, усі лексеми речення мають вектор, то можемо обчислити семантичну подібність речень:

```
doc2 = nlp("Війська РФ завдали ракетного удару по Дніпропетровщині, на місці прильоту працюють усі служби. Фото")doc3 = nlp("Як Європа допомагає Україні через фонд миру: пояснення посла ЄС")print(doc1, "<>", doc2, doc1.similarity(doc2))
print(doc1, "<>", doc3, doc1.similarity(doc3))
```

```
Українські військові відбили атаки окупантів у районі восьми населених пунктів – Генштаб
<> Війська РФ завдали ракетного удару по Дніпропетровщині, на місці прильоту працюють усі
служби. Фото 0.7152659483500141
Українські військові відбили атаки окупантів у районі восьми населених пунктів – Генштаб
<> Як Європа допомагає Україні через фонд миру: пояснення посла ЄС 0.3512228591622778
```

Тут ми використовували вже готову модель з 200 тис. унікальними векторами.

Навчання ваших власних вбудованих слів не повинно бути складним завданням і, для конкретних проблемних областей, призведе до підвищення продуктивності в порівнянні з попередньо навченими моделями.

Розглянемо приклад з розпізнавання належності назви посади

до категорії менеджмент та обслуговування.

```
import pandas as pdcorpus =
pd.read_csv('https://drive.google.com/uc?export=download&i
d=1JSUVmAkz8ECIMcJZGt8AnscBs0jmSLee', header = 0, sep =
';')
corpus
```

```
        titles                                          is_service
   0    Авербандник                                     1
   1    Авіаційний механік з планера та двигунів        1
   2    Авіаційний механік з приладів та електроустатк...  1
   3    Авіаційний механік з радіоустаткування          1
   4    Авіаційний технік (механік) з парашутних та ав...  1
   ...  ...  ...
   9097 Юрист                                           1
   9098 Юрист-міжнародник                               1
   9099 Юстирувальник деталей та приладів               1
   9100 Юстирувальник оптичних приладів                 1
   9101 Юстирувальник                                   1
```

Отже, це наш набір даних з колонкою *is_service* (0 — менеджмент, 1 — обслуговування), яку потрібно вбудувати в модель.

Виконайте попередню обробку даних: потрібно привести к малому регістру та видалити пунктуацію.

```
import stringcorpus['titles'] =
corpus['titles'].apply(lambda x :
str(x).lower().strip())corpus['titles'] =
corpus['titles'].apply(lambda x:''.join([i for i in x if i
not in string.punctuation]))
corpus['titles']
```

```
   0                                          авербандник
   1                 авіаційний механік з планера та двигунів
   2        авіаційний механік з приладів та електроустатк...
   3                  авіаційний механік з радіоустаткування
   4        авіаційний технік механік з парашутних та авар...
                                   ...
   9097                                                 юрист
   9098                                        юристміжнародник
   9099                      юстирувальник деталей та приладів
   9100                        юстирувальник оптичних приладів
   9101                                          юстирувальник
   Name: titles, Length: 9102, dtype: object
```

Створимо вектори до «*titles*», для простоти напишемо функцію:

```
def get_vector(x):
  doc = nlp(x)
  vector = doc.vector
return vectorcorpus['vectors'] =
corpus['titles'].apply(lambda x: get_vector(x))
corpus.head()
```

```
                  titles  is_service                                          vectors
0            авербандник           1  [0.4767424, 0.030759603, -2.3752034, 0.0990520...
1  авіаційний механік з...         1  [0.50640464, 0.88554525, 0.45917043, 0.9238674...
2  авіаційний механік з...         1  [0.80926394, 0.974323, 0.465778, 1.3394418, -4...
3  авіаційний механік з...         1  [0.8235947, 1.5725664, 0.21599993, 1.059803, -...
4  авіаційний технік мех...        1  [1.3383067, 0.47077453, -0.004781276, 0.634015...
```

Підготуємо до навчання для нашої моделі. По-перше реорганізуємо масив векторів в один стовпець.

```
X = corpus['vectors'].to_numpy()
X = X.reshape(-1, 1)
X.shape
```

```
(9102, 1)
```

Тепер потрібно перетворити ці дані, щоб зробити розмірність 300.

```
import numpy as np
X = np.concatenate(np.concatenate(X, axis=0), axis =
0).reshape(-1, 300)
X.shape
```

```
(9102, 300)
```

Після цього розділимо на два набори: тренування та тестування.

```
from sklearn.model_selection import train_test_split
y = corpus['is_service']
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=0 )
X_train.shape, X_test.shape
```

```
((6371, 300), (2731, 300))
```

Питання дослідження визначатиме модель машинного навчання для виконання, і це сильно залежить від ваших тренувальних та тестових змінних. Чи розумієте ви, що станеться, коли вам потрібно буде перевірити кожну модель машинного навчання для ваших даних? — Тож це дуже складне завдання.

Але для вирішення цієї проблеми існує пакет *Python LazyPredict* – чудовий інструмент, який автоматично запускає дані через різні типи моделей і повертає інформацію про продуктивність кожної моделі.

```
!pip install lazypredict
from lazypredict.Supervised import LazyClassifier

clf = LazyClassifier(verbose=0, ignore_warnings=True,
custom_metric=None, classifiers="all", random_state=0)

models, predictions = clf.fit(X_train, X_test, y_train,
y_test)

models
```

Model	Accuracy	Balanced Accuracy	ROC AUC	F1 Score	Time Taken
PassiveAggressiveClassifier	0.99	0.97	0.97	0.99	0.38
LogisticRegression	0.99	0.96	0.96	0.99	0.61
LinearSVC	0.98	0.96	0.96	0.98	0.61
Perceptron	0.98	0.95	0.95	0.98	0.22
SVC	0.98	0.94	0.94	0.98	2.48
CalibratedClassifierCV	0.98	0.93	0.93	0.98	3.19
LinearDiscriminantAnalysis	0.97	0.91	0.91	0.97	0.66
SGDClassifier	0.97	0.91	0.91	0.97	1.15
GaussianNB	0.91	0.89	0.89	0.91	0.12
KNeighborsClassifier	0.96	0.89	0.89	0.96	1.04
AdaBoostClassifier	0.96	0.89	0.89	0.96	14.90
LGBMClassifier	0.97	0.89	0.89	0.97	9.86
BernoulliNB	0.89	0.88	0.88	0.90	0.40
NearestCentroid	0.88	0.88	0.88	0.90	0.15
RidgeClassifier	0.97	0.88	0.88	0.97	0.18
RidgeClassifierCV	0.97	0.88	0.88	0.97	0.54
XGBClassifier	0.96	0.88	0.88	0.96	12.37
BaggingClassifier	0.95	0.85	0.85	0.95	22.11
ExtraTreesClassifier	0.95	0.82	0.82	0.95	1.71
RandomForestClassifier	0.95	0.82	0.82	0.95	9.55
DecisionTreeClassifier	0.92	0.79	0.79	0.92	3.76
QuadraticDiscriminantAnalysis	0.92	0.79	0.79	0.91	0.51
ExtraTreeClassifier	0.90	0.76	0.76	0.90	0.11
LabelSpreading	0.14	0.51	0.51	0.05	2.86
LabelPropagation	0.14	0.51	0.51	0.05	2.15
DummyClassifier	0.87	0.50	0.50	0.82	0.10

Отже для наших змінних найкращою моделлю машиного навчання буде PassiveAggressiveClassifier.

```
  from sklearn.linear_model import
PassiveAggressiveClassifier
from sklearn.metrics import classification_reportclf =
PassiveAggressiveClassifier(C = 0.5, random_state = 5)
clf.fit(X_train, y_train)y_predict = clf.predict(X_test)
print(classification_report(y_test, y_predict))
```

	precision	recall	f1-score	support
0	0.95	0.91	0.93	344
1	0.99	0.99	0.99	2387
accuracy			0.98	2731
macro avg	0.97	0.95	0.96	2731
weighted avg	0.98	0.98	0.98	2731

Наостанок збережемо модель для можливості переносу та виконання нових прогнозів. Ви можете використовувати операцію *pickle*, щоб серіалізувати свої моделі машинного навчання у файл.

```
  import pickle
pickle.dump(clf, open('model_service.pkl', 'wb'))model =
pickle.load(open('model_service.pkl',
'rb'))model.predict(np.array(get_vector('Керівник відділу
продажу')).reshape(1, -1))
```

```
array([0])
```

5.4. Контрольні запитання

1. Напишіть визначення термінологічних сполучень «розпізнавання іменованих сутностей» та «вкладання слів».

2. За допомогою якої бібліотеки було використано попередньо навчену модель *NER*. Опишіть її функціонал.

3. Яка технологія дозволяє ефективно використовувати вбудовування слів з використанням неглибокої нейронної мережі. Яким чином забезпечується доступ до технології в Python?

4. Під час роботи на лабораторній роботі було ознайомлено з інтерфейсом веб-додатку WebAnno. Докладно розпишіть як створюються кастомні сутності в програмі.

5. Як саме вирішується питання перевірки кожної моделі машинного навчання для ваших даних?

Лабораторна робота №6. Штучні нейронні мережі

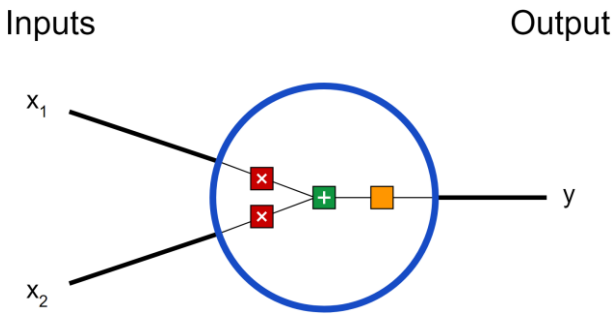
6.1. Мета роботи

Навчитися створювати штучну нейронну мережу в Python.

6.2. Теоретичні відомості

1. Будівельні блоки: нейрони

По-перше, ми повинні поговорити про нейрони, основну одиницю нейронної мережі. Нейрон приймає вхідні дані, обчислює їх і створює один вихід. Ось як виглядає нейрон із двома входами:



Тут відбувається 3 речі. Спочатку кожен вхід множиться на вагу: ■

$x_1 \rightarrow x_1 * w_1$

$x_2 \rightarrow x_2 * w_2$

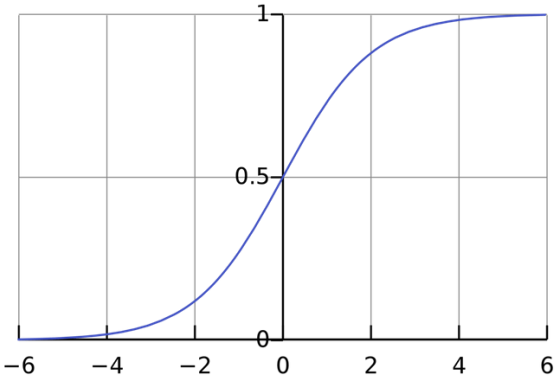
Потім зважені входи складаються, і до них додається значення порога b: ■

$(x_1 * w_1) + (x_2 * w_2) + b$

Нарешті, сума передається через функцію активації: ■

$y = f(x_1 * w_1 + x_2 * w_2 + b)$

Функція активації використовується для перетворення необмеженого входу на вихід, який має гарну, передбачувану форму. Часто використовуваною функцією активації є сигмоїдна функція:



Сігмоїда видає результати в інтервалі (0, 1). Можна уявити, що вона «упаковує» інтервал від мінус нескінченності до плюс нескінченності (0, 1): великі негативні числа перетворюються на числа, близькі до 0, а великі позитивні – до 1.

Простий приклад

Допустимо, наш двовходовий нейрон використовує сигмоїдну функцію активації та має наступні параметри:

$w = [0, 1]$ $b = 4$

w=[0, 1] – це лише запис w1=0, w2=1 у векторному вигляді. Тепер поставимо нашому нейрону вхідні дані: x=[2, 3]. Ми використовуємо скалярний добуток векторів, щоб записати формулу в стислому вигляді:

Наш нейрон видав 0.999 на входах x=[2, 3]. От і все! Процес передачі значень входів далі, щоб отримати вихід, називається прямим зв'язком (feed forward).

6.3. Порядок виконання роботи

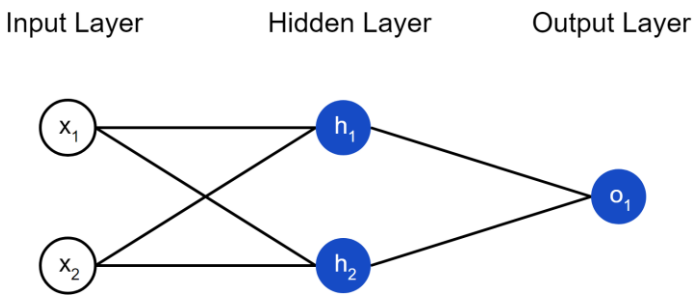
Завантажуємо необхідну для роботи бібліотеку *numpy* та створимо класс *Neuron* з прикладними до нього функціями:

```
import numpy as np

def sigmoid(x):
  # Наша функція активації: f(x) = 1 / (1 + e^(-x))
  return 1 / (1 + np.exp(-x))

class Neuron:
  def __init__(self, weights, bias):
    self.weights = weights
    self.bias = bias

  def feedforward(self, inputs):
    # Введіть вагу, додайте зсув, а потім скористайтеся
функцією активації
    total = np.dot(self.weights, inputs) + self.bias
    return sigmoid(total)

weights = np.array([0, 1]) # w1 = 0, w2 = 1
bias = 4                   # b = 4
n = Neuron(weights, bias)

x = np.array([2, 3])       # x1 = 2, x2 = 3
print(n.feedforward(x))    # 0.9990889488055994
```

Врешті отримуємо ту саму відповідь.

Об’єднання нейронів у нейронну мережу

Нейронна мережа – це не що інше, як група нейронів, з'єднаних разом. Ось як може виглядати проста нейронна мережа:



Ця мережа має два входи, прихований шар з двома нейронами (*h1* і *h2*) і вихідний шар з одним нейроном (*o1*). Зверніть увагу, що входи для *o1* – це виходи з *h1* та *h2*. Саме це створює з нейронів мережу.

Прямий зв'язок

Давайте використаємо мережу, зображену вище, і вважатимемо, що це нейрони мають однакові ваги `w=[0, 1]`, однакові порогові значення `b=0`, і однакову функцію активації – сигмоїду. Нехай *h1*, *h2* та *o1* позначають вихідні значення відповідних нейронів.

Що вийде, якщо ми подамо на вхід `x = [2, 3]`?

$$h_1 = h_2 = f(w * x + b) = f\big((0 * 2) + (1 * 3) + 0\big) = f(3) = 0.9526$$
$$o_1 = f(w * [h_1, h_2] + b) = f\big((0 * h_2) + (1 * 3) + 0\big) = f(0.9526) = 0.7216$$

Нейронна мережа може мати **будь-яку кількість шарів** і в цих шарах може бути **будь-яка кількість нейронів**. Основна ідея залишається такою ж: передаємо вхідні дані по нейронах мережі, доки не отримуємо вихідні значення. Для простоти будемо використовувати мережу, наведену вище, до кінця лабораторної.

Пишемо код нейронної мережі

Давайте реалізуємо прямий зв'язок нашої нейронної мережі.

```
import numpy as np

# ... вставте сюди код із попереднього розділу

class OurNeuralNetwork:
  '''
  Нейронна мережа з:
     - 2 входами
     - прихованим шаром з 2 нейронами (h1, h2)
     - вихідним шаром з 1 нейроном (o1)
   Усі нейрони мають однакові ваги та пороги:
     - w = [0, 1]
```

```
      - b = 0
  '''
  def __init__(self):
    weights = np.array([0, 1])
    bias = 0

    # Використовуємо клас Neuron із попереднього розділу
    self.h1 = Neuron(weights, bias)
    self.h2 = Neuron(weights, bias)
    self.o1 = Neuron(weights, bias)

  def feedforward(self, x):
    out_h1 = self.h1.feedforward(x)
    out_h2 = self.h2.feedforward(x)
    out_o1 = self.o1.feedforward(np.array([out_h1,
out_h2]))

    return out_o1

network = OurNeuralNetwork()
x = np.array([2, 3])
print(network.feedforward(x)) # 0.7216325609518421
```

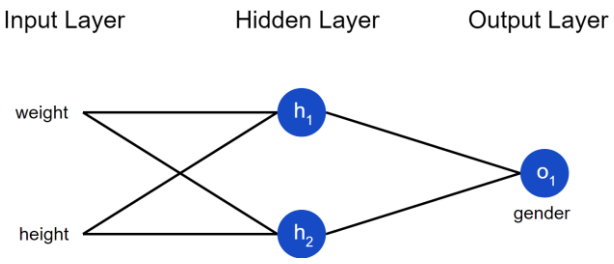
Оскільки результат знову співпав, то наша мережа працює.

Навчаємо нейронну мережу

Припустимо, у нас є такі виміри:

Ім’я	Вага (у фунтах)	Зріст (у дюймах)	Стать
Аліса	133 (54.4 кг)	65 (165,1 см)	Ж
Боб	160 (65,44 кг)	72 (183 см)	М
Чарлі	152 (62.2 кг)	70 (178 см)	М
Діана	120 (49 кг)	60 (152 см)	Ж

Давайте навчимо нашу нейронну мережу передбачати стать людини за її зростанням (*height*) і вагою (*weight*).



Виставимемо чоловічу стать як 0, жіночу – як 1, а також зрушимо дані, щоб їх було простіше використовувати:

Ім'я	Вага (мінус 135)	Зріст (мінус 66)	Стать
Аліса	-2	-1	1
Боб	25	6	0
Чарлі	17	4	0
Діана	-15	-6	1

Примітка. Були обрані величини зрушень (135 і 66) так, щоб числа виглядали простіше. Зазвичай зрушують середнє значення.

Втрати

Перш ніж навчати нашу нейронну мережу, нам потрібно якось виміряти, наскільки «добре» вона працює, щоб вона змогла працювати «краще». Цей вимір і є втратою (*loss*).

Ми використовуємо для розрахунку втрат середню квадратичну помилку (*mean squared error*, *MSE*):

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_{true} - y_{pred})^2$$

Давайте розглянемо всі використовувані змінні:

- n – це кількість вимірів, у нашому випадку 4 (Аліса, Боб, Чарлі та Діана).
- y представляє передбачуване значення, Стать.
- y_{true} – справжнє значення змінної («правильна відповідь»). Наприклад, для Аліси y_{true} дорівнюватиме 1 (жіноча стать).

- y_{pred}– передбачене значення змінної. Це те, що видасть наша нейронна мережа.

$(y_{true} - y_{pred})^2$ називається квадратичною помилкою. Наша функція втрати просто бере середнє значення всіх квадратичних помилок – тому вона і називається середньою квадратичною помилкою. Чим кращими будуть наші прогнози, тим меншими будуть наші втрати!

Приклад розрахунку втрат

Припустимо, що наша мережа завжди повертає 0 – інакше кажучи, вона впевнена, що всі люди чоловіки. Наскільки великі будуть наші втрати?

Ім'я	y_{true}	y_{pred}	$(y_{true} - y_{pred})^2$
Аліса	1	0	1
Боб	0	0	0
Чарлі	0	0	0
Діана	1	0	1

$$MSE = \frac{1}{4}(1 + 0 + 0 + 1) = 0.5$$

Пишемо функцію середньої квадратичної помилки

Ось невеликий шматок коду, який розрахує наші втрати.

```
import numpy as np

def mse_loss(y_true, y_pred):
   # y_true та y_pred - масиви numpy однакової довжини.
   return ((y_true - y_pred) ** 2).mean()

y_true = np.array([1, 0, 0, 1])
y_pred = np.array([0, 0, 0, 0])

print(mse_loss(y_true, y_pred)) # 0.5
```

Навчаємо нейронну мережу

Тепер ми маємо чітку мету: **мінімізувати втрати** нейронної мережі. Ми знаємо, що можемо змінювати ваги та пороги нейронів, щоб змінити її передбачення, але як нам робити це таким чином, щоб мінімізувати втрати?

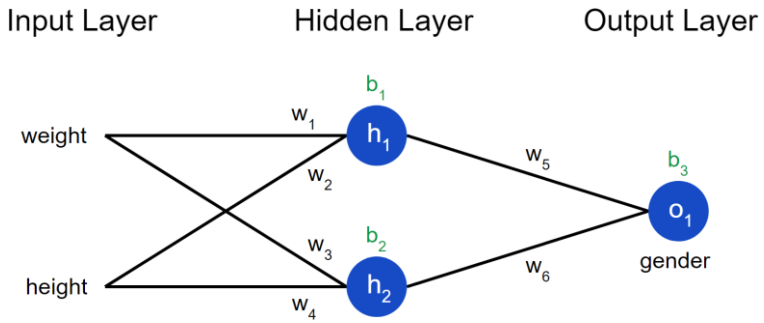
Для простоти уявімо, що в нашому наборі даних тільки одна Аліса.

Ім'я	Вага (мінус 135)	Зріст (мінус 66)	Стать
Аліса	-2	-1	1

Тоді середня квадратична помилка буде квадратичною помилкою лише Аліси:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_{true} - y_{pred})^2 = (y_{true} - y_{pred})^2 = (1 - y_{pred})^2$$

Інший метод – це розглядати функцію втрат як функцію від ваги та порогів. Давайте відзначимо всі ваги та пороги нашої нейронної мережі:



Тепер ми можемо записати функцію втрат як функцію від кількох змінних:

$$L(w_1, w_2, w_3, w_4, w_5, w_6, b_1, b_2, b_3)$$

Припустимо, ми хочемо налаштувати w_1. Як зміниться значення втрати L у разі зміниw_1? Це питання може відповісти часткова похідна **dL/dw1**.

Насамперед, давайте перепишемо цю часткову похідну через **dypred/dw1**, скориставшись ланцюговим правилом:

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial y_{pred}} * \frac{\partial y_{pred}}{\partial w_1}$$

Ми можемо розрахувати **dL/dypred**, оскільки ми вже з'ясували вище, що **L=(1-ypred)²**:

$$\frac{\partial L}{\partial y_{pred}} = \frac{\partial (1 - y_{pred})^2}{\partial y_{pred}} = -2(1 - y_{pred})$$

Тепер давайте вирішимо, що робити з **dypred/dw1**. Позначаючи виходи нейронів, як раніше, h_1, h_2 та o_1, отримуємо:

$$y_{pred} = o_1 = f(w_5 h_1 + w_6 h_2 + b_3)$$

Згадайте, що $f()$ – це наша функція активації сигмоїда. Оскільки **w1** впливає тільки на **h1** (але не на **h2**), ми можемо знову використовувати ланцюгове правило і записати:

$$\frac{\partial y_{pred}}{\partial w_1} = \frac{\partial y_{pred}}{\partial h_1} * \frac{\partial h_1}{\partial w_1}$$

$$\frac{\partial y_{pred}}{\partial h_1} = w_5 * f'(w_1 h_1 + w_2 h_2 + b_1)$$

Ми можемо зробити те саме для **dh1/dw1**, знову застосовуючи ланцюгове правило:

$$h_1 = f(w_1 x_1 + w_2 x_2 + b_1)$$

$$\frac{\partial h_1}{\partial w_1} = x_1 * f'(w_1 h_1 + w_2 h_2 + b_1)$$

У цій формулі $\mathbf{x_1}$ – це вага, а $\mathbf{x_2}$ – ріст. Залишилося визначити $f'(x)$ – похідну сигмоїдної функції:

$$f(x) = \frac{1}{1 + e^{-x}}$$
$$f'(x) = \frac{e^{-x}}{(1 + e^{-x})^2} = f(x) * (1 - f(x))$$

Ми змогли розкласти **dL/dw₁** на кілька частин, які ми можемо розрахувати:

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial y_{pred}} * \frac{\partial y_{pred}}{\partial h_1} * \frac{\partial h_1}{\partial w_1}$$

Такий метод розрахунку часткових похідних «від кінця до початку» називається **методом зворотного розповсюдження** (*backpropagation*).

Приклад. Визначення часткової похідної

Ми, як і раніше, вважаємо, що наш набір даних складається з однієї Аліси:

Ім'я	Вага (мінус 135)	Зріст (мінус 66)	Стать
Аліса	-2	-1	1

Ініціалізуємо всі ваги як 1, а всі пороги як 0. Якщо ми виконаємо прямий прохід нейронною мережею, то отримаємо:

$$h_1 = f(w_1 h_1 + w_2 h_2 + b_1) = f(-2 + (-1) + 0) = 0.0474$$
$$h_2 = f(w_3 h_1 + w_4 h_2 + b_2) = 0.0474$$
$$o_1 = f(w_5 h_1 + w_6 h_2 + b_3) = f(0.0474 + 0.0474 + 0) = 0.524$$

Наша мережа видає $y_{pred} = 0.524$, що знаходиться приблизно посередені між Чоловічою статтю (0) та Жіночою (1). Давайте розрахуємо **dL/dw₁**:

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial y_{pred}} * \frac{\partial y_{pred}}{\partial h_1} * \frac{\partial h_1}{\partial w_1}$$

$$\frac{\partial L}{\partial y_{pred}} = -2\left(1 - y_{pred}\right) = -2(1 - 0.524) = -0.952$$

$$\frac{\partial y_{pred}}{\partial h_1} = w_5 * f'(w_1 h_1 + w_2 h_2 + b_1) = f'(0.0474 + 0.0474 + 0)$$

$$= f(0.948) * \left(1 - f(0.948)\right) = 0.249$$

$$\frac{\partial h_1}{\partial w_1} = x_1 * f'(w_1 h_1 + w_2 h_2 + b_1) = -2 * f(-3) * \left(1 - f(-3)\right) = -0.0904$$

$$\frac{\partial L}{\partial w_1} = -0.952 * 0.249 * -0.0904 = 0.0214$$

Результат каже нам, що зі збільшенням $\boldsymbol{w_1}$, функція помилки трохи підвищується.

Стохастичний градієнтний спуск

Використаємо алгоритм оптимізації під назвою стохастичний градієнтний спуск (*stochastic gradient descent*), який визначить, як ми змінюватимемо наші ваги та пороги для мінімізації втрат. Фактично він полягає в наступній формулі оновлення:

$$w_1 \leftarrow w_1 - \eta \frac{\partial L}{\partial w_1}$$

$\eta(eta)$ – константа, названа швидкістю навчання (*learning rate*)

Швидкість навчання визначає, як швидко наша мережа навчається. Все, що ми робимо – це віднімаємо **eta*dL/dw1** з **w1**:

- Якщо **dL/dw1** позитивна, **w1** зменшиться, що зменшить **L**.
- Якщо **dL/dw1** негативна, **w1** збільшиться, що зменшить **L**.

Якщо ми зробимо те саме для кожної ваги та порога в мережі, втрати будуть поступово зменшуватися, і наша мережа видаватиме точніші результати.

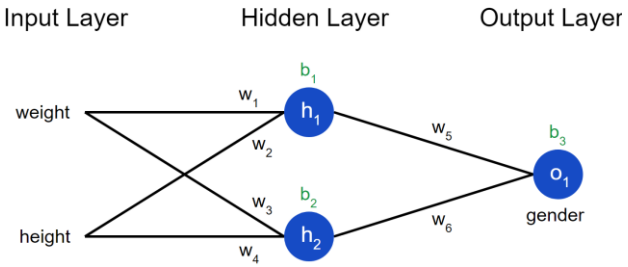
Процес навчання мережі виглядатиме приблизно так:

1. Вибираємо **одне** спостереження із набору даних. Саме те, що ми працюємо лише з одним спостереженням, робить наш градієнтний спуск стохастичним.
2. Вважаємо всі приватні похідні функції втрат за всіма вагами та порогами (**dL/dw_1**, **dL/dw_2** і т.д.)
3. Використовуємо формулу оновлення, щоб оновити значення кожної ваги та порога.
4. Знову переходимо до кроку 1.

Пишемо код усієї нейронної мережі

Час реалізувати всю нейронну мережу.

Ім'я	Вага (мінус 135)	Зріст (мінус 66)	Стать
Аліса	-2	-1	1
Боб	25	6	0
Чарлі	17	4	0
Діана	-15	-6	



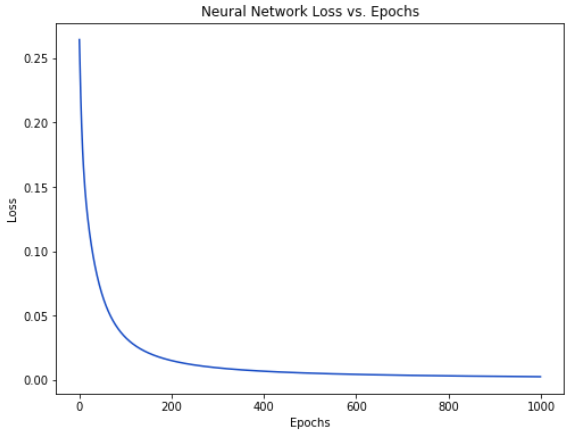

```
import numpy as np

def sigmoid(x):
   # Сигмоїдна функція активації: f(x) = 1 / (1 + e^(-x))
   return 1 / (1 + np.exp(-x))

def deriv_sigmoid(x):
   # Похідна сигмоїди: f'(x) = f(x) * (1 - f(x))
   fx = sigmoid(x)
   return fx * (1 - fx)

def mse_loss(y_true, y_pred):
   # y_true та y_pred - масиви numpy однакової довжини.
```

```
    return ((y_true - y_pred) ** 2).mean()

class OurNeuralNetwork:
    '''
    def __init__(self):
      #Вага
      self.w1 = np.random.normal()
      self.w2 = np.random.normal()
      self.w3 = np.random.normal()
      self.w4 = np.random.normal()
      self.w5 = np.random.normal()
      self.w6 = np.random.normal()

      # Пороги
      self.b1 = np.random.normal()
      self.b2 = np.random.normal()
      self.b3 = np.random.normal()

    def feedforward(self, x):
      # x is a numpy array with 2 elements.
      h1 = sigmoid(self.w1 * x[0] + self.w2 * x[1] +
self.b1)
      h2 = sigmoid(self.w3 * x[0] + self.w4 * x[1] +
self.b2)
      o1 = sigmoid(self.w5 * h1 + self.w6 * h2 + self.b3)
      return o1

    def train(self, data, all_y_trues):
      '''
      - data - масив numpy (n x 2) numpy, n = до
спостережень у наборі.
      - all_y_trues – масив numpy з n елементами.
        Елементи all_y_trues відповідають спостереженням
data.
      '''
      learn_rate = 0.1
      epochs = 1000 # скільки разів пройти по всьому набору
даних

      for epoch in range(epochs):
        для x, y_true в zip(data, all_y_trues):
          # --- Прямий прохід (ці значення нам знадобляться
пізніше)
```

```
            sum_h1 = self.w1 * x [0] + self.w2 * x [1] +
self.b1
            h1 = sigmoid(sum_h1)

            sum_h2 = self.w3 * x [0] + self.w4 * x [1] +
self.b2
            h2 = sigmoid(sum_h2)

            sum_o1 = self.w5 * h1 + self.w6 * h2 + self.b3
            o1 = sigmoid(sum_o1)
            y_pred = o1

            # --- Вважаємо приватні похідні.
            # --- Імена: d_L_d_w1 = "приватна похідна L по
w1"
            d_L_d_ypred = -2 * (y_true - y_pred)

            # Нейрон o1
            d_ypred_d_w5 = h1 * deriv_sigmoid(sum_o1)
            d_ypred_d_w6 = h2 * deriv_sigmoid(sum_o1)
            d_ypred_d_b3 = deriv_sigmoid(sum_o1)

            d_ypred_d_h1 = self.w5 * deriv_sigmoid(sum_o1)
            d_ypred_d_h2 = self.w6 * deriv_sigmoid(sum_o1)

            # Нейрон h1
            d_h1_d_w1 = x[0] * deriv_sigmoid(sum_h1)
            d_h1_d_w2 = x[1] * deriv_sigmoid(sum_h1)
            d_h1_d_b1 = deriv_sigmoid(sum_h1)

            # Нейрон h2
            d_h2_d_w3 = x[0] * deriv_sigmoid(sum_h2)
            d_h2_d_w4 = x[1] * deriv_sigmoid(sum_h2)
            d_h2_d_b2 = deriv_sigmoid(sum_h2)

            # --- Оновлюємо ваги та пороги
            # Нейрон h1
            self.w1 -= learn_rate * d_L_d_ypred *
d_ypred_d_h1 * d_h1_d_w1
            self.w2 -= learn_rate * d_L_d_ypred *
d_ypred_d_h1 * d_h1_d_w2
            self.b1 -= learn_rate * d_L_d_ypred *
d_ypred_d_h1 * d_h1_d_b1
```

```
            # Нейрон h2
            self.w3 -= learn_rate * d_L_d_ypred *
d_ypred_d_h2 * d_h2_d_w3
            self.w4 -= learn_rate * d_L_d_ypred *
d_ypred_d_h2 * d_h2_d_w4
            self.b2 -= learn_rate * d_L_d_ypred *
d_ypred_d_h2 * d_h2_d_b2

            # Нейрон o1
            self.w5 -= learn_rate * d_L_d_ypred *
d_ypred_d_w5
            self.w6 -= learn_rate * d_L_d_ypred *
d_ypred_d_w6
            self.b3 -= learn_rate * d_L_d_ypred *
d_ypred_d_b3

          # --- Вважаємо повні втрати наприкінці кожної епохи
          if epoch % 10 == 0:
            y_preds = np.apply_along_axis(self.feedforward,
1, data)
            loss = mse_loss(all_y_trues, y_preds)
            print("Epoch %d loss: %.3f" % (epoch, loss))

   # Визначимо набір даних
   data = np.array([
      [-2, -1], # Аліса
      [25, 6], # Боб
      [17, 4], # Чарлі
      [-15, -6], # Діана
   ])
   all_y_trues = np.array([
      1, # Аліса
      0, # Боб
      0, # Чарлі
      1, # Діана
   ])

   # Навчаємо нашу нейронну мережу
   network = OurNeuralNetwork()
   network.train(data, all_y_trues)
```

Під час навчання мережі її втрати поступово зменшуються.



Тепер ми можемо використовувати нашу мережу для передбачення статі:

```
# Робимо пару передбачень
emily = np.array([-7, -3]) # 128 фунтів (52.35 кг), 63 дюйми (160 см)
frank = np.array([20, 2]) # 155 pounds (63.4 кг), 68 inches (173 см)
print("Емілі: %.3f" % network.feedforward(emily)) # 0.951 - Ж
print("Френк: %.3f" % network.feedforward(frank)) # 0.039 - М
```

6.4. Контрольні запитання

1. Дайте визначення нейрону, основної одиниці нейронної мережі. Опишіть його складову.
2. Що таке функція сигмоїдної активації?
3. Як багато шарів та нейронів у кожному з шарів може мати нейронна мережа? Наведіть приклад простої трьохшарової мережі.
4. Опишіть алгоритм роботи функції зменшення втрат.
5. В чому полягає метод зворотного розповсюдження та стохастичний спуск градієнта?

Лабораторна робота №7. Обробка природної мови (NLP) у Python з кодом. Обробка природної мови

7.1. Мета роботи

Вивчити принцип роботи та створити програму розпізнавання мови на базі бібліотеки *SpeechRecognition*.

7.2. Теоретичні відомості

Розпізнавання мови, як випливає з назви, відноситься до автоматичного розпізнавання людської мови, є одним із найважливіших завдань у галузі взаємодії людини з комп'ютером. Якщо ви коли-небудь спілкувалися з *Amazon Alexa* або наказували *Siri* виконати завдання, ви вже випробували силу розпізнавання мови.

Розпізнавання мовлення має різні додатки – від автоматичної транскрипції мовних даних (наприклад, голосової пошти) до взаємодії з роботами у вигляді мови.

На даній лабораторній ви побачите, як ми можемо розробити дуже простий додаток для розпізнавання мови, здатний розпізнавати мову як з аудіофайлів, так і в режимі реального часу з мікрофона.

У *Python* було розроблено кілька бібліотек розпізнавання мови. Однак ми будемо використовувати бібліотеку *SpeechRecognition*, яка є найпростішою з усіх бібліотек.

7.3. Порядок виконання роботи

Встановлення бібліотеки SpeechRecognition

Виконайте наступну команду для встановлення бібліотеки:

```
pip install SpeechRecognition
```

Розпізнавання мовлення з аудіо файлів

У цьому розділі ви побачите, як ми можемо перекладати мову

з аудіофайлу до тексту. Аудіофайл, який ми будемо використовувати як вхідні дані, можна завантажити за цим посиланням. Завантажте файл у локальну файлову систему.

Першим кроком, як завжди, є імпорт необхідних бібліотек. У цьому випадку нам потрібно імпортувати щойно завантажену бібліотеку *speech_recognition*.

```
import speech_recognition as speech_recog
```

Для перетворення мови в текст нам потрібний єдиний клас – це клас *Recognizer* із модуля *speech_recognition*. Залежно від базового *API*, що використовується для перетворення мови на текст, клас *Recognizer* має такі методи:

1. *recognize_bing*(): Використовує *Microsoft Bing Speech API*
2. *recognize_google*(): Використовує *Google Speech API*
3. *recognize_google_cloud*(): Використовує *Google Cloud Speech API*
4. *recognize_houndify*(): Використовує *Houndify API* від *SoundHound*
5. *recognize_ibm*(): Використовує *IBM Speech to Text API*
6. *recognize_sphinx*(): Використовує *PocketSphinx API*

Серед усіх перерахованих способів метод *recognize_sphinx*() можна використовувати в автономному режимі для перекладу мови в текст.

Щоб розпізнати мову з аудіофайлу, ми повинні створити об'єкт класу модуля *AudioFile speech_recognition*. Шлях аудіофайлу, який потрібно перекласти в текст, передається в конструктор класу *AudioFile*. Виконайте наступний скрипт:

```
sample_audio = speech_recog.AudioFile('E:/Datasets/my_audio.wav')
```

У наведеному вище коді оновіть шлях до аудіофайлу, який потрібно розшифрувати.

Ми будемо використовувати метод *recognize_google*() для розшифровування наших аудіо файлів. Тим не менш, метод

recognize_google() вимагає об’єкта *AudioData* модуля *speech_recognition* як параметр. Щоб перетворити наш аудіофайл на об’єкт *AudioData*, ми можемо використовувати метод *record*() класу *Recognizer*. Нам потрібно передати об’єкт *AudioFile* методу *record*(), як показано нижче:

```
with sample_audio as audio_file:
    audio_content = recog.record(audio_file)
```

Тепер, якщо ви перевірите тип змінної *audio_content*, ви побачите, що вона має тип *speech_recognition.AudioData*.

```
type(audio_content)
```

Результат:

```
speech_recognition.AudioData
```

Тепер ми можемо просто передати об’єкт *audio_content* методу *recognize_google*() об’єкта класу *Recognizer*(), і аудіофайл буде перетворено на текст. Виконайте наступний скрипт:

```
recog.recognize_google(audio_content)
```

Результат:

```
'Bristol O2 left shoulder take the winding path to reach the lake no closely
the size of the gas tank degrees office 30 face before you go out the race
was badly strained and hung them the stray cat gave birth to kittens the
young girl gave no clear response the meal was called before the bells ring
what weather is in living'
```

Наведений вище результат показує текст аудіофайлу. Ви можете бачити, що файл не був на 100% правильно транскрибований, але точність досить висока.

Встановлення тривалості та значень усунення

Замість того, щоб транскрибувати повне мовлення, ви також можете транскрибувати певний сегмент аудіофайлу. Наприклад,

якщо ви хочете транскрибувати лише перші 10 секунд аудіофайлу, вам потрібно передати 10 як значення параметра *duration* метод *record*(). Подивіться на наступний скрипт:

```
sample_audio = speech_recog.AudioFile('E:/Datasets/my_audio.wav')
with sample_audio as audio_file:
    audio_content = recog.record(audio_file, duration=10)

recog.recognize_google(audio_content)
```

Результат:

```
'Bristol O2 left shoulder take the winding path to reach the lake no closely
the size of the gas'
```

Так само ви можете пропустити деяку частину аудіофайлу з самого початку, використовуючи параметр *offset*. Наприклад, якщо ви не хочете транскрибувати перші 4 секунди звуку, передайте 4 як значення для атрибута *offset*. Наприклад, наступний скрипт пропускає перші 4 секунди аудіофайлу, а потім транскрибує аудіофайл протягом 10 секунд.

```
sample_audio = speech_recog.AudioFile('E:/Datasets/my_audio.wav')
with sample_audio as audio_file:
    audio_content = recog.record(audio_file, offset=4, duration=10)

recog.recognize_google(audio_content)
```

Результат:

```
'take the winding path to reach the lake no closely the size of the gas tank
web degrees office dirty face'
```

Обробка шуму

Аудіо файл може містити шум з різних причин. Шум справді може вплинути на якість перекладу мови у текст. Щоб зменшити шум, клас *Recognizer* містить метод *adjust_for_ambient_noise*(), який приймає об’єкт *AudioData* як параметр. Наступний скрипт показує, як можна покращити якість транскрипції, видаливши шум з аудіофайлу:

```
sample_audio = speech_recog.AudioFile('E:/Datasets/my_audio.wav')
with sample_audio as audio_file:
     recog.adjust_for_ambient_noise(audio_file)
     audio_content = recog.record(audio_file)

recog.recognize_google(audio_content)
```

Результат:

```
'Bristol O2 left shoulder take the winding path to reach the lake no closely
the size of the gas tank web degrees office 30 face before you go out the
race was badly strained and hung them the stray cat gave birth to kittens
the younger again no clear response the mail was called before the bells
ring what weather is in living'
```

Висновок дуже схожий на те, що ми отримали раніше; це пов'язано з тим, що в аудіофайлі було мало шуму.

Розпізнавання мовлення з живого мікрофона

У цьому розділі ви побачите, як можна транслювати живе аудіо, отримане через мікрофон у вашій системі.

Існує кілька способів обробки аудіовходу, отриманого через мікрофон, і для цього було розроблено різні бібліотеки. Однією з таких бібліотек є *PyAudio*. Виконайте наступний скрипт для встановлення бібліотеки *PyAudio*:

```
pip install PyAudio
```

Тепер джерелом звуку, що транскрибується, є мікрофон. Щоб захопити звук із мікрофона, нам потрібно спочатку створити об’єкт класу *Microphone* модуля *Speach_Recogniton*, як показано нижче:

```
mic = speech_recog.Microphone()
```

Щоб побачити список всіх мікрофонів у системі, ви можете використовувати метод *list_microphone_names*():

```
speech_recog.Microphone.list_microphone_names()
```

Результат:

```
['Microsoft Sound Mapper - Input',
 'Microphone (Realtek High Defini',
 'Microsoft Sound Mapper - Output',
 'Speakers (Realtek High Definiti',
 'Microphone Array (Realtek HD Audio Mic input)',
 'Speakers (Realtek HD Audio output)',
 'Stereo Mix (Realtek HD Audio Stereo input)']
```

Це список мікрофонів, доступних у системі. Майте на увазі, що ваш список, швидше за все, виглядатиме інакше.

Наступним кроком є захоплення звуку з мікрофона. Для цього потрібно викликати метод *listen*() класу *Recognizer*(). Як і метод *record*(), метод *listen*() також повертає об'єкт *speech_recognition.AudioData*, який може бути переданий методу *recognize_google*().

Наступний скрипт пропонує користувачеві щось сказати у мікрофон, а потім друкує все, що сказав користувач:

```
with mic as audio_file:
    print("Speak Please")

    recog.adjust_for_ambient_noise(audio_file)
    audio = recog.listen(audio_file)

    print("Converting Speech to Text...")
    print("You said: " + recog.recognize_google(audio))
```

Як тільки ви виконаєте наведений вище скрипт, ви побачите таке повідомлення:

```
Speak Please
```

У цей момент промовте все, що хочете, і зробіть паузу. Як тільки ви зробили паузу, ви побачите транскрипцію всього, що ви сказали. Ось результат, який я отримав:

```
Converting Speech to Text...
You said: hello this is normally from stack abuse abuse this is an article
```

```
on speech recognition I hope you will like it and this is just a test speech
and when I will stop speaking are you in today thank you for Reading
```

Важливо, що якщо метод *recognize_google*() не може порівняти слова, які ви кажете, з будь-яким із слів у своєму сховищі, видається виняток. Ви можете перевірити це, сказавши кілька незрозумілих слів. Ви повинні побачити наступний виняток:

```
Speak Please
Converting Speech to Text...
---------------------------------------------------------------------------
UnknownValueError                         Traceback (most recent call last)
 in 
      8     print("Converting Speech to Text...")
      9
---> 10     print("You said: " + recog.recognize_google(audio))
     11
     12

~\Anaconda3\lib\site-packages\speech_recognition\__init__.py in
recognize_google(self, audio_data, key, language, show_all)
    856         # return results
    857         if show_all: return actual_result
--> 858         if not isinstance(actual_result, dict) or
len(actual_result.get("alternative", [])) == 0: raise UnknownValueError()
    859
    860         if "confidence" in actual_result["alternative"]:

UnknownValueError:
```

Найкращим підходом є використання блоку *try* при виклику методу *recognize_google*(), як показано нижче:

```
with mic as audio_file:
    print("Speak Please")

    recog.adjust_for_ambient_noise(audio_file)
    audio = recog.listen(audio_file)

    print("Converting Speech to Text...")

    try:
        print("You said: " + recog.recognize_google(audio))
    except Exception as e:
        print("Error: " + str(e))
```

7.4. Контрольні запитання

1. Наведіть приклад використання розпізнавання мови в повсякденному житті.

2. Чи можливо транскрибувати лиш певну частину файлу? Які функції відповідають за це?

3. Які *API* може використовувати клас *Recognizer* із модуля *speech_recognition*?

4. Які функції відповідно використовуються для запису голосу в живу та з аудіофайлу? Яким чином це вказується у програмі?

5. Спробуйте записати для програми набір різномовних слів. Який вивід ви отримали?

Лабораторна робота №8. Розпізнавання рукописних цифр

8.1. Мета роботи

Вивчити основи роботи з нейронними мережами для створення, навчання і тестування моделей мереж для розпізнавання рукописних цифр за допомогою мови програмування *Python*.

8.2. Теоретичні відомості

Щоб зробити машини більш інтелектуальними, розробники занурюються в технології машинного та глибокого навчання. Людина вчиться виконувати завдання, практикуючись і повторюючи його знову і знову, щоб запам'ятати, як виконувати завдання. Тоді нейрони в його мозку автоматично спрацьовують, і вони можуть швидко виконати завдання, якому навчилися. Глибоке навчання також дуже схоже на це. Використовуються різні типи архітектур нейронних мереж для різних типів проблем. Наприклад – розпізнавання об'єктів, класифікація зображень і звуків, виявлення об'єктів, сегментація зображення тощо.

Розпізнавання рукописних цифр — це здатність комп'ютерів розпізнавати цифри, написані людиною. Це важке завдання для машини, тому що рукописні цифри не є досконалими і можуть бути створені з різними відтінками. Розпізнавання рукописних цифр є рішенням цієї проблеми, яке використовує зображення цифри та розпізнає цифру, присутню на зображенні.

8.3. Порядок виконання роботи

1. Імпортуйте бібліотеки та завантажте набір даних

По-перше, ми збираємося імпортувати всі модулі, які нам знадобляться для навчання нашої моделі. Кожен з них вже раніше використовувався на попередніх лабораторних.

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import load_digits
from sklearn import metrics, neighbors
from sklearn.model_selection import train_test_split
%matplotlib inline
```

Завантажуємо дані. Бібліотека *sklearn* вже містить деякі набори даних, і *load_digits* є одним із них. Складається із зображень цифр розміром 8x8 пікселів.

Для початку виведемо атрибути даного набору та переглянемо вміст даних з яким можна працювати.

```
digits = load_digits()
digits.keys()
dict_keys(['data', 'target', 'frame', 'feature_names',
'target_names', 'images', 'DESCR'])
```

Як бачимо, маємо 7 атрибутів, а саме (відповідно) згладжена матриця даних, ціль класифікації, назви стовпців набору даних, назви цільових класів, масив 8x8 значень градацій сірого для кожного зображення, повний опис набору даних.

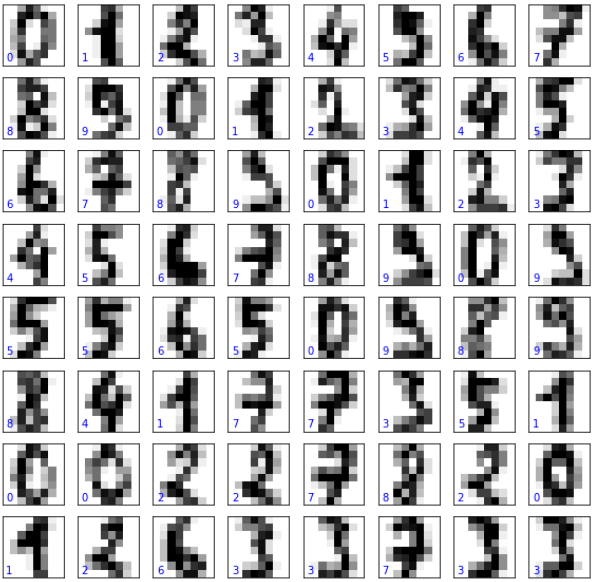
```
digits.target_names
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])

X = digits.data
y = digits.target

X.shape
(1797, 64)
```

Отже, цей набір даних складається з 1797 зображень 8x8. Щоб використати фігури 8x8, ми будемо використовувати цей вектор довжиною $8 \times 8 = 64$.

```
N, d = X.shape
```

Виведемо перші значення кожного рядка:

```
plt.figure(figsize = (10, 10)) # Розмір вікна у дюймах
for i in range(64):
    plt.subplot(8, 8, i + 1)
    plt.xticks([])
    plt.yticks([])
    plt.imshow(np.reshape(X[i, :], (8, 8)))
    # Виведемо значення зображення в нижньому кутку
клітинки (0, 7)
    plt.text(0, 7, str(digits.target[i]), color = 'b')
    plt.set_cmap('binary')
```



Також виведемо перше число і побачимо, що воно складається з масиву 1x64 значеннями від 0 до 15, які відображають колір зображення. Це і є причина зміни форми масиву функцією *np.reshape* до 8x8.

```
X[0,:]
```

```
array([ 0.,  0.,  5., 13.,  9.,  1.,  0.,  0.,  0.,  0.,
13., 15., 10.,
       15.,  5.,  0.,  0.,  3., 15.,  2.,  0., 11.,  8.,
0.,  0.,  4.,
       12.,  0.,  0.,  8.,  8.,  0.,  0.,  5.,  8.,  0.,
0.,  9.,  8.,
        0.,  0.,  4., 11.,  0.,  1., 12.,  7.,  0.,  0.,
2., 14.,  5.,
       10., 12.,  0.,  0.,  0.,  0.,  6., 13., 10.,  0.,
0.,  0.])
```

Метод *kNN*

KNeighborsClassifier може внутрішньо обчислювати дані найближчих сусідів, але їхнє попереднє обчислення може мати кілька переваг, наприклад точніший контроль параметрів, кешування для багаторазового використання або користувацькі реалізації.

Тут ми використовуємо властивість кешування *pipelines* для кешування графіка найближчих сусідів між декількома підгонками *KNeighborsClassifier*. Перший виклик повільний, оскільки він обчислює графік сусідів, а наступні виклики швидші, оскільки їм не потрібно повторно обчислювати графік. Перевага методу суттєвіша, якщо набір даних або сітка параметрів для пошуку досить великі.

```
X_train, X_test, y_train, y_test = train_test_split(X,
      y, test_size = 0.33,random_state = 7)
model = neighbors.KNeighborsClassifier(n_neighbors = 3)
model.fit(X_train, y_train)
y_test_pred = model.predict(X_test)

N_test = X_test.shape[0]
N_test
594
print(metrics.accuracy_score(y_test, y_test_pred))
print(np.mean(y_test != y_test_pred))
```

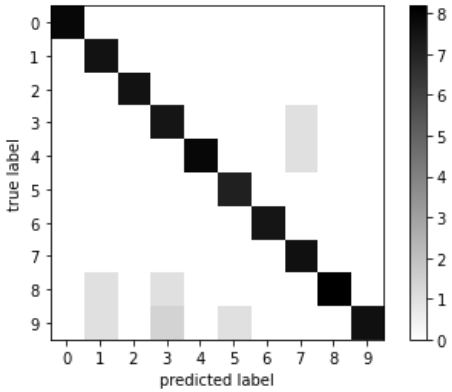
```
0.9865319865319865
0.013468013468013467
```

Матриця невідповідностей

Побудуємо матрицю для визначення справжніх значень цифр і прогнозованих значень цифр.

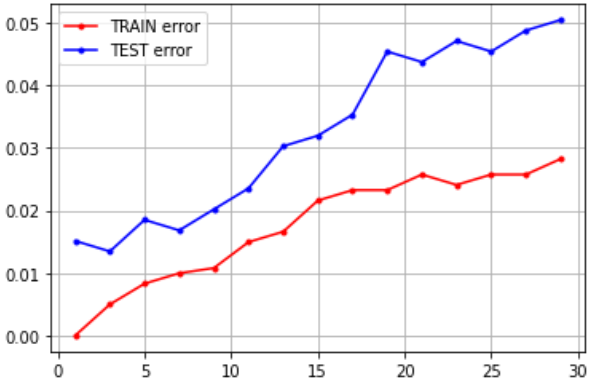
```
metrics.confusion_matrix(y_test, y_test_pred)
array([[63,  0,  0,  0,  0,  0,  0,  0,  0,  0],
       [ 0, 57,  0,  0,  0,  0,  0,  0,  0,  0],
       [ 0,  0, 57,  0,  0,  0,  0,  0,  0,  0],
       [ 0,  0,  0, 56,  0,  0,  0,  1,  0,  0],
       [ 0,  0,  0,  0, 63,  0,  0,  1,  0,  0],
       [ 0,  0,  0,  0,  0, 51,  0,  0,  0,  0],
       [ 0,  0,  0,  0,  0,  0, 56,  0,  0,  0],
       [ 0,  0,  0,  0,  0,  0,  0, 58,  0,  0],
       [ 0,  1,  0,  1,  0,  0,  0,  0, 67,  0],
       [ 0,  1,  0,  2,  0,  1,  0,  0,  0, 58]])
```

```
M = metrics.confusion_matrix(y_test, y_test_pred)
M = np.sqrt(M)
plt.imshow(M, interpolation = 'nearest')
plt.set_cmap('binary')
plt.grid(False)
plt.xticks(range(10))
plt.yticks(range(10))
plt.xlabel("predicted label")
plt.ylabel("true label")
plt.colorbar()
<matplotlib.colorbar.Colorbar at 0x7fd345729c90>
```

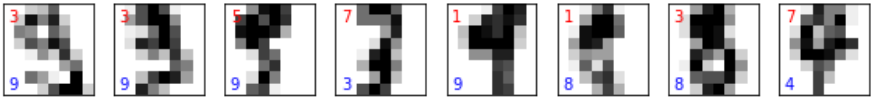



```
kk = range(1, 30, 2)
err_train = []
err_test = []
for k in kk:
    model = neighbors.KNeighborsClassifier(n_neighbors
= k)
    model.fit(X_train, y_train)
    err_train.append(np.mean(model.predict(X_train) !=
y_train))
    err_test.append(np.mean(model.predict(X_test) != y_
test))

plt.plot(kk, err_train, '.-r', label = 'Train error')
plt.plot(kk, err_test,  '.-
b', label = 'Test error')
plt.legend(loc = 2)
```

```
<matplotlib.legend.Legend at 0x7fd3456c6950>
```




```
print(min(err_test))
print(kk[err_test.index(min(err_test))])
0.013468013468013467
3

model = neighbors.KNeighborsClassifier(n_neighbors = 3)
model.fit(X_train, y_train)
i=1
plt.figure(figsize = (10, 10)) # Размір вікна в дюймах
i_subplot = 1
yi_test_pred =model.predict(X_test)
for i in range(N_test):
    if yi_test_pred[i]!= y_test[i]:
        plt.subplot(8, 8, i_subplot)
        i_subplot += 1
        plt.xticks([])
        plt.yticks([])
        plt.imshow(np.reshape(X_test[i, :], (8, 8)), cm
ap = plt.cm.binary)
        plt.text(0, 7, str(y_test[i]),    color = 'b')

        plt.text(0, 1, str(yi_test_pred[i]), color = 'r
')
```



Плюси та мінуси методу *kNN*

Плюси

- Простий метод
- Для низки завдань показує непогані результати (як, наприклад, завдання розпізнавання рукописних цифр)
- Достатньо стійкий до викидів (при підходящому виборі *k*)
- Працює як із числовими, так і номінальними ознаками

Мінуси

- Скільки брати сусідів?
- Яку метрику використати?
- Необхідно зберігати всю вибірку
- Занадто повільний (але можна використовувати прискорені алгоритми, наприклад, *kd*-дерева тощо)

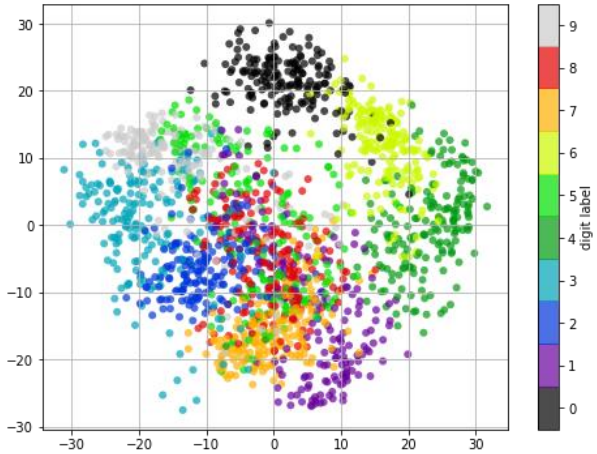
Методи зниження розмірності

PCA

Дозволяє проектувати дані з вихідного 64-вимірного простору в нижчий вимірний простір, в нашому випадку 2-вимірний; використовується для побудови даних і кластерів у цьому новому просторі.

```
from sklearn.decomposition import PCA
pca = PCA(n_components = 2)
pca.fit(X)
X_pca = pca.transform(X) # або відразу
pca.fit_transform(X)
X_pca.shape
(1797, 2)
```

```
plt.figure(figsize = (8, 6))
plt.scatter(X_pca[:, 0], X_pca[:, 1], marker="x",
c = digits.target,
            edgecolor = 'none', alpha = .7)
plt.set_cmap(plt.cm.get_cmap('nipy_spectral', 10))
plt.colorbar(label = 'digit label', ticks = range(10))
plt.clim(-0.5, 9.5)
```



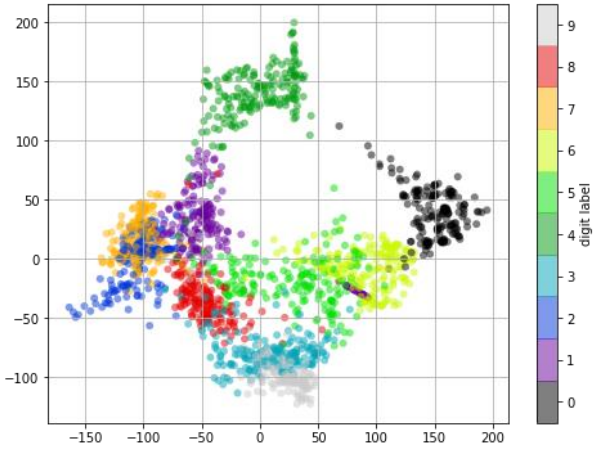
Isomap

Зменшує нелінійну розмірності за допомогою ізометричного відображення

```
from sklearn.manifold import Isomap

iso = Isomap(n_components = 2)
iso.fit(X)
X_iso = iso.transform(X)

X_iso.shape
```

```
(1797, 2)
```

```
plt.figure(figsize = (8, 6))
plt.scatter(X_iso[:, 0], X_iso[:, 1], c = digits.target
, edgecolor = 'none', alpha = 0.5)
plt.set_cmap(plt.cm.get_cmap('nipy_spectral', 10))
plt.colorbar(label = 'digit label', ticks = range(10))
plt.clim(-0.5, 9.5)
```



Випадкова проекція

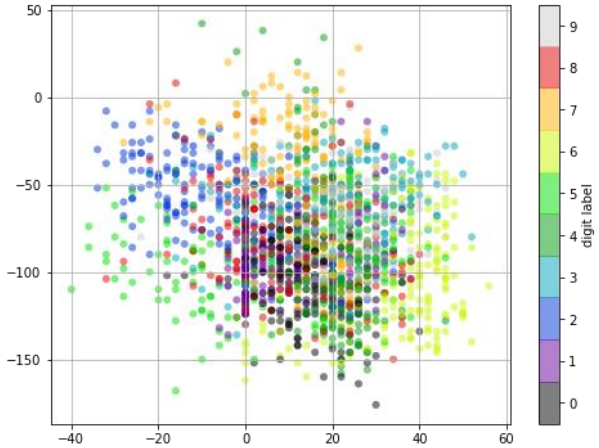
Випадкові прогнози — це простий і обчислювально ефективний спосіб зменшити розмірність даних шляхом обміну контрольованої величини точності (як додаткової дисперсії) на швидшу обробку та менші розміри моделі.

Розміри та розподіл матриць випадкових проекцій контролюються таким чином, щоб зберегти попарні відстані між будь-якими двома вибірками набору даних.

Основним теоретичним результатом ефективності випадкової проекції є лема Джонсона-Лінденштрауса.

```
from sklearn import random_projection
```

```
rp = random_projection.SparseRandomProjection(n_compone
nts = 2, random_state = 41)
rp.fit(X)
X_rp = rp.transform(X)

plt.figure(figsize = (8, 6))
plt.scatter(X_rp[:, 0], X_rp[:, 1], c = digits.target,
edgecolor = 'none', alpha = 0.5)
plt.set_cmap(plt.cm.get_cmap('nipy_spectral', 10))
plt.colorbar(label = 'digit label', ticks = range(10))
plt.clim(-0.5, 9.5)
```



LDA

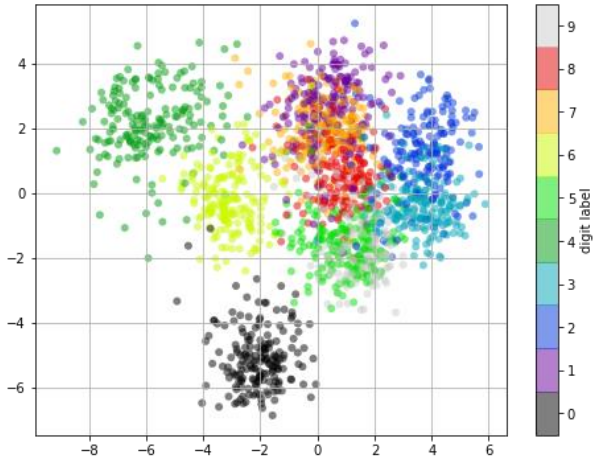
Класифікатор із лінійною межею прийняття рішень, створений підгонкою умовної щільності класу до даних і використанням правила Байєса.

Модель підбирає щільність Гауса для кожного класу, припускаючи, що всі класи мають однакову коваріаційну матрицю.

На відміну від інших методів *LinearDiscriminantAnalysis* використовує надані мітки, а тому є контрольованим методом

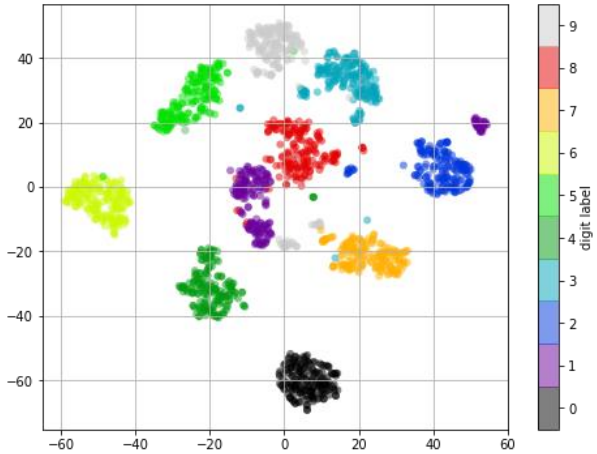
зменшення розмірності. Зменшення розмірності вхідних даних відбувається за допомогою проектування їх у найбільш дискримінаційних напрямках за допомогою методу трансформації.

```
from sklearn.discriminant_analysis import LinearDiscrim
inantAnalysis as LDA
mlda = LDA(n_components = 2)
mlda.fit(X, y)
X_lda = mlda.transform(X)

plt.figure(figsize = (8, 6))
plt.scatter(X_lda[:, 0], X_lda[:, 1], c = digits.target
, edgecolor = 'none', alpha = 0.5)
plt.set_cmap(plt.cm.get_cmap('nipy_spectral', 10))
plt.colorbar(label = 'digit label', ticks = range(10))
plt.clim(-0.5, 9.5)
```




```
# Візуалізуємо 4-мірні дані на площину
from sklearn.manifold import TSNE
# Визначаємо модель та швидкість навчання
model = TSNE(learning_rate=100)
# Навчаємо модель
transformed = model.fit_transform(X)
```

```
# Представляємо результат у двовимірних координатах
x_axis = transformed[:, 0]
y_axis = transformed[:, 1]
plt.figure(figsize = (8, 6))
plt.scatter(x_axis, y_axis, c = digits.target, edgecolo
r = 'none', alpha = 0.5)
plt.set_cmap(plt.cm.get_cmap('nipy_spectral', 10))
plt.colorbar(label = 'digit label', ticks = range(10))
plt.clim(-0.5, 9.5)
```



8.4. Контрольні запитання

1. Опишіть коротко суть методів пониження розмірностей.
2. Яким чином саме виводяться зображення рукописних чисел з набору даних *load_digits* з бібліотеки *sklearn*?
3. На прикладі методу *kNN* проведіть перевірку точності методів пониження розмірностей та порівняйте їх між собою.
4. Наведіть визначення терміну «розпізнавання рукописних цифр» та приклади застосування даної технології.
5. Вкажіть плюси та мінуси методу *kNN*.

Лабораторна робота №9. Самонавчання штучного інтелекту. Навчання з підкріпленням

9.1. Мета роботи

Розробити систему стабілізації перевернутого маятника на основі навчання з підкріпленням.

9.2. Теоретичні відомості

Що таке навчання з підкріпленням, або Reinforcement learning? Майже те ж саме, що й навчання з учителем, але в ролі "вчителя" виступає справжнє або віртуальне середовище. Наприклад: «робота» кидають в якийсь лабіринт, з якого він сам повинен знайти вихід. У процесі пошуку робот отримує від зовнішнього середовища інформацію про те, де виходу немає, таким чином вивчає навколишній світ і вчиться знаходити шлях до виходу.

Нагородою за успішно виконане завдання є можливість взятися за нове, а також набрані в процесі виконання бали. Чим ефективніше виконана задача, тим більше нараховується балів.

Навчання з підкріпленням застосовується у випадках коли потрібно вибрати кращий варіант серед багатьох або досягти складної мети за безліч ходів. Алгоритми підкріплення, які включають в себе глибоке навчання, можуть перемогти чемпіонів світу в грі Go, починаючи з базового розуміння правил гри і тренуючись від партії до партії.

Таким чином, це вже штучний інтелект в дії: машина намагається вирішити задачу різними способами, помиляється, вчиться на своїх помилках, покращує показники. Цей метод використовують у першу чергу там, де потрібно навчити машину виживати в реальному середовищі.

Навчання з підкріпленням: дві основні мети навчання.

Наприклад, на дорозі – у випадку з безпілотним автомобілем. У навчанні безпілотних автомобілів у машини немає завдання запам'ятати детальну карту міста, країни, континенту, всі вулиці

та повороти. Але вона обов'язково повинна зрозуміти шаблони повторюваних ситуацій і узагальнити їх.

Перша мета робота в навчанні з підкріпленням – мінімізувати помилки. Машина вчиться аналізувати інформацію перед кожним наступним ходом. Наприклад, безпілотний автомобіль під час навчання вчиться вчасно реагувати на сигнал світлофора, зупинитися перед пішоходом на переході, пропустити автомобіль, що швидко рухається або спецтранспорт справа. Щоб досягти кращого результату, машина навчається у віртуальній моделі міста з випадковими пішоходами та іншими учасниками дорожнього руху.

Друга мета робота в Reinforcement learning – отримати від виконання завдання максимальну вигоду. Сама вигода при цьому повинна бути запрограмована заздалегідь: максимально швидкий час проходження маршруту, оптимальне витрачання ресурсів підприємства, обслуговування якомога більшої кількості відвідувачів

Навчання з підкріпленням застосовується там, де потрібно порівняти відкладену вигоду – мету – з ситуативним прийняттям рішення. Цей вид навчання вирішує складне завдання співвіднесення негайних дій із відстроченою віддачею, яку вони продукують. Як і людям, алгоритмам підкріплення навчання іноді доводиться чекати, щоб побачити плоди своїх рішень. У таких випадках часто буває складно зрозуміти, яка дія призводить до якого результату.

Області практичного застосування reinforcement learning:

Постановка цілей

Планування

Системи сприйняття

Боти для комп'ютерних ігор

Трейдингові боти

Чат боти, які навчаються від діалогу до діалогу

Розглянемо декілька варіантів застосування навчання з підкріпленням на практиці. Один з найбільш інтуїтивно зрозумілих прикладів це використання навчання з підкріпленням для ігрових автоматів.

У нашій постановці завдання є *n* гральних автоматів, у кожному з яких фіксовано можливість виграшу. Тоді мета агента – знайти слот-машину з найбільшим очікуваним виграшем і завжди обирати саме її. Для простоти у нас буде лише чотири ігрові автомати, з яких потрібно буде вибирати.

Це завдання можна з натяжкою віднести до *reinforcement learning*, оскільки завданням з цього класу характерні такі властивості:

- **Різні дії призводять до різних виграшів**. Наприклад, при пошуку скарбів у лабіринті поворот ліворуч може означати купу діамантів, а поворот праворуч – яму отруйних змій.
- **Агент отримує виграш із затримкою у часі**. Це означає, що, повернувши ліворуч у лабіринті, ми не одразу зрозуміємо, що це правильний вибір.
- **Виграш залежить від стану системи**. Продовжуючи приклад вище, поворот ліворуч може бути правильним у поточній частині лабіринту, але не обов'язково в інших.

У задачі про ігрові автомати немає ні другої, ні третьої умови, що суттєво її спрощує і дозволяє нам сконцентруватися лише на виявленні оптимальної дії з усіх можливих варіантів. На мові *reinforcement learning* означає знайти «правило поведінки» (*policy*). Ми будемо використовувати метод, званий *policy gradients*, при якому нейромережа оновлює своє правило поведінки наступним чином: агент здійснює дію, отримує зворотний зв'язок від середовища та на її основі коригує ваги моделі через градієнтний спуск.

У сфері навчання з підкріпленням є й інший підхід, у якому агенти навчають *value functions*. Замість того, щоб знаходити оптимальну дію в поточному стані, агент вчиться передбачати, наскільки вигідно перебувати в даному стані та здійснювати дану дію. Обидва підходи дають хороші результати, проте логіка *policy gradient* очевидніша.

Policy Gradient

Як ми з'ясували, у разі очікуваний виграш кожного з ігрових автоматів залежить від поточного стану середовища. Виходить, що наша нейромережа складатиметься лише з набору терезів, кожна з яких відповідає одному ігровому автомату. Ці ваги і визначатимуть, за яку ручку треба смикнути, щоб отримати максимальний виграш. Наприклад, якщо всі ваги ініціалізувати рівними 1, агент буде однаково оптимістичний з приводу виграшу у всіх ігрових автоматах.

Для оновлення ваги моделі ми будемо використовувати *e*-подібну лінію поведінки. Це означає, що в більшості випадків агент вибиратиме дію, що максимізує очікуваний виграш, проте іноді (з ймовірністю рівною *e*) дія буде випадковою. Так буде забезпечено вибір всіх можливих варіантів, що дозволить нейромережі «дізнатися» більше про кожну із них.

Здійснивши одну з дій, агент отримує зворотний зв'язок від системи: 1 або -1 залежно від того, чи він виграв. Це значення потім використовується для розрахунку функції втрат:

$$Loss = -log(\text{п}) * A$$

де A (*advantage*) – важливий елемент усіх алгоритмів навчання із підкріпленням. Він показує, наскільки досконала дія краща, ніж якийсь *baseline*. Надалі ми будемо використовувати складніший *baseline*, а поки приймемо його рівним 0, тобто *A* буде просто дорівнює нагороді за кожну дію (1 або -1); п – це правило поведінки, вага нейромережі, що відповідає ручці слот-машини, яку ми вибрали на поточному етапі.

Інтуїтивно зрозуміло, що функція втрат повинна набувати таких значень, щоб ваги дій, що призвели до виграшу збільшувалися, а ті, що призвели до програшу, зменшувалися. В результаті ваги будуть оновлюватися, а агент все частіше і частіше вибиратиме ігровий автомат з найбільшою фіксованою ймовірністю виграшу, поки, нарешті, він не вибиратиме його завжди.

Вирішення повноцінного завдання навчання з підкріпленням

Тепер, коли ми знаємо, як створити агента, здатного вибирати оптимальне рішення з кількох можливих, перейдемо до розгляду складнішого завдання, яке і буде прикладом повноцінного *reinforcement learning*: оцінюючи поточний стан системи, агент повинен вибирати дії, які максимізують виграш не тільки зараз, але й у майбутньому.

Системи, в яких може бути вирішена навчання з підкріпленням, називаються Марківськими процесами прийняття рішень (*Markov Decision Processes, MDP*). Для таких систем характерні виграші та дії, що забезпечують перехід з одного стану до іншого, причому ці виграші залежать від поточного стану системи та рішення, яке приймає агент у цьому стані. Виграш може бути отриманий із затримкою у часі.

Формально Марківський процес прийняття рішень можна визначити так. *MDP* складається з набору всіх можливих станів S і дій A , причому в кожний момент часу він знаходиться в стані s і здійснює дію з цих наборів. Таким чином, дано кортеж (s, a) і для нього визначено $T(s, a)$ — можливість переходу в новий стан s' і $R(s, a)$ — виграш. У результаті в будь-який момент часу в *MDP* агент знаходиться в стані s , приймає рішення a і у відповідь отримує новий стан s' і виграш r.

Наприклад, навіть процес відчинення дверей можна розглядати у вигляді Марківського процесу прийняття рішень. Станом буде наш погляд на двері, а також розташування нашого тіла та двері у світі. Всі можливі рухи тіла, що ми можемо зробити, є набором A , а виграш – це успішне відкриття дверей. Певні дії (наприклад, крок у бік дверей) наближають нас до досягнення мети, проте власними силами не приносять виграшу, оскільки його забезпечує лише відкривання дверей. У результаті агент повинен здійснювати такі дії, які рано чи пізно призведуть до вирішення завдання.

Задача стабілізації перевернутого маятника

Скористаємося *OpenAI Gym* – платформою для розробки та тренування *AI* ботів за допомогою ігор та алгоритмічних випробувань та візьмемо класичне завдання звідти: завдання стабілізації перевернутого маятника або *Cart-Pole*. У нашому випадку суть завдання полягає в тому, щоб якомога довше утримувати стрижень у вертикальному положенні, рухаючи візок по горизонталі:

На відміну від задачі про багаторукий бандит, в даній системі є:

- **Спостереження**. Агент повинен знати, де стрижень знаходиться зараз і під яким кутом. Це спостереження нейромережа використовуватиме оцінки ймовірності тієї чи іншої дії.
- **Відстрочений виграш**. Необхідно рухати візок таким чином, щоб це було вигідно як зараз, так і в майбутньому. Для цього зіставлятимемо пару «спостереження — дія» зі скоригованим значенням виграшу. Коригування здійснюється функцією, яка зважує дії за часом.

Щоб врахувати затримку виграшу у часі, нам потрібно використовувати *policy gradient* метод із деякими поправками.

По-перше, тепер необхідно оновлювати агента, який має більше одного спостереження за одиницю часу. Для цього всі спостереження ми збиратимемо в буфер, а потім використовувати їх одночасно, щоб оновити ваги моделі. Цей набір спостережень за одиницю часу зіставляється з дисконтованим виграшем.

Таким чином, кожна дія агента буде здійснена з урахуванням не тільки миттєвого виграшу, а й усіх наступних. Також тепер ми будемо використовувати скоригований виграш як оцінку елемента *A* (*advantage*) у функції втрат.

15.3. Порядок виконання роботи

Спершу ми створимо наших пристроїв (ігровий апарат, «однорукий бандит» тощо). У прикладі їх буде 4. Функція *pullBandit* генерує випадкове число зі стандартного нормального розподілу, а потім порівнює його зі значенням пристрою і повертає результат гри. Що далі за списком перебуває пристрій, то більша ймовірність, що агент виграє, обравши саме його. Таким чином ми хочемо, щоб наш агент навчився завжди вибирати останнього пристрою.

```
import tensorflow as tf
import numpy as np

#Список наших бандитів. Наразі бандит 4 (індекс №3) налаштований на те, що
найчастіше дає позитивну винагороду.
bandits = [0.2,0,-0.2,-5]
num_bandits = len(bandits)
def pullBandit(bandit):
    result = np.random.randn(1)
    if result > bandit:
        #повертає позитивну винагороду
        return 1
    else:
        #повертає негативну винагороду
        return -1
```

Агент. Частина коду нижче створює наш простий агент, який

складається з набору значень для бандитів. Кожне значення відповідає виграшу/програшу, залежно від вибору того чи іншого бандита. Щоб оновлювати ваги агента, ми використовуємо *policy gradient*, тобто вибираємо дії, що мінімізують функцію втрат:

```
tf.reset_default_graph()

# Ці два рядки створюють feed-forward частину нейромережі. Тут і
відбувається вибір дії.
weights = tf.Variable(tf.ones([num_bandits]))
chosen_action = tf.argmax(weights,0)

# Наступні 6 рядків встановлюють процедуру навчання. Нейросеть приймає на
вхід дію та її результат, щоб оцінити функцію втрат і оновити ваги мережі.
reward_holder = tf.placeholder(shape=[1],dtype=tf.float32)
action_holder = tf.placeholder(shape=[1],dtype=tf.int32)
responsible_weight = tf.slice(weights,action_holder,[1])
loss = -(tf.log(responsible_weight)*reward_holder)
optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.001)
update = optimizer.minimize(loss)
```

Навчання агента. Ми будемо навчати агента шляхом вибору певних дій та отримання виграшів/програшів. Використовуючи отримані значення, ми знатимемо, як саме оновити ваги моделі, щоб частіше вибирати бандитів із великим очікуваним виграшем:

```
total_episodes = 1000 # Кількість ітерацій навчання
total_reward = np.zeros(num_bandits) # Початковий виграш усіх бандитів
дорівнює 0
e = 0.1 # Ймовірність випадкового вибору
init = tf.global_variables_initializer()

# Запускаємо граф tensorflow
with tf.Session() as sess:
    sess.run(init)
    i = 0
    while i < total_episodes:

        # Вибираємо дію або випадково або на основі нейромережі
        if np.random.rand(1) < e:
            action = np.random.randint(num_bandits)
```

```
        else:
            action = sess.run(chosen_action)
        # Отримуємо результат гри, обравши одного з бандитів
        reward = pullBandit(bandits[action])

        # Оновлюємо ваги
        _,resp,ww = sess.run([update,responsible_weight,weights],

feed_dict={reward_holder:[reward],action_holder:[action]})

        # Оновлюємо загальний виграш кожного бандита
        total_reward[action] += reward
        if i % 50 == 0:
            print("Нагорода за те, що працює" + str(num_bandits) +
            " бандити: " + str(total_reward))
        i+=1
print("Агент думає, що бандит №" + str(np.argmax(ww)+1) + " є найбільш
перспективним...")
if np.argmax(ww) == np.argmax(-np.array(bandits)):
    print("...і він правий!")
else:
    print("...але він помилився!")
```

Результат:

```
Нагорода за те, що працює 4 бандити: [ -1.   0.   0.  0.]
Нагорода за те, що працює 4 бандити: [ -1.  -1.   0.  45.]
Нагорода за те, що працює 4 бандити: [ -2.  -1.   1.  91.]
Нагорода за те, що працює 4 бандити: [ -3.  -1.   1.  138.]
Нагорода за те, що працює 4 бандити: [ -2.   0.   2.  183.]
Нагорода за те, що працює 4 бандити: [  0.   0.   2.  229.]
Нагорода за те, що працює 4 бандити: [ -1.   0.   3.  277.]
Нагорода за те, що працює 4 бандити: [ -1.  -1.   4.  323.]
Нагорода за те, що працює 4 бандити: [ -2.  -1.   2.  370.]
Нагорода за те, що працює 4 бандити: [ -2.  -2.   3.  416.]
Нагорода за те, що працює 4 бандити: [ -2.  -2.   3.  464.]
Нагорода за те, що працює 4 бандити: [ -3.  -1.   3.  508.]
Нагорода за те, що працює 4 бандити: [ -7.   1.   4.  549.]
Нагорода за те, що працює 4 бандити: [ -9.   0.   3.  593.]
Нагорода за те, що працює 4 бандити: [ -9.   0.   4.  640.]
Нагорода за те, що працює 4 бандити: [ -9.   0.   5.  687.]
Нагорода за те, що працює 4 бандити: [ -9.  -2.   5.  735.]
Нагорода за те, що працює 4 бандити: [ -9.  -4.   7.  781.]
Нагорода за те, що працює 4 бандити: [ -10. -4.   8.  829.]
Нагорода за те, що працює 4 бандити: [ -13. -4.   7.  875.]
Агент думає, що бандит №4 є найбільш перспективним...
... і він правий!
```

Вирішення повноцінного завдання навчання з підкріпленням

Імпортуємо бібліотеки та завантажимо середовище завдання *Cart-Pole*:

```
import tensorflow as tf
import tensorflow.contrib.slim as slim
import numpy as np
import gym
import matplotlib.pyplot as plt
%matplotlib inline

try:
    xrange = xrange
except:
    xrange = range
```

```
env = gym.make('CartPole-v0') # завантажуємо середовище завдання
```

Агент. Спочатку створимо функцію, яка дисконтуватиме всі наступні виграші на поточний момент:

```
gamma = 0.99 # коефіцієнт дисконтування

def discount_rewards(r):
    """ приймає одновимірний float масив винагород і обчислює знижену
винагороду """
    discounted_r = np.zeros_like(r)
    running_add = 0
    for t in reversed(xrange(0, r.size)):
        running_add = running_add * gamma + r[t]
        discounted_r[t] = running_add
    return discounted_r
```

Тепер створимо нашого агента:

```
class agent():
    def __init__(self, lr, s_size,a_size,h_size):
        # Нижче ініціалізована feed-forward частина нейромережі.
        # Агент оцінює стан середовища та здійснює дію
        self.state_in= tf.placeholder(shape=[None,s_size],dtype=tf.float32)
        hidden = slim.fully_connected(self.state_in,h_size,
                      biases_initializer=None,activation_fn=tf.nn.relu)
        self.output = slim.fully_connected(hidden,a_size,
                      activation_fn=tf.nn.softmax,biases_initializer=None)
        self.chosen_action = tf.argmax(self.output,1) # вибір дії

        # Наступні 6 рядків встановлюють процедуру навчання.
        # Нейросеть приймає на вхід вибрану дію
        # та відповідний виграш,
        # щоб оцінити функцію втрат та оновити ваги моделі.
        self.reward_holder = tf.placeholder(shape=[None],dtype=tf.float32)
        self.action_holder = tf.placeholder(shape=[None],dtype=tf.int32)

        self.indexes = tf.range(0,
        tf.shape(self.output)[0])*tf.shape(self.output)[1] +
self.action_holder

        self.responsible_outputs = tf.gather(tf.reshape(self.output, [-1]),
        self.indexes)
        # функція втрат
        self.loss = -tf.reduce_mean(tf.log(self.responsible_outputs)*
        self.reward_holder)

        tvars = tf.trainable_variables()
        self.gradient_holders = []
        for idx,var in enumerate(tvars):
            placeholder = tf.placeholder(tf.float32,name=str(idx)+'_holder')
            self.gradient_holders.append(placeholder)

        self.gradients = tf.gradients(self.loss,tvars)

        optimizer = tf.train.AdamOptimizer(learning_rate=lr)
        self.update_batch =
optimizer.apply_gradients(zip(self.gradient_holders,
        tvars))
```

Навчання агента. Тепер, нарешті, перейдемо до навчання агента:

```
tf.reset_default_graph() # Очищаємо граф tensorflow

myAgent = agent(lr=1e-2,s_size=4,a_size=2,h_size=8) # Ініціалізуємо агента
```

```
total_episodes = 5000 # Кількість ітерацій навчання
max_ep = 999
update_frequency = 5

init = tf.global_variables_initializer()

# Запуск графа tensorflow
with tf.Session() as sess:
    sess.run(init)
    i = 0
    total_reward = []
    total_lenght = []

    gradBuffer = sess.run(tf.trainable_variables())
    for ix,grad in enumerate(gradBuffer):
        gradBuffer[ix] = grad * 0

    while i < total_episodes:
        s = env.reset()
        running_reward = 0
        ep_history = []
        for j in range(max_ep):
            # Вибрати дію на основі ймовірностей, оцінених нейромережею
            a_dist =
sess.run(myAgent.output,feed_dict={myAgent.state_in:[s]})
            a = np.random.choice(a_dist[0],p=a_dist[0])
            a = np.argmax(a_dist == a)

            s1,r,d,_ = env.step(a) # Отримати нагороду за досконалу дію
            ep_history.append([s,a,r,s1])
            s = s1
            running_reward += r
            if d == True:
                # Оновити нейромережу
                ep_history = np.array(ep_history)
                ep_history[:,2] = discount_rewards(ep_history[:,2])
                feed_dict = {myAgent.reward_holder:ep_history[:,2],
                             myAgent.action_holder:ep_history[:,1],
                             myAgent.state_in:np.vstack(ep_history[:,0])}
                grads = sess.run(myAgent.gradients, feed_dict=feed_dict)
                for idx,grad in enumerate(grads):
                    gradBuffer[idx] += grad

                if i % update_frequency == 0 and i != 0:
                    feed_dict = dictionary =
dict(zip(myAgent.gradient_holders,
                                     gradBuffer))
                    _ = sess.run(myAgent.update_batch, feed_dict=feed_dict)
                    for ix,grad in enumerate(gradBuffer):
                        gradBuffer[ix] = grad * 0

                total_reward.append(running_reward)
```

```
                total_lenght.append(j)
                break


            # Оновити загальний виграш
        if i % 100 == 0:
            print(np.mean(total_reward[-100:]))
        i += 1
```

Результат:

```
16.0
21.47
25.57
38.03
43.59
53.05
67.38
90.44
120.19
131.75
162.65
156.48
168.18
181.43
```

9.4. Контрольні запитання

1. Опишіть модель ігрового автомату.
2. Яка стратегія була використана в даній моделі для рішення задачі розподілу обмеженої множини ресурсів між конкуруючими альтернативами?
3. Чи можлива задача без початкових знань про дані?
4. Які обов’язкові умови характерні для задач навчання з підкріпленням?
5. Що таке Марківські процеси прийняття рішень?

Лабораторна робота №10. Розпізнавання зображень на Python за допомогою TensorFlow та Keras

10.1. Мета роботи

Навчитися розроблювати програму на мові *Python* для розпізнавання зображень на базі *TensorFlow* та *Keras*.

10.2. Теоретичні відомості

TensorFlow

Бібліотека з відкритим кодом, створена для *Python* командою *Google Brain*. *TensorFlow* компілює безліч різних алгоритмів та моделей, дозволяючи користувачеві реалізувати глибокі нейронні мережі для використання у таких завданнях, як розпізнавання та класифікація зображень, а також обробка природної мови. *TensorFlow* – це потужний фреймворк, який функціонує шляхом реалізації низки вузлів(«граф») обробки, кожен із яких представляє математичну операцію.

Keras

Високорівневий *API* (інтерфейс прикладного програмування), який може використовувати функції *TensorFlow* (також інші бібліотеки *ML*, такі, як *Theano*). *Keras* був розроблений із зручністю та модульністю як керівні принципи. З практичної точки зору *Keras* дозволяє реалізувати безліч потужних, але найчастіше складних функцій *TensorFlow* максимально просто, до того ж він налаштований для роботи з *Python* без серйозних змін або налаштувань.

Класифікація розпізнавання зображень

Розпізнавання зображення відноситься до завдання введення зображення в нейронну мережу та присвоєння будь-якої мітки для цього зображення. Мітка, яку виводить мережа, буде відповідати заздалегідь визначеному класу. Може бути присвоєно як декілька класів, так і лише один. Якщо є лише

один клас, зазвичай застосовується термін «розпізнавання», тоді як завдання розпізнавання кількох класів часто називається «класифікацією».

Підмножина класифікацій зображень є вже визначенням об'єктів, коли певні екземпляри об'єктів ідентифікуються як такі, що належать до певного класу, наприклад, тварини, автомобілі або люди.

Яскравим прикладом такої класифікації є рішення найпоширенішої капчі – *ReCaptcha v2* від *Google*, де з набору картинок необхідно вибрати лише ті, що належать до вказаного в описі класу.

Функція вилучення

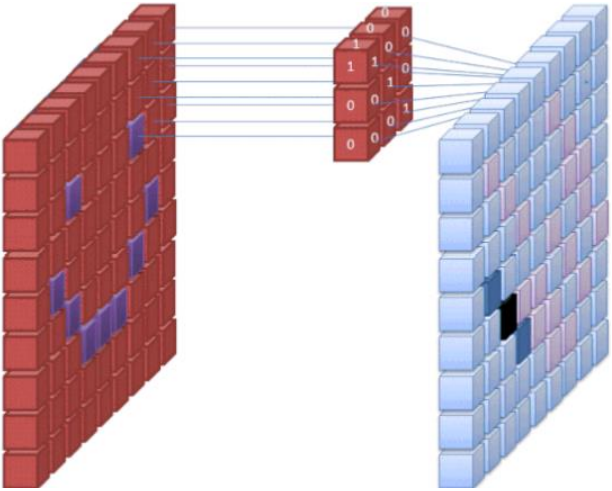
Щоб виконати розпізнавання або класифікацію зображень, нейронна мережа повинна витягти ознаки – елементи даних, які репрезентують максимальний інтерес та будуть передані по нейромережі. У конкретному випадку розпізнавання зображень такими ознаками є групи пікселів, такі як лінії та точки, які мережа аналізуватиме на наявність патерну.

Розпізнавання ознак (або вилучення ознак) – це процес отримання відповідних ознак з вхідного зображення, щоб їх можна було проаналізувати. Багато зображень містять анотації або метадані, які допомагають нейромережі знаходити відповідні ознаки.

Як нейронні мережі вчаться розпізнавати зображення

Розуміння про те, як нейронна мережа розпізнає зображення, допоможе вам при реалізації моделі нейронної мережі, тому коротко розглянемо процес розпізнавання зображень у наступних розділах.

Вилучення ознак за допомогою фільтрів



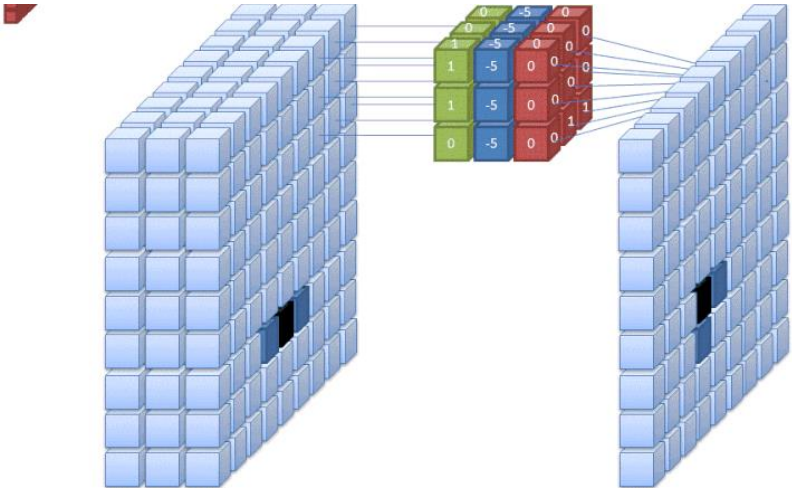
Перший шар нейронної мережі приймає всі пікселі зображення. Після того, як усі дані введені в мережу, до зображення застосовують різні фільтри, які формують розуміння різних частин зображення. Це витяг ознак, що створює «карти ознак».

Цей процес отримання ознак із зображення виконується за допомогою «згорткового шару», і згортка просто формує уявлення частини зображення. Саме з цієї концепції згортки ми отримуємо термін «Згорткова нейронна мережа» (*Convolutional Neural Network, CNN*) – тип нейронної мережі, що найчастіше використовується в класифікації та розпізнаванні зображень.

Якщо ви хочете представити, як саме працює створення карт ознак, уявіть собі процес піднесення ліхтарика до зображення у темній кімнаті. Коли ви ковзаєте променем по картинці, ви дізнаєтеся про особливості зображення. Фільтр це те, що мережа використовує для формування уявлення про зображення, і в цій метафорі світло від ліхтарика є фільтром.

Ширина променя ліхтарика визначає розмір фрагмента зображення, який ви переглядаєте за один раз, і нейронні мережі мають аналогічний параметр – розмір фільтра. Розмір фільтра впливає на те, скільки пікселів перевіряється за один раз. Загальний розмір фільтра, використовуваного CNN, дорівнює 3,

і він охоплює як висоту, так і ширину, тому фільтр перевіряє область пікселів 3 х 3.



У той час як розмір фільтра покриває висоту та ширину фільтра, глибина фільтра також має бути вказана.

Але як 2D зображення може мати глибину?

Справа в тому, що цифрові зображення відображаються у вигляді висоти, ширини і деякого значення RGB, яке визначає колір пікселя, тому «глибина», що відстежується, – це кількість колірних каналів, які має зображення. Зображення в градаціях сірого (не кольорові) мають лише 1 кольоровий канал, тоді як кольорові зображення мають глибину 3 канали.

Все це означає, що для фільтра розміром 3, застосованого до повнокольорового зображення, підсумкові розміри цього фільтра будуть 3x3x3. Для кожного пікселя, охоплюваного цим фільтром, мережа помножує значення фільтра на значення самих пікселів, щоб отримати числове подання цього пікселя . Потім цей процес виконується для всього зображення, щоб отримати повне уявлення. Фільтр переміщається по решті зображення відповідно до параметра, званого «крок», який визначає, на скільки пікселів повинен бути переміщений фільтр після того, як він обчислить значення у своїй поточній позиції.

Типовий розмір кроку для CNN - 2.

Кінцевим результатом цих розрахунків є карта ознак. Цей процес зазвичай виконується з кількома фільтрами, які допомагають зберегти складність зображення.

Функції активації

Після того, як карта ознак зображення була створена, значення, що представляють зображення, передаються через активацію або шар активації. Функція активації набуває цих значень, які завдяки згортковому шару знаходяться в лінійній формі (тобто просто список чисел) і збільшує їх нелінійність, оскільки самі зображення є нелінійними.

Типовою функцією активації, що використовується для досягнення цієї мети, є випрямлена лінійна одиниця (ReLU), хоча є деякі інші функції активації, які також іноді використовуються.

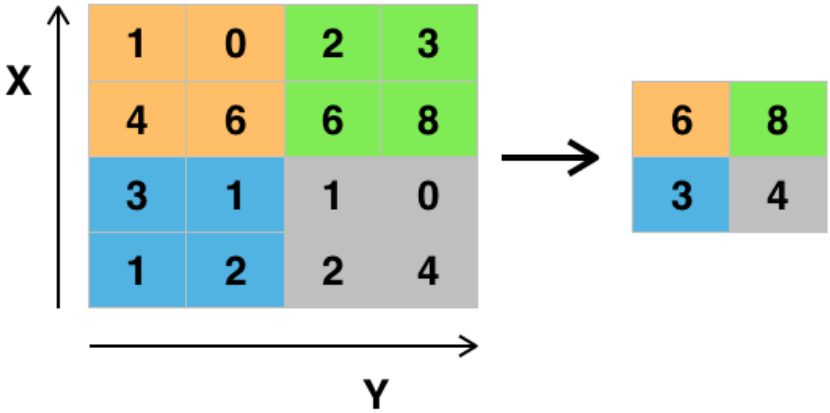
Об'єднання шарів

Після активації дані відправляються через шар, що об'єднує. Об'єднання «спрощує» зображення: бере інформацію, яка представляє зображення, і стискає її. Процес об'єднання в пул робить мережу гнучкішою і здатною краще розпізнавати об'єкти та зображення на основі відповідних функцій.

Коли ми дивимося на зображення, нас зазвичай хвилює не вся інформація (наприклад, що на задньому плані зображення), а лише ознаки, які нас цікавлять – люди, тварини тощо.

Аналогічно, об'єднуючий шар *CNN* позбавиться від непотрібних частин зображення, залишивши тільки ті частини, які він вважає релевантними, в залежності від заданого розміру об'єднуючого шару.

Оскільки мережа повинна приймати рішення щодо найбільш важливих частин зображення, розрахунок йде на те, що вона вивчить тільки ті частини зображення, які дійсно є суть об'єкта, що розглядається. Це допомагає запобігти «перенавчання» – коли мережа надто добре вивчає всі аспекти навчального прикладу і вже не може узагальнювати нові дані, оскільки враховує нерелевантні відмінності.



Існують різні способи поєднання значень, але найчастіше використовується максимальне об’єднання, тобто взяття максимального значення серед пікселів у межах одного фільтра (фрагмента зображення). Це відсіює 75% інформації за умови використання фільтра розміром 2x2.

Максимальні значення пікселів використовуються для врахування можливих спотворень зображення, а кількість параметрів (розмір зображення) зменшено, щоб контролювати перенавчання. Існують інші принципи об’єднання, такі як середнє чи сумарне об’єднання, але вони використовуються не так часто, оскільки максимальне об'єднання дає більшу точність.

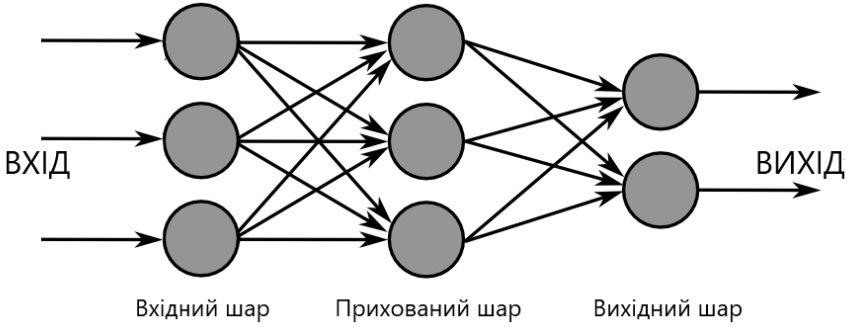
Стиснення

Останні шари нашої CNN, щільно пов'язані шари, вимагають, щоб дані були представлені у формі вектора для подальшої обробки. З цієї причини дані необхідно «звести докупи». Для цього значення стискуються у довгий вектор або стовпець послідовно впорядкованих чисел.

Цілком пов’язаний шар

Кінцеві шари CNN є щільно пов’язані шари або штучну нейронну мережу (*Artificial neural networks* (*ANN*)). Основною функцією *ANN* є аналіз вхідних ознак та об'єднання їх у різні атрибути, які допоможуть у класифікації. Ці шари утворюють набори нейронів, які представляють різні частини об'єкта, що розглядається, а набір нейронів може являти собою, наприклад, висячі вуха собаки або почервоніння яблука. Коли достатня

кількість цих нейронів активується у відповідь вхідне зображення, воно буде класифіковано як об'єкт.



Помилка або різниця між розрахованими значеннями та очікуваним значенням у навчальному наборі розраховується за допомогою *ANN*. Потім мережа піддається методу зворотного розповсюдження помилки, де розраховується вплив даного нейрона на нейрон у наступному шарі, а потім його вплив (вага) коригується. Це зроблено для оптимізації продуктивності моделі. Цей процес повторюється знову і знову, так мережа навчається на даних та вивчає зв'язки між вхідними ознаками та вихідними класами.

Нейрони в середніх пов’язаних шарах виводитимуть двійкові значення, які стосуються можливих класів. Якщо у вас є чотири різних класи (скажімо, собака, машина, будинок і людина), нейрон матиме значення «1» для класу, який, як він вважає, представляє зображення, і значення «0» для інших класів.

Кінцевий повністю пов'язаний шар, отримавши вихідні дані попереднього шару, надає можливість кожному з класів у межах одиниці (у сукупності). Якщо категорії «собака» надано значення 0,75 – ймовірність того, що зображення є собакою.

Робочий процес машинного навчання

Перш ніж ми перейдемо до прикладу навчання класифікатора зображень, давайте приділимо трохи часу розумінню робочого процесу або «конвеєра» машинного навчання. Процес навчання моделі нейронної мережі є досить стандартним і може бути поділений на чотири різні етапи.

Класифікатор зображень тепер навчений і зображення можуть бути передані CNN, яка тепер виведе припущення про зміст цього зображення.

Створення моделі

Створення моделі нейронної мережі включає вибір різних параметрів та гіперпараметрів. Ви повинні прийняти рішення про кількість шарів, що використовуються у вашій моделі, про те, яким буде розмір вхідних та вихідних шарів, які функції активації ви використовуватимете, чи використовуватимете виняток (*Dropout*) і т.д.

Навчання моделі

Після того, як ваша модель створена, вам просто залишається створити екземпляр моделі та підігнати його до своїх даних для навчання. Найбільша увага під час навчання моделі приділяється кількості необхідного на навчання часу. Ви можете вказати тривалість навчання мережі, задавши кількість епох навчання. Чим довше ви тренуєте модель, тим вища її ефективність, але якщо використовувати занадто багато епох навчання – ви ризикуєте перенавчити модель.

Оцінка моделі

Існує кілька кроків для оцінки моделі. Першим кроком є порівняння продуктивності моделі з набором перевірочних даних: тих даних, у яких модель була навчена. Таким чином, ви перевірите роботу моделі з цим новим набором даних та проаналізуєте її ефективність за допомогою різних показників.

Існують різні метрики для визначення продуктивності моделі нейронної мережі, але найпоширенішою є «точність», тобто кількість правильно класифікованих зображень, поділена на загальну кількість зображень у вашому наборі даних.

Після того, як ви побачите точність моделі в перевірочному наборі даних, ви, ймовірно, знову повернетеся і до-навчите мережу, використовуючи злегка підправлені параметри, оскільки навряд чи будете задоволені ефективністю своєї мережі при першому тренуванні. Ви будете продовжувати налаштовувати параметри своєї мережі, повторно навчати її та вимірювати ефективність, доки не будете задоволені точністю

мережі.

10.3. Порядок виконання роботи

Розпізнавання зображень із CNN

Спочатку нам знадобиться набір даних для навчання. У цьому прикладі ми будемо використовувати відомий набір даних *CIFAR*-10. *CIFAR*-10 – це великий набір даних, що містить понад 60 000 зображень, що представляють 10 різних класів об'єктів, таких як кішки, літаки та автомобілі.

Зображення є повнокольоровими RGB, але вони досить малі, всього 32 x 32. Відмінною особливістю набору даних *CIFAR*-10 є те, що він поставляється в комплекті з *Keras*, тому завантажити набір даних дуже просто, а самі зображення потребують лише мінімальної попередньої обробки. Почнемо з імпорту необхідних бібліотек.

```
import numpy
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten, BatchNormalization,
Activation
from keras.layers.convolutional import Conv2D, MaxPooling2D
from keras.constraints import maxnorm
from keras.utils import np_utils
```

Ми збираємося використовувати випадковий SEED (симетричний блоковий криптоалгоритм на основі Мережі Фейстеля), щоб результати, отримані в цій статті, могли бути відтворені вами:

```
# Set random seed for purposes of reproducibility
seed = 21
```

Підготовка данних

Тепер нам потрібно зробити ще один імпорт: сам набір даних.

```
# loading in the data
(X_train, y_train), (X_test, y_test) = cifar10.load_data()
```

Однією з дій, які ми хочемо зробити, буде нормалізація вхідних даних. Якщо значення вхідних даних знаходяться в дуже широкому діапазоні, це може негативно вплинути на

роботу мережі. У нашому випадку вхідними значеннями є пікселі у зображенні, які мають значення від 0 до 255.

Таким чином, щоб нормалізувати дані, треба розділити значення зображення на 255. Для цього нам спочатку потрібно перевести дані у формат із плаваючою комою, оскільки в даний час вони є цілими числами. Ми можемо зробити це, використовуючи *Numpy* команду *astype*(), а потім оголосити бажаний тип даних:

```
# normalize the inputs from 0-255 to between 0 and 1 by dividing by 255
X_train = X_train.astype('float32')
X_test = X_test.astype('float32')
X_train = X_train / 255.0
X_test = X_test / 255.0
```

Наступна річ, яку потрібно зробити, щоб підготувати дані для мережі – перевести їх в унітарний код. Не будемо вдаватися до подробиць унітарного кодування, але знайте, що зображення не можуть використовуватися нейромережею в тому вигляді, в якому вони є – їх потрібно спочатку кодувати, а найкраще при проведенні двійкової класифікації використовувати саме унітарне кодування.

Ми успішно застосовуємо тут двійкову класифікацію, тому що зображення або належить одному певному класу, або ні: воно не може бути посередині. Для унітарного кодування використається команда *Numpy to_categorical*().

Нам також потрібно задати кількість класів у наборі даних, щоб ми зрозуміли, до скількох нейронів стискати кінцевий шар:

```
# one hot encode outputs
y_train = np_utils.to_categorical(y_train)
y_test = np_utils.to_categorical(y_test)
class_num = y_test.shape[1]
```

Проектування моделі

Ми досягли стадії проектування моделі CNN. Перше, що потрібно зробити, це визначити формат, який ми б хотіли використовувати для моделі. *Keras* має кілька різних форматів (планів) для побудови моделей, але найчастіше

використовується *Sequential*.

Створення моделі

```
model = Sequential()
```

Перший шар прийматиме вхідні дані та пропускатиме їх через згорткові фільтри.

При реалізації цього в Keras, ми повинні вказати кількість каналів (фільтрів), які нам потрібні (а це 32), розмір фільтра (3x3 у нашому випадку), форму входу (при створенні першого шару), функцію активації та відступи.

Відступи ми визначимо через *padding = 'same'*:

```
model.add(Conv2D(32, (3, 3), input_shape=X_train.shape[1:], padding='same'))
model.add(Activation('relu'))
```

Тепер ми створимо виключний шар для запобігання перенавчанню, який випадково усуває з’єднання між шарами (0.2 означає, що він відкидає 20% існуючих з'єднань):

```
model.add(Dropout(0.2))
```

Також ми можемо виконати пакетну нормалізацію. Вона нормалізує вхідні дані, що надходять у наступний шар, гарантуючи, що мережа завжди створює функції активації з тим самим розподілом, який нам потрібний:

```
model.add(BatchNormalization())
```

Ще один згортковий шар, але розмір фільтра збільшується, так що мережа вже може вивчати складніші уявлення:

```
model.add(Conv2D(64, (3, 3), padding='same'))
model.add(Activation('relu'))
```

А ось і шар, що об’єднує, який, як обговорювалося раніше, допомагає зробити класифікатор зображень коректнішим, щоб

він міг вивчати релевантні шаблони. Знову опишемо виняток (*Dropout*) та пакетну нормалізацію:

```
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.2))
model.add(BatchNormalization())
```

Це основа робочого процесу у першій частині реалізації *CNN*: згортка, активація, виняток, об’єднання. Тепер ви розумієте, навіщо ми імпортували *Dropout*, *BatchNormalization*, *Activation*, *Conv2d* та *MaxPooling2d*.

Ви можете варіювати кількість згорткових шарів на свій смак, але кожен з них збільшує обчислювальні витрати. Зверніть увагу, що при додаванні згорткових шарів ви зазвичай збільшуєте і кількість фільтрів, щоб модель могла вивчити складніші уявлення. Якщо числа, вибрані для цих шарів, здаються дещо довільними, то просто знайте, що рекомендується збільшувати фільтри поступово, встановлюючи значення 2^n, що може дати невелику перевагу при навчанні моделі на GPU.

Важливо не мати занадто багато рівнів, що об'єднують, тому що кожен з них відкидає частину даних. Надто часте об’єднання призведе до того, що щільно пов’язані шари майже нічого не дізнаються, коли дані досягнуть їх.

Необхідна кількість шарів, що об'єднують, залежить від виконуваної задачі. Оскільки зображення в нашому наборі вже досить малі, ми не об’єднуватимемо їх більше двох разів.

Тепер ви можете повторити ці шари, щоб дати вашій мережі більше уявлень для роботи:

```
model.add(Conv2D(64, (3, 3), padding='same'))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.2))
model.add(BatchNormalization())
model.add(Conv2D(128, (3, 3), padding='same'))
model.add(Activation('relu'))
model.add(Dropout(0.2))
model.add(BatchNormalization())
```

Після того, як ми закінчили з шарами згортки, нам потрібно стиснути дані, тому ми імпортували функцію *Flatten* вище.

```
model.add(Flatten())
model.add(Dropout(0.2))
```

Використаємо імпортовану функцію *Dense* та створимо перший щільно пов'язаний шар. Нам потрібно вказати кількість нейронів у щільному шарі. Зверніть увагу, що кількість нейронів у наступних шарах зменшується, зрештою наближаючись до того ж числа нейронів, що і класи в наборі даних (в даному випадку 10). Обмеження ядра може впорядкувати дані в процесі навчання, що також допомагає запобігти перенавченню. Ось чому ми імпортували *maxnorm* раніше.

```
model.add(Dense(256, kernel_constraint=maxnorm(3)))
model.add(Activation('relu'))
model.add(Dropout(0.2))
model.add(BatchNormalization())
model.add(Dense(128, kernel_constraint=maxnorm(3)))
model.add(Activation('relu'))
model.add(Dropout(0.2))
model.add(BatchNormalization())
```

У цьому останньому шарі ми зрівнюємо кількість класів із числом нейронів. Кожен нейрон представляє клас, тому на виході цього шару буде вектор з 10 нейронів, кожен з яких зберігає деяку ймовірність того, що зображення, що розглядається, належить його класу.

Нарешті, функція активації *softmax* вибирає нейрон з найбільшою ймовірністю як своє вихідне значення, припускаючи, що зображення належить саме цьому класу:

```
model.add(Dense(class_num))
model.add(Activation('softmax'))
```

Тепер, коли ми розробили модель, яку хочемо використати, залишається лише її скомпілювати. Вкажемо кількість епох для навчання, а також оптимізатор, який ми хочемо використовувати.

Оптимізатор – це те, що налаштує ваги у вашій мережі так, щоб наблизитись до точки з найменшими втратами. Алгоритм Адама є одним із найчастіше використовуваних оптимізаторів, тому що він дає високу продуктивність у більшості завдань:

```
epochs = 25
optimizer = 'adam'
```

Тепер скомпілюємо модель з вибраними параметрами. Давайте вкажемо також метрику для оцінки.

```
model.compile(loss='categorical_crossentropy', optimizer=optimizer,
metrics=['accuracy'])
```

Ми також можемо роздрукувати зведення про модель, щоб отримати уявлення про модель загалом.

```
print(model.summary())
```

Роздруківка дасть нам деяку інформацію:

```
Results:

Layer (type) Output Shape Param #
=================================================================
conv2d_1 (Conv2D) (None, 32, 32, 32) 896
_________________________________________________________________
activation_1 (Activation) (None, 32, 32, 32) 0
_________________________________________________________________
dropout_1 (Dropout) (None, 32, 32, 32) 0
_________________________________________________________________
batch_normalization_1 (Batch (None, 32, 32, 32) 128
_________________________________________________________________
conv2d_2 (Conv2D) (None, 32, 32, 64) 18496
_________________________________________________________________
activation_2 (Activation) (None, 32, 32, 64) 0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 16, 16, 64) 0
_________________________________________________________________
dropout_2 (Dropout) (None, 16, 16, 64) 0
_________________________________________________________________
batch_normalization_2 (Batch (None, 16, 16, 64) 256
_________________________________________________________________
conv2d_3 (Conv2D) (None, 16, 16, 64) 36928
_________________________________________________________________
activation_3 (Activation) (None, 16, 16, 64)
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 8, 8, 64) 0
_________________________________________________________________
dropout_3 (Dropout) (None, 8, 8, 64) 0
_________________________________________________________________
batch_normalization_3 (Batch (None, 8, 8, 64) 256
_________________________________________________________________
conv2d_4 (Conv2D) (None, 8, 8, 128) 73856
_________________________________________________________________
activation_4 (Activation) (None, 8, 8, 128) 0
_________________________________________________________________
dropout_4 (Dropout) (None, 8, 8, 128) 0
_________________________________________________________________
batch_normalization_4 (Batch (None, 8, 8, 128) 512
_________________________________________________________________
flatten_1 (Flatten) (None, 8192) 0
_________________________________________________________________
dropout_5 (Dropout) (None, 8192) 0
_________________________________________________________________
dense_1 (Dense) (None, 256) 2097408
_________________________________________________________________
activation_5 (Activation) (None, 256) 0
_________________________________________________________________
dropout_6 (Dropout) (None, 256) 0
_________________________________________________________________
batch_normalization_5 (Batch (None, 256) 1024
_________________________________________________________________
dense_2 (Dense) (None, 128) 32896
_________________________________________________________________
activation_6 (Activation) (None, 128) 0
_________________________________________________________________
dropout_7 (Dropout) (None, 128) 0
_________________________________________________________________
batch_normalization_6 (Batch (None, 128) 512
_________________________________________________________________
dense_3 (Dense) (None, 10) 1290
_________________________________________________________________
activation_7 (Activation) (None, 10) 0
=================================================================
Total params: 2,264,458
Trainable params: 2,263,114
Non-trainable params: 1,344
```

Тепер ми розпочинаємо навчання моделі. Для цього нам потрібно викликати функцію *fit*() для моделі та передати вибрані параметри.

Ось де використовується *SEED*, вибраний для відтворення.

```
1.  numpy.random.seed(seed)
2.  model.fit(X_train, y_train, validation_data=(X_test, y_test),
epochs=epochs, batch_size=64)
numpy.random.seed(seed)
model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=epochs,
batch_size=64)
```

Візьмемо тренувальний набір 50000 зразків і перевірочний 10000 зразків.

Запуск цієї частини коду видасть:

```
Epoch 1/25

64/50000 [..............................] - ETA: 16:57 - loss: 3.1479 - acc: 0.0938
128/50000 [..............................] - ETA: 10:12 - loss: 3.0212 - acc: 0.0938
192/50000 [..............................] - ETA: 7:57 - loss: 2.9781 - acc: 0.1250
256/50000 [..............................] - ETA: 6:48 - loss: 2.8830 - acc: 0.1484
320/50000 [..............................] - ETA: 6:07 - loss: 2.8878 - acc: 0.1469
384/50000 [..............................] - ETA: 5:40 - loss: 2.8732 - acc: 0.1458
448/50000 [..............................] - ETA: 5:20 - loss: 2.8842 - acc: 0.1406

...
...
...

49664/50000 [============================>.] - ETA: 1s - loss: 1.5160 - acc: 0.4611
49728/50000 [============================>.] - ETA: 1s - loss: 1.5157 - acc: 0.4612
49792/50000 [============================>.] - ETA: 1s - loss: 1.5153 - acc: 0.4614
49856/50000 [============================>.] - ETA: 0s - loss: 1.5147 - acc: 0.4615
49920/50000 [============================>.] - ETA: 0s - loss: 1.5144 - acc: 0.4617
49984/50000 [============================>.] - ETA: 0s - loss: 1.5141 - acc: 0.4617
50000/50000 [==============================] - 262s 5ms/step - loss: 1.5140 - acc: 0.4618 - val_loss: 1.0715
- val_acc: 0.6195

End of Epoch 1
```

Зверніть увагу, що в більшості випадків вам потрібно мати перевірочний набір, відмінний від набору для тестування, тому ви повинні вказати відсоток даних навчання, які будуть використовуватися як набір для перевірки. У цьому випадку ми просто передамо тестові дані, щоб переконатися, що вони відкладені та не використовувалися для навчання.

Тепер ми можемо оцінити модель та подивитися, як вона працює. Просто викличте *model.evaluate*():

```
# Model evaluation
scores = model.evaluate(X_test, y_test, verbose=0)
print("Accuracy: %.2f%%" % (scores[1]*100))
```

І ось ми отримуємо результат:

```
Accuracy: 83.01%
```

10.4. Контрольні запитання

1. Який функціонал має бібліотека *TensorFlow* та *API Keras*?

2. Дайте опис першого шару нейронної мережі та поясніть, що буде означати розмір фільтра 2x2.

3. Опишіть стисло принцип роботи розпізнавання зображення, що був використаний у наведеному прикладі.

4. Для чого використовуються команди *Dropout*, *BatchNormalization*, *Activation*?

5. Чи зможе нейронна мережа працювати без унітарного кодування? Поясніть чому.

Лабораторна робота № 11. Розпізнавання мови з Python

11.1. Мета роботи

Вивчити методи оброблення природної мови (NLP) у Python. Навчитися аналізувати тональності та класифікувати тексти.

11.2. Теоретичні відомості

1. Аналіз тональності текстів на прикладі новин.

Обробка природної мови (Natural Language Processing – NLP) – це технологія, яка допомагає комп'ютеру розуміти природну мову людини. Використовуючи NLP, розробники можуть систематизувати і структурувати знання для виконання таких завдань, як автоматичне реферування, переклад, розпізнавання іменованих сутностей, аналіз тональності, розпізнавання мови, тематична сегментація та інше.

NLTK (Natural Language Toolkit) – провідна платформа для створення NLP-програм на Python. У неї легкі у використанні інтерфейси для багатьох мовних корпусів, а також бібліотеки для обробки текстів для класифікації, токенізації, стемінгу, розмічування частин мови (POS-тегування), синтаксичного аналізу та семантичного аналізу.

VADER (Valence Aware Dictionary and sEntiment Reasoner) – це інструмент оцінки тональності, доданий в NLTK у 2014 році. На відміну від інших методів, що вимагають навчання зв'язаного тексту перед використанням, VADER готовий до аналізу без будь-якої спеціального налаштування. VADER унікальний тим, що чітко розрізняє позитивність і негативність різного ступеня.

2. Класифікацію текстів на прикладі новин.

Частина 1. Аналіз тональності.

Класифікація текстів це процес **класифікації** документів в одну або кілька певних категорій. Слід відрізняти класифікацію текстів від кластеризації. В останньому випадку тексти також об'єднуються за деякими критеріями, але заздалегідь задані категорії відсутні. Класифікація може здійснюватися власноруч або автоматично, за допомогою створеного набору правил чи із застосуванням методів машинного навчання.

Присвоєння категорій текстів, які можуть бути веб-сторінкою, книгою, статтею для ЗМІ та іншим, має безліч застосувань, наприклад: аналіз настроїв, позначення тем, класифікація новин, визначення мови, виявлення намірів, виявлення спаму, маршрутизація клієнтів, класифікація резюме та інше.

11.3. Порядок виконання роботи

1. Аналіз тональності текстів на прикладі новин.

У нашому прикладі ми будемо використовувати, два інструменти NLTK:

Інструмент «Аналіз настроїв VADER» (генерує позитивні, негативні і нейтральні оцінки настроїв для заданих вхідних даних).

Інструмент токенізатора «word_tokenize» (розбиває великий текст на послідовність більш дрібних одиниць, таких як речення або слова).

Щоб використовувати VADER і word_tokenize, нам спочатку потрібно завантажити і встановити додаткові данні для NLTK.

```
import nltk
```

```
nltk.download('stopwords')
nltk.download('punkt')
nltk.download('vader_lexicon')
```

Обчислити тональність заголовку

```
pip install feedparser
Collecting feedparser
  Downloading feedparser-6.0.10-py3-none-any.whl (81 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━
81.1/81.1 kB 2.3 MB/s eta 0:00:00
Collecting sgmllib3k
  Downloading sgmllib3k-1.0.0.tar.gz (5.8 kB)
  Preparing metadata (setup.py) ... done
Installing collected packages: sgmllib3k, feedparser
  DEPRECATION: sgmllib3k is being installed using the legacy 'setup.py install'
method, because it does not have a 'pyproject.toml' and the 'wheel' package is
not installed. pip 23.1 will enforce this behaviour change. A possible re
placement is to enable the '--use-pep517' option. Discussion can be found at
https://github.com/pypa/pip/issues/8559
  Running setup.py install for sgmllib3k ... done
Successfully installed feedparser-6.0.10 sgmllib3k-1.0.0
```

```
import feedparser
posts = []
rss_url='https://www.pravda.com.ua/ukr/rss/view_news/'
response = feedparser.parse(rss_url)
for each in response['entries']:
  if each['title'] in [x['title'] for x in posts]:
    pass
  else:
    posts.append({
        "title": each['title'],
        "link": each['links'][0]['href'],
        "tags": [x['term'] for x in each['tags']],
        "date": time.strftime('%Y-%m-%d',
each['published_parsed'])
        })
for i, post in enumerate(p['title'] for p in posts):
  print(i, post)
```

```
0 Зеленський: Україна ламає так звану другу армію світу
1 В Україні за два тижні різко зріс попит на павербанки, свічки й буржуйки - Rozetka
2 Заніс біс Достоєвського: Пропагандист РФ вирішив виправдатися за заклик вбивати українських дітей
3 За щоденником маріупольчанки зняли анімаційний серіал про життя в окупації
4 Іран готовий з Україною розслідувати постачання Росії дронів і обіцяє "не залишитися байдужим"
5 Росія вже 85 разів вдарила по енергооб'єктах України, 51 атака була в жовтні - ОГП
6 Петиція про блокування "Страни.ua" набрала необхідні для розгляду 25 тисяч голосів
7 Кирило Буданов: У політичному керівництві РФ ніхто не радий від анексії наших областей
8 Стало відомо, коли Ріші Сунак стане прем’єром Британії
9 Музеєм Марії Примаченко, який спалили окупанти, можна "пройтися" у 3D-турі
10 Адвокат Богуслаєва підтвердив його російський паспорт
11 В українських лікарнях з’явиться послуга "духовна опіка" - МОЗ
12 Українська розвідка назвала причину падіння російського винищувача в Іркутську
13 Венеційська комісія похвалила Україну за процедуру очищення Вищої ради правосуддя
14 Бізнес вимагає удосконалити законодавство про бронювання працівників
15 Богуслаєв заявив, що задоволений рішенням суду, який взяв його під варту
16 РФ і "брудна бомба": США досі не бачать ознак підготовки Росії до ядерного удару
17 Нафтові санкції ЄС шкодять Росії ще до набуття чинності - Bloomberg
18 РосЗМІ: В Росії могли мобілізувати уже майже пів мільйона осіб
19 Справа "Мотор-Січі": суд взяв під варту Богуслаєва та Дзюбу
```

Завантажемо тональний словник української мови

```
import requests
import csv
url = 'https://raw.githubusercontent.com/lang-uk/tone-dict-uk/master/tone-dict-uk.tsv'
r = requests.get(url)
with open(nltk.data.path[0]+'/tone-dict-uk.tsv', 'wb') as f:
    f.write(r.content)

d = {}
with open(nltk.data.path[0]+'/tone-dict-uk.tsv', 'r') as csv_file:
    for row in csv.reader(csv_file, delimiter='\t'):
        d[row[0]] = float(row[1])

from nltk.sentiment.vader import SentimentIntensityAnalyzer
SIA = SentimentIntensityAnalyzer()

SIA.lexicon.update(d)
```

Алгоритм VADER виводить оцінки настроїв для 4 класів настроїв :

neg: Негативний
neu: Нейтральний
pos: Позитивний
compound: Складний (Сукупний бал)

Давайте порахуємо оцінку настрою для наданих заголовків за допомогою VADER, щоб зрозуміти, на що здатний цей інструмент.

```
for i, post in enumerate(p['title'] for p in posts):
  print(i, post, SIA.polarity_scores(post)["compound"])
```

```
0 У Нью-Йорку розпочався суд у справі про шахрайство бізнес-імперії Трампа 0.0
1 Економічні новини 24 жовтня: Шевченка оголосили в розшук, яким буде рівень безробіття на кінець року 0.0
2 Росія може отримати від Ірану новий вид дронів - Повітряні сили 0.0
3 Третій за добу: Ще один ворожий вертоліт Ка-52 "приземлили" на Херсонщині 0.0
4 Зеленський: Україна ламає так звану другу армію світу 0.0
5 В Україні за два тижні різко зріс попит на павербанки, свічки й буржуйки - Rozetka 0.0
6 Заніс біс Достоєвського: Пропагандист РФ вирішив виправдатися за заклик вбивати українських дітей 0.0
7 За щоденником маріупольчанки зняли анімаційний серіал про життя в окупації 0.0
8 Іран готовий з Україною розслідувати постачання Росії дронів і обіцяє "не залишитися байдужим" 0.0
9 Росія вже 85 разів вдарила по енергооб'єктах України, 51 атака була в жовтні - ОГП 0.0
10 Петиція про блокування "Страни.ua" набрала необхідні для розгляду 25 тисяч голосів 0.0
11 Кирило Буданов: У політичному керівництві РФ ніхто не радий від анексії наших областей 0.0
12 Стало відомо, коли Ріші Сунак стане прем’єром Британії 0.0
13 Музеєм Марії Примаченко, який спалили окупанти, можна "пройтися" у 3D-турі 0.0
14 Адвокат Богуслаєва підтвердив його російський паспорт 0.0
15 В українських лікарнях з’явиться послуга "духовна опіка" - МОЗ 0.0
16 Українська розвідка назвала причину падіння російського винищувача в Іркутську 0.0
17 Венеційська комісія похвалила Україну за процедуру очищення Вищої ради правосуддя 0.0
18 Бізнес вимагає удосконалити законодавство про бронювання працівників 0.0
19 Богуслаєв заявив, що задоволений рішенням суду, який взяв його під варту 0.0
```

Як видно з принту результату обробки — багато позицій визначено як нейтральна тональність (0.0), це через те, що ми не виконали попередньої підготовки тексту. Попередня обробка

тексту використовується для поліпшення роботи алгоритмів. Тож, виконаємо очищення від стоп-слів та приведемо слова в нормальну форму. Т.к. у NLTK, поки немає корпусу української мови, то для мормологічного аналізу скористаємось pymorphy2

```
pip install git+https://github.com/kmike/pymorphy2.git
Collecting git+https://github.com/kmike/pymorphy2.git
  Cloning https://github.com/kmike/pymorphy2.git to
c:\users\shich\appdata\local\temp\pip-req-build-hwd140oh
  Running command git clone --filter=blob:none --quiet
https://github.com/kmike/pymorphy2.git 'C:\Users\shich\AppData\Local\Temp\pip-
req-build-hwd140oh'
  Resolved https://github.com/kmike/pymorphy2.git to commit
92d546f042ff14601376d3646242908d5ab786c1
  Preparing metadata (setup.py) ... done
Collecting dawg-python>=0.7.1
  Downloading DAWG_Python-0.7.2-py2.py3-none-any.whl (11 kB)
Collecting pymorphy2-dicts-ru<3.0,>=2.4
  Downloading pymorphy2_dicts_ru-2.4.417127.4579844-py2.py3-none-any.whl (8.2
MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 8.2/8.2
MB 8.9 MB/s eta 0:00:00
Collecting docopt>=0.6
  Downloading docopt-0.6.2.tar.gz (25 kB)
  Preparing metadata (setup.py) ... done
Installing collected packages: pymorphy2-dicts-ru, docopt, dawg-python,
pymorphy2
  DEPRECATION: docopt is being installed using the legacy 'setup.py install'
method, because it does not have a 'pyproject.toml' and the 'wheel' package is
not installed. pip 23.1 will enforce this behaviour change. A possible repla
cement is to enable the '--use-pep517' option. Discussion can be found at
https://github.com/pypa/pip/issues/8559
  Running setup.py install for docopt ... done
  DEPRECATION: pymorphy2 is being installed using the legacy 'setup.py install'
method, because it does not have a 'pyproject.toml' and the 'wheel' package is
not installed. pip 23.1 will enforce this behaviour change. A possible re
placement is to enable the '--use-pep517' option. Discussion can be found at
https://github.com/pypa/pip/issues/8559
  Running setup.py install for pymorphy2 ... done
Successfully installed dawg-python-0.7.2 docopt-0.6.2 pymorphy2-0.9.1 pymorphy2-
dicts-ru-2.4.417127.4579844
```

```
pip install -U pymorphy2-dicts-uk
Collecting pymorphy2-dicts-uk
  Downloading pymorphy2_dicts_uk-2.4.1.1.1460299261-py2.py3-none-any.whl (5.0
MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 5.0/5.0
MB 9.6 MB/s eta 0:00:00
Installing collected packages: pymorphy2-dicts-uk
Successfully installed pymorphy2-dicts-uk-2.4.1.1.1460299261
```

```
url =
'https://raw.githubusercontent.com/olegdubetcky/Ukraini
an-Stopwords/main/ukrainian'
r = requests.get(url)
with open(nltk.data.path[0] +
```

```
'/corpora/stopwords/ukrainian', 'wb') as f:
    f.write(r.content)
# Retrieve HTTP meta-data
print(r.status_code)
print(r.headers['content-type'])
print(r.encoding)

stopwords = stopwords.words("ukrainian")

print("Print 3:")
morph = pymorphy2.MorphAnalyzer(lang='uk')
stop_words = frozenset(stopwords +
list(string.punctuation))
for i, post in enumerate(p['title'] for p in posts):
    sentences = nltk.sent_tokenize(post)
    for sentence in sentences:
        words = nltk.word_tokenize(sentence)
        without_stop_words = [word for word in words if
not word in stop_words]
        normal_words = []
        for token in without_stop_words:
            p = morph.parse(token)[0]
            normal_words.append(p.normal_form)

        print(i, post, "RAW: ",
SIA.polarity_scores(post)["compound"], "NORM: ",
              SIA.polarity_scores('
'.join(normal_words))["compound"])
```

```
200
text/plain; charset=utf-8
utf-8
```

```
0 Росія і "брудна бомба": МАГАТЕ відвідає два ядерні об’єкти в Україні на її
запит RAW:  0.0 NORM:  0.0
1 30 лівих демократів закликають Байдена до прямих переговорів із Росією RAW:
0.0 NORM:  0.0
2 Під час окупації Київщини загарбники незаконно вивезли 147 мирних українців –
ОВА RAW:  0.0 NORM:  0.0
3 Уряд Молдови пропонує обмежити використання ліфтів та освітлення для економії
електроенергії RAW:  0.0 NORM:  0.0
4 Журналісти ідентифікували 30 російських військових, які наводять ракети по
Україні RAW:  0.0 NORM:  0.0
5 В Брянській області Росії підірвали залізничну колію, що веде до Білорусі RAW:
0.0 NORM:  0.0
6 У Нью-Йорку розпочався суд у справі про шахрайство бізнес-імперії Трампа RAW:
0.0 NORM:  0.0
7 Економічні новини 24 жовтня: Шевченка оголосили в розшук, яким буде рівень
безробіття на кінець року RAW:  0.0 NORM:  0.0
8 Росія може отримати від Ірану новий вид дронів – Повітряні сили RAW:  0.0
NORM:  0.0
9 Третій за добу: Ще один ворожий вертоліт Ка-52 "приземлили" на Херсонщині RAW:
0.0 NORM:  0.0
10 Зеленський: Україна ламає так звану другу армію світу RAW:  0.0 NORM:  0.0
11 В Україні за два тижні різко зріс попит на павербанки, свічки й буржуйки –
Rozetka RAW:  0.0 NORM:  0.0
12 Заніс біс Достоєвського: Пропагандист РФ вирішив виправдатися за заклик
вбивати українських дітей RAW:  0.0 NORM:  0.0
13 За щоденником маріупольчанки зняли анімаційний серіал про життя в окупації
RAW:  0.0 NORM:  0.0
14 Іран готовий з Україною розслідувати постачання Росії дронів і обіцяє "не
залишитися байдужим" RAW:  0.0 NORM:  0.0
15 Росія вже 85 разів вдарила по енергооб'єктах України, 51 атака була в жовтні
– ОГП RAW:  0.0 NORM:  0.0
16 Петиція про блокування "Страни.ua" набрала необхідні для розгляду 25 тисяч
голосів RAW:  0.0 NORM:  0.0
17 Кирило Буданов: У політичному керівництві РФ ніхто не радий від анексії наших
областей RAW:  0.0 NORM:  0.0
18 Стало відомо, коли Ріші Сунак стане прем’єром Британії RAW:  0.0 NORM:  0.0
19 Музеєм Марії Примаченко, який спалили окупанти, можна "пройтися" у 3D-турі
RAW:  0.0 NORM:  0.0
```

Тож, результат значно краще.

2. Класифікацію текстів на прикладі даних.

```
import pandas as pd
df =
pd.read_csv('https://raw.githubusercontent.com/olegdubetcky/Text-Classification-with-ML-Project/main/news.csv', encoding='utf8')
```

Отже, це наш набір даних.

Текстові значення не можна використовувати безпосередньо в моделі машинного навчання. Щоб використовувати їх, нам потрібно перетворити їх в числові значення або

функції. Виділення ознак і розробка ознак в тексті відносяться до вилучення інформації з даних в числовому форматі. Щоб не ускладнювати завдання, ми використовуємо простий і інтуїтивно зрозумілий метод перетворення нашого тексту в корисні функції TF-IDF.

```
#перетворіть усі дані у нижній регістр, а потім
збережіть їх у форматі списку
corpus = df['title'].apply(lambda x :
str(x).lower()).tolist()
#встановити цільові змінні
y = df['category']
```

Ми обрали метод для перетворення нашого тексту в функції. Зараз залишається лише підібрати модель, але яку? Важливо випробувати різні моделі, перш ніж приймати рішення про остаточну. Крім того, моделювання найкращого алгоритму класифікації є дуже простим завданням за допомогою scikit-learn — ми можемо бачити результати різних моделей за допомогою функцій fit() та predict().

Реалізовані моделі:

- Multinomial Naive Bayes
- Support Vector Machines (SVM, LinearSVM)
- Neural Network with Softmax Layer
- Decision Trees
- Random Forests

Метрики, що використовуються для оцінки продуктивності моделей:

- Точність
- Влучність
- Повнота
- F1 міра
- Cohens Kappa міра
- Матриця невідповідностей

Ми оцінюємо здатність кожного класифікатора вибрати відповідну категорію з урахуванням назви статті. Матриця

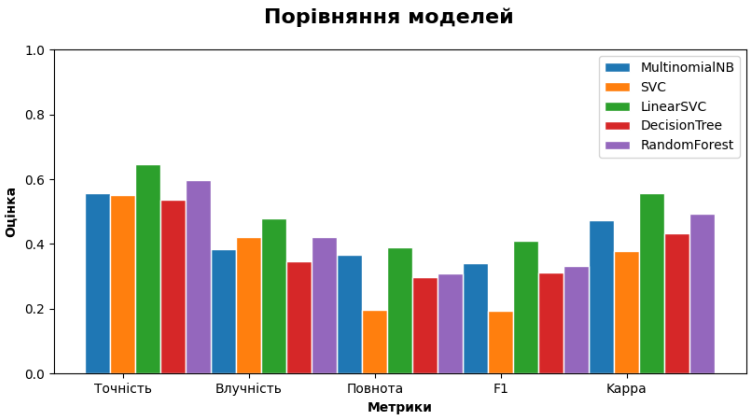
невідповідностей створюється для вивчення результатів і розрахунку показників.

```
import pickle
from sklearn.feature_extraction.text import
CountVectorizer

count_vect = CountVectorizer()
X_train_counts = count_vect.fit_transform(corpus)

#зберегти ветори слів
pickle.dump(count_vect.vocabulary_,
open("feature.pkl","wb"))

from sklearn.feature_extraction.text import
TfidfTransformer

#перетворення векторів слів до TF IDF
tfidf_transformer = TfidfTransformer()
X = tfidf_transformer.fit_transform(X_train_counts)

#зберегти TF-IDF
pickle.dump(tfidf_transformer, open("tfidf.pkl","wb"))

#розділити дані на зразки для навчання та тестування
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,
y, test_size=0.25, random_state=42)
```

Щоб оцінити, яка з наших моделей працює краще, нам потрібно розрахувати та порівняти точність між класифікаторами. Оскільки це повторюваний процес, ми можемо полегшити нам, визначивши функцію для оцінки класифікаторів.

```
from sklearn.naive_bayes import MultinomialNB
from sklearn.neural_network import MLPClassifier
from sklearn.svm import SVC, LinearSVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier

from sklearn import metrics
```

```
# проста функція оцінки
def evaluate(clf, X_train=X_train, X_test=X_test,
y_train=y_train, y_test=y_test):

    import seaborn as sns

    print('Класифікатор : {}'.format(clf))
    clf.fit(X_train, y_train)
    # Predict Test Data
    y_pred = clf.predict(X_test)

    # Calculate accuracy, precision, recall, f1-score,
and kappa score
    acc = metrics.accuracy_score(y_test, y_pred)
    prc = metrics.precision_score(y_test, y_pred,
average='macro')
    rec = metrics.recall_score(y_test, y_pred,
average='macro')
    f1 = metrics.f1_score(y_test, y_pred,
average='macro')
    kappa = metrics.cohen_kappa_score(y_test, y_pred)

    # Display confussion matrix
    cm = metrics.confusion_matrix(y_test, y_pred)

    print('Метрики : \n')
    # Print result
    print('Точність:', acc)
    print('Влучність:', prc)
    print('Повнота:', rec)
    print('F1 міра:', f1)
    print('Cohens Kappa міра:', kappa)
    print('Матриця невідповідностей:\n', cm)

    print('*' * 100)
    print('\n\n')

    return {'acc': acc, 'prc': prc, 'rec': rec, 'f1':
f1, 'kappa': kappa, 'cm': cm}
```




```
from sklearn.model_selection import GridSearchCV

#спробуйте налаштувати параметри. Для LinearSVC C є
регульованим параметром
params = {'C': [0.1, 1, 10, 100, 1000]}

#використовувати клас GridSearchCV. Основними
аргументами є (класифікатор, параметри), для яких ми
хочемо виконати найкращий пошук параметрів.
grid = GridSearchCV(LinearSVC(), params, refit=True,
verbose=3)

#пристосувати дані навчання до нашої моделі grid_search
та перевірити прогнози на нашому тестовому наборі
grid.fit(X_train, y_train)
y_pred = grid.predict(X_test)

#Перевірте точність і влучність нашої моделі
print('Точність :
{:.2f}%'.format(100*metrics.accuracy_score(y_pred,
y_test)))
print('Влучність :
\n{}'.format(100*metrics.precision_score(y_pred,
y_test, average = None)))
```

Оскільки ми знайшли нашу кращу модель і налаштували її, ми хочемо зберегти її, щоб виконувати прогнози на майбутнє безпосередньо, без необхідності заново навчати класифікатор.

Нагадаємо, що ми використовували векторизатор, який також відповідав нашим даними. Таким чином, вкрай важливо зберігати один і той же векторизатор для наших тестових і прогнозних даних.

```
#зберегти модель категорії
pickle.dump(grid, open('LinearSVM_model.pkl', 'wb'))

# завнтажити модель категорії та перевірити
tuned_model = pickle.load(open('LinearSVM_model.pkl',
'rb'))
result = tuned_model.score(X_test, y_test)*100

print('Точність : {:.2f}%'.format(result))

loaded_vec =
CountVectorizer(vocabulary=pickle.load(open("feature.pk
l", "rb")))
loaded_tfidf = pickle.load(open("tfidf.pkl","rb"))
loaded_model =
pickle.load(open("LinearSVM_model.pkl","rb"))

docs_new = "Макрон планує знову обговорити з Путіним
ситуацію в Україні"
X_new_counts = loaded_vec.transform([docs_new])
X_new_tfidf = loaded_tfidf.transform(X_new_counts)
predicted = loaded_model.predict(X_new_tfidf)
print('Категорія: ', predicted[0])
```

11.4. Контрольні запитання

1. Що називають «Обробка природної мови»?
2. Що таке «Natural Language Toolkit»?
3. Що таке «VADER»?
4. Опишіть класифікацію текстів на прикладі новин.
5. Опишіть класифікацію текстів на прикладі даних.

Лабораторна робота №12. Виявлення об’єктів, використовуючи YOLO, OpenCV та PyTorch у Python

12.1. Мета роботи

Вивчити складову програмування по виявленню об’єктів, використовуючи *YOLO*, *OpenCV* та *PyTorch* у *Python* і навчитися створювати власну програму.

12.2. Теоретичні відомості

Виявлення об’єктів – це завдання комп’ютерного зору та обробки зображень, пов’язане з виявленням об’єктів на зображеннях або відео. Наразі рішення подібних завдань актуальні в різних реальних додатках, включаючи відеоспостереження, безпілотні автомобілі, відстеження об’єктів тощо.

Наприклад, для того, щоб автомобіль був дійсно автономним, він повинен розуміти і відстежувати навколишні об’єкти (такі як автомобілі, пішоходи та світлофори), і основним джерелом інформації для цього є камера. Більш того, щоб автомобіль міг безпечно пересуватися вулицею, дуже важливо виявляти об’єкти в режимі реального часу.

YOLO (*You Only Look Once*) – це алгоритм виявлення об’єктів у реальному часі, який є єдиною згортковою нейронною мережею, яка розбиває вхідне зображення на набір осередків, що утворюють сітку, тому, на відміну від класифікації зображень або виявлення осіб, кожен осередок сітки в алгоритмі *YOLO* у вихідних даних буде мати зв’язаний вектор, який повідомляє нам:

- Чи є об’єкт у комірці сітки.
- Клас об’єкта (тобто мітка).
- Ймовірні геометричні характеристики об’єкта (місце розташування).

Існують і інші підходи, такі як *Fast R-CNN*, *Faster R-CNN*, які

використовують ковзаючі за зображенням вікна, що вимагає тисячі прогнозів для одного зображення (у кожному вікні), як ви можете здогадатися, це робить *YOLO* v3 приблизно в 1000 разів швидше, ніж *R-CNN*, і у 100 разів швидше, ніж *Fast R-CNN*.

12.3. Порядок виконання роботи

Перш ніж ми заглибимося в код, давайте встановимо необхідні бібліотеки:

```
pip3 install opencv-python numpy matplotlib
```

Створити всю систему *YOLOv3* (модель і використані методи) з нуля досить складно. Бібліотеки з відкритим кодом, такі як *Darknet* або *OpenCV*, уже зробили це для вас, або навіть звичайні люди створили сторонні проекти для *YOLOv3*.

Імпорт необхідних модулів:

```
import cv2
import numpy as np
import time
import sys
import os
```

Давайте визначимо деякі змінні та параметри, які нам знадобляться:

```
CONFIDENCE = 0.5
SCORE_THRESHOLD = 0.5
IOU_THRESHOLD = 0.5
# конфигурация нейронной сети
config_path = "cfg/yolov3.cfg"
# файл весов сети YOLO
weights_path = "weights/yolov3.weights"
# weights_path = "weights/yolov3-tiny.weights"
# загрузка всех меток классов (объектов)
labels = open("data/coco.names").read().strip().split("\n")
# генерируем цвета для каждого объекта и последующего
построения
```

```
colors = np.random.randint(0, 255, size=(len(LABELS), 3),
dtype="uint8")
```

Ми ініціалізували наші параметри, але поговоримо про них згодом. `config_path` і `weights_path` є відповідно конфігурацією моделі і попередньо навчені ваги моделі. `labels` – це список всіх позначок класів для різних об'єктів. Намалюємо кожен клас об'єктів унікальним кольором. Ось чому ми знгенерували випадкові кольори.

За необхідними файлами переходимо до цього репозиторію та виконуємо необхідні інструкції.

Наведений нижче код завантажує модель:

```
# завантажуємо мережу YOLO
net = cv2.dnn.readNetFromDarknet(config_path, weights_path)
```

Підготовка зображення

Завантажимо приклад зображення (зображення є в репозиторії):

```
path_name = "images/street.jpg"
image = cv2.imread(path_name)
file_name = os.path.basename(path_name)
filename, ext = file_name.split(".")
```

Потім потрібно нормалізувати, масштабувати і змінити це зображення, щоб воно підходило як вхідні дані для нейронної мережі:

```
h, w = image.shape[:2]
# створити 4D blob
blob = cv2.dnn.blobFromImage(image, 1/255.0, (416, 416),
swapRB=True, crop=False)
```

Це нормалізує значення пікселів у діапазоні від `0` до `1`, змінить розмір зображення до `(416, 416)` і змінить його форму.

```
print("image.shape:", image.shape)
print("blob.shape:", blob.shape)
```

Результат:

```
image.shape: (1200, 1800, 3)
blob.shape: (1, 3, 416, 416)
```

Прогнозування

Тепер завантажимо зображення в нейронну мережу та отримаємо прогноз на виході:

```
# встановлює blob як вхід мережі
net.setInput(blob)
# отримуємо імена всіх шарів
ln = net.getLayerNames()
ln = [ln[i[0] - 1] for i in net.getUnconnectedOutLayers()]
# прямий зв'язок (виведення) та отримання виходу мережі
# вимір часу для обробки в секундах
start = time.perf_counter()
layer_outputs = net.forward(ln)
time_took = time.perf_counter() - start
print(f"Знадобилося: {time_took:.2f}s")
```

Як результат, для отримання вихідних даних нейронної мережі знадобилося:

```
Знадобилося: 1.54s
```

Тепер ви, можливо, дивуєтесь, чому це не так швидко? Ми використовуємо наш ЦП лише для висновків, що не ідеально підходить для реальних проблем. Ось чому ми перейдемо до *PyTorch* пізніше. З іншого боку, 1.5 секунди – це порівняно краще з іншими методами, такими як *R-CNN*.

Тепер нам потрібно перебрати вихідні дані нейронної мережі та відкинути всі об’єкти, рівень достовірності ідентифікації яких менший за параметр *CONFIDENCE*, вказаний нами раніше.

```
font_scale = 1
thickness = 1
```

```
boxes, confidences, class_ids = [], [], []
# перебираємо кожен із виходів шару
for output in layer_outputs:
    # перебираємо кожне виявлення об'єкта
    for detection in output:
        # отримуємо ідентифікатор класу (мітку) та
достовірність (як імовірність)
         # Виявлення поточного об'єкта
        scores = detection[5:]
        class_id = np.argmax(scores)
        confidence = scores[class_id]
        # відхиляємо слабкі передбачення, переконавшись у їх
виявленні
        if confidence > CONFIDENCE:
            # масштабуємо координати обмежуючого рамки
відносно
            # розміру зображення, враховуючи, що YOLO
            # повертає координати центру (x, y)
обмежувальної
            # рамки, а потім її ширину та висоту
            box = detection[:4] * np.array([w, h, w, h])
            (centerX, centerY, width, height) =
box.astype("int")
            # використовуємо координати центру (x, y), щоб
отримати верхній
            # і лівий кути обмежувальної рамки
            x = int(centerX - (width / 2))
            y = int(centerY - (height / 2))
            # оновити наш список координат обмежувальної
рамки, конфіденційності,
            # та ідентифікаторів класу
            boxes.append([x, y, int(width), int(height)])
            confidences.append(float(confidence))
            class_ids.append(class_id)
```

Тут перебираються всі прогнози та зберігаються об’єкти з високим ступенем достовірності, давайте подивимося, що є вектором `detection`:

```
print(detection.shape)
```

Результат:

```
(85,)
```

У прогнозі кожного об’єкта є вектор з 85 елементів. Перші чотири значення представляють розташування об’єкта, (x, y) координати центруючої точки та ширину та висоту обмежувальної рамки, а решта чисел відповідають мітки об’єктів, оскільки це набір даних COCO. Має 80 міток класів.

Наприклад, якщо виявлений об’єкт – людина, перше значення у векторі довжини 80 має бути 1, а всі інші значення повинні бути 0, число 2 для велосипеда, 3 для автомобіля, аж до 80 об’єкта. Ось чому ми використовуємо функцію *np.argmax*() для отримання класу ідентифікатора, оскільки вона повертає індекс максимального значення з цього вектора довжиною 80.

Малюємо виявлені об’єкти

Тепер у нас є все, що потрібно, і ми зможемо намалювати прямокутники об’єктів та мітки.

```
# перебираємо збережені індекси
for i in range(len(boxes)):
    # вилучаємо координати обмежуючого прямокутника
    x, y = boxes[i][0], boxes[i][1]
    w, h = boxes[i][2], boxes[i][3]
    # малюємо прямокутник обмежувальної рамки, і підписуємо
на зображенні
    color = [int(c) for c in colors[class_ids[i]]]
    cv2.rectangle(image, (x, y), (x + w, y + h),
color=color, thickness=thickness)
    text = f"{labels[class_ids[i]]}: {confidences[i]:.2f}"
    # обчислюємо ширину та висоту тексту, щоб малювати
прозорі поля як тло тексту
    (text_width, text_height) = cv2.getTextSize(text,
cv2.FONT_HERSHEY_SIMPLEX, fontScale=font_scale,
thickness=thickness)[0]
    text_offset_x = x
    text_offset_y = y - 5
```

```
    box_coords = ((text_offset_x, text_offset_y),
(text_offset_x + text_width + 2, text_offset_y -
text_height))
    overlay = image.copy()
    cv2.rectangle(overlay, box_coords[0], box_coords[1],
color=color, thickness=cv2.FILLED)
    # додати непрозорість (прозорість поля)
    image = cv2.addWeighted(overlay, 0.6, image, 0.4, 0)
    # тепер помістіть текст (мітка: впевненість %)
    cv2.putText(image, text, (x, y - 5),
cv2.FONT_HERSHEY_SIMPLEX,
        fontScale=font_scale, color=(0, 0, 0),
thickness=thickness)
```

Напишемо малюнок:

```
cv2.imwrite(filename + "_yolo3." + ext, image)
```

У поточному каталозі з’явиться нове зображення, яке впевнено маркує кожний виявлений об’єкт. Однак подивіться на цю частину зображення:

ї

Як ви вже здогадалися, дві рамки, що обмежують, для одного об’єкта, це проблема, чи не так? Що ж, творці YOLO для її вирішення використовували техніку під назвою *Non-Maximal Suppression*.

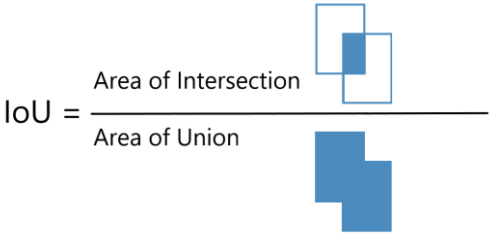
Non-maximal Suppression

Non-maximal Suppression – метод, який заглушує обмежуючі прямокутники, що накладаються, які не мають максимальної ймовірності виявлення об’єкта. В основному це досягається у два етапи:

- Вибір обмежувальної рамки з найвищою достовірністю.
- Порівняння її з ймовірностями всіх інших обмежуючих рамок, і видаляємо ті, які мають високий *IoU*.

Що таке IoU

IoU (*Intersection over Union* або перетин над об'єднанням) – це метод, що використовується в *Non-maximal Suppression* для порівняння того, наскільки близькі два різні прямокутники, що обмежують. Це просто показано на наступному малюнку:



Що вищий *IoU*, то ближче обмежувальні рамки. *IoU*, рівний 1, означає, що дві обмежувальні рамки ідентичні, тоді як *IoU*, рівний 0, означає, що вони навіть не перетинаються..

В результаті ми будемо використовувати граничне значення *IoU* 0.5 і це означає, що буде видалено будь-яка рамка, що обмежує значення нижче значення обмежуючої рамки з максимальною ймовірністю.

SCORE_THRESHOLD видаляє будь-яку обмежуючу рамку, яка має достовірність нижче цього значення:

```
# виконати не максимальне придушення, враховуючи бали,
визначені раніше
idxs = cv2.dnn.NMSBoxes(boxes, confidences, SCORE_THRESHOLD,
IOU_THRESHOLD)
```

Тепер давайте знову намалюємо рамки:

```
# перевірка, що виявлено хоча б один об'єкт
if len(idxs) > 0:
    for i in idxs.flatten():
        x, y = boxes[i][0], boxes[i][1]
        w, h = boxes[i][2], boxes[i][3]
        color = [int(c) for c in colors[class_ids[i]]]
        cv2.rectangle(image, (x, y), (x + w, y + h),
color=color, thickness=thickness)
        text = f"{labels[class_ids[i]]}:
{confidences[i]:.2f}"
        (text_width, text_height) = cv2.getTextSize(text,
cv2.FONT_HERSHEY_SIMPLEX, fontScale=font_scale,
thickness=thickness)[0]
        text_offset_x = x
        text_offset_y = y - 5
        box_coords = ((text_offset_x, text_offset_y),
(text_offset_x + text_width + 2, text_offset_y -
text_height))
        overlay = image.copy()
        cv2.rectangle(overlay, box_coords[0], box_coords[1],
color=color, thickness=cv2.FILLED)
        image = cv2.addWeighted(overlay, 0.6, image, 0.4, 0)
        cv2.putText(image, text, (x, y - 5),
cv2.FONT_HERSHEY_SIMPLEX,
            fontScale=font_scale, color=(0, 0, 0),
thickness=thickness)
```

Ви можете використовувати *cv2.imshow*("*image*", *image*) для виведення зображення, але ми просто збережемо його на робочому просторі:

```
cv2.imwrite(filename + "_yolo3." + ext, image)
```

Крім того, якщо зображення має високу роздільну здатність, переконайтеся, що ви збільшили параметр *font_scale*, щоб побачити обмежуючі рамки та відповідні мітки.

Код PyTorch

Як згадувалося раніше, якщо ви хочете використовувати графічний процесор для виведення, можете спробувати бібліотеку *PyTorch*, яка підтримує обчислення *CUDA*, ось код для цього (завантажте *darknet.py* та *utils.py* з репозиторію):

```
import cv2
import matplotlib.pyplot as plt
from utils import *
from darknet import Darknet

# Встановити поріг NMS
nms_threshold = 0.6
# Встановити поріг IoU
iou_threshold = 0.4
cfg_file = "cfg/yolov3.cfg"
weight_file = "weights/yolov3.weights"
namesfile = "data/coco.names"
m = Darknet(cfg_file)
```

```
m.load_weights(weight_file)
class_names = load_class_names(namesfile)
# m.print_network()
original_image = cv2.imread("images/city_scene.jpg")
original_image = cv2.cvtColor(original_image,
cv2.COLOR_BGR2RGB)
img = cv2.resize(original_image, (m.width, m.height))
# виявляємо об'єкти
boxes = detect_objects(m, img, iou_threshold, nms_threshold)
# викреслюємо зображення з обмежувальними рамками і
відповідними мітками класів об'єктів
plot_boxes(original_image, boxes, class_names,
plot_labels=True)
```

12.4. Контрольні запитання

1. Наведіть визначення терміну «виявлення об’єктів» та опишіть сфери використання даної технології.
2. Який функціонал наявний YOLO у реалізації виявлення об’єктів на базі мови програмування Python?
3. Опишіть етап програми на якому відбувається прогнозування.
4. Чому деколи виникає проблема обмежуючих прямокутників, що накладаються один на одного? Як це вирішити?
5. За що відповідає *IoU*?

Лабораторна робота №13. Виявлення об’єктів у TensorFlow: виявлення об’єктів у реальному часі

13.1. Мета роботи

Вивчити процедуру створення програми виявлення об’єктів у реальному часі.

13.2. Теоретичні відомості

Створення точних моделей машинного навчання, здатних ідентифікувати та локалізувати кілька об’єктів на одному зображенні, залишається основною проблемою комп’ютерного зору. Але з недавніми досягненнями в галузі глибокого навчання, програми для виявлення об’єктів стали легше розробляти, ніж будь-коли раніше. Виявлення об’єкта може бути зроблено декількома способами:

- Виявлення об’єктів на основі об’єктів
- Виявлення об’єкту Віола Джонс
- Класифікація *SVM* з функціями *HOG*
- Глибоке виявлення об’єкта навчання

Програми виявлення об’єктів

Розпізнавання обличчя

Група дослідників із *Facebook* розробила систему глибокого навчання особі під назвою «*DeepFace*», яка дуже ефективно ідентифікує людські обличчя у цифровому зображенні. *Google* використовує свою власну систему розпізнавання облич у *Google* Фото, яка автоматично поділяє всі фотографії на основі людини на зображенні. У розпізнаванні осіб беруть участь різні компоненти, такі як очі, ніс, рот та брови.

Підрахунок людей

Виявлення об’єктів також можна використовувати для підрахунку людей. Він використовується для аналізу продуктивності магазину або натовп статистики під час

фестивалів. Це, як правило, складніше, тому що люди швидко виходять за межі.

Це дуже важлива програма, оскільки під час скупчення людей ця функція може використовуватися для кількох цілей.

Промислова перевірка якості

Виявлення об’єктів також використовують у виробничих процесах для ідентифікації продуктів. Пошук конкретного об’єкта з допомогою візуального огляду є основним завданням, яка задіяна у багатьох виробничих процесах, як-от сортування, управління запасами, обробка, управління якістю, упаковка тощо.

Управління запасами може бути дуже складним, оскільки елементи важко відстежувати як реального часу. Автоматичний підрахунок та локалізація об’єктів дозволяє підвищити точність інвентаризації.

Самостійні автомобілі

Самостійні автомобілі – це майбутнє; у цьому немає жодних сумнівів. Але працювати за ним дуже складно, оскільки він поєднує в собі різні методи для сприйняття навколишнього оточення, включаючи радар, лазерне випромінювання, GPS, одометрію та комп'ютерний зір.

Удосконалені системи управління інтерпретують сенсорну інформацію для визначення відповідних шляхів навігації, а також перешкод, і як датчик зображення виявляє будь-які ознаки живої істоти на своєму шляху, він автоматично зупиняється. Це відбувається з дуже високою швидкістю та є великим кроком на шляху до автомобілів без водія.

Безпека

Виявлення об’єктів відіграє дуже важливу роль у безпеці. Будь то ідентифікація обличчя *Apple* або сканування сітківки, яке використовується у всіх науково-фантастичних фільмах.

Він також використовується урядом для доступу до каналів

безпеки та зіставлення їх із існуючою базою даних, щоб знайти злочинців або виявити автомобіль грабіжників.

Робочий процес виявлення об’єктів

Кожен алгоритм виявлення об’єктів має свій спосіб роботи, але всі вони працюють за одним і тим же принципом.

Вилучення функцій: вони витягують елементи з наявних зображень і використовують ці функції для визначення класу зображення. Чи це через *MatLab*, *Open CV*, Альт-Джонс або глибоке навчання.

12.3. Порядок виконання роботи

Передумови

Перш ніж розпочати демонстрацію, давайте подивимося на попередні умови:

- *Python*
- *TensorFlow*
- *TensorBoard*
- *Protobuf v*3.4 або вище

Налаштування середовища

Щоб завантажити *TensorFlow* та *TensorFlow GPU*, ви можете використовувати команди *pip* або *conda*:

```
# For CPU

pip install tensorflow

# For GPU

pip install tensorflow-gpu
```

Для решти бібліотек ми можемо використовувати *pip* або *conda* для їх встановлення. Код наданий нижче:

```
pip install --user Cython
```

```
pip install --user contextlib2

pip install --user pillow

pip install --user lxml

pip install --user jupyter

pip install --user matplotlib
```

Далі, у нас є *Protobuf*: *Protocol Buffers* – це не залежний від мови чи платформи, механізм *Google*, що розширюється, для серіалізації структурованих даних. Подібно до *XML*, але швидше та простіше. Вам потрібно завантажити *Protobuf* версії 3.4 або вище для цієї демонстрації та розпакувати її.

Тепер вам потрібно клонувати або завантажити *TensorFlow's Model* з *GitHub*. Після завантаження та вилучення перейменуйте «моделі-майстри» на «моделі».

Для простоти ми зберігатимемо «*models*» та «*protobuf*» в одній папці з ім'ям «*Tensorflow*».

Далі нам потрібно перейти до папки *Tensorflow*, а потім до папки дослідження і запустити звідти *protobuf* за допомогою цієї команди:

```
"path_of_protobuf's bin"./bin/protoc object_detection/protos/
```

Щоб перевірити, чи це спрацювало чи ні, ви можете перейти в папку *protos models> object_detection> protos* і там, ви можете побачити, що для кожного *proto*-файлу створено один файл *python*.

Основний код

Після налаштування середовища вам потрібно перейти до каталогу *object_detection* і створити новий файл *Python*. Ви можете використати *Spyder* або *Jupyter*, щоб написати свій код.

Насамперед, нам потрібно імпортувати всі бібліотеки.

```
import numpy as np

import os
```

```
import six.moves.urllib as urllib

import sys

import tarfile

import tensorflow as tf

import zipfile

from collections import defaultdict

from io import StringIO

from matplotlib import pyplot as plt

from PIL import Image

sys.path.append("..")

from object_detection.utils import ops as utils_ops

from utils import label_map_util

from utils import visualization_utils as vis_util
```

Далі ми завантажимо модель, яка навчається на наборі *COCO даних*. *COCO* розшифровується як *Common Objects in Context*, і цей набір даних містить близько 330 тис. зображень. Вибір моделі є важливим, тому що вам потрібно знайти компроміс між швидкістю і точністю. Залежно від ваших вимог та системної пам'яті має бути обрана правильна модель.

«*Models > research > object_detection > g3doc > Detection_model_zoo*» містить усі моделі з різною швидкістю та точністю (*mAP*).

Model name	Speed (ms)	COCO mAP[^1]	Outputs
ssd_mobilenet_v1_coco	30	21	Boxes
ssd_mobilenet_v1_0.75_depth_coco ☆	26	18	Boxes
ssd_mobilenet_v1_quantized_coco ☆	29	18	Boxes
ssd_mobilenet_v1_0.75_depth_quantized_coco ☆	29	16	Boxes
ssd_mobilenet_v1_ppn_coco ☆	26	20	Boxes
ssd_mobilenet_v1_fpn_coco ☆	56	32	Boxes
ssd_resnet_50_fpn_coco ☆	76	35	Boxes
ssd_mobilenet_v2_coco	31	22	Boxes

Далі ми надаємо необхідну модель та заморожений граф виводу, згенерований *Tensorflow*.

```
MODEL_NAME = 'ssd_mobilenet_v1_coco_2017_11_17'
MODEL_FILE = MODEL_NAME + '.tar.gz'
DOWNLOAD_BASE = 'http://download.tensorflow.org/models/object_detection/'
PATH_TO_CKPT = MODEL_NAME + '/frozen_inference_graph.pb'
PATH_TO_LABELS = os.path.join('data', 'mscoco_label_map.pbtxt')
NUM_CLASSES = 90
```

Цей код завантажить цю модель з Інтернету та витягне заморожений графік виведення цієї моделі.

```
opener = urllib.request.URLopener()
opener.retrieve(DOWNLOAD_BASE + MODEL_FILE, MODEL_FILE)
tar_file = tarfile.open(MODEL_FILE)
for file in tar_file.getmembers():
  file_name = os.path.basename(file.name)
  if 'frozen_inference_graph.pb' in file_name:
    tar_file.extract(file, os.getcwd())
detection_graph = tf.Graph()
with detection_graph.as_default():
```

```
  od_graph_def = tf.GraphDef()

  with tf.gfile.GFile(PATH_TO_CKPT, 'rb') as fid:

    serialized_graph = fid.read()

    od_graph_def.ParseFromString(serialized_graph)

    tf.import_graph_def(od_graph_def, name='')
```

Далі ми збираємося завантажити всі мітки

```
label_map = label_map_util.load_labelmap(PATH_TO_LABELS)

categories = label_map_util.convert_label_map_to_categories(label_map,
max_num_classes=NUM_CLASSES, use_display_name=True)

category_index = label_map_util.create_category_index(categories)
```

Тепер ми перетворимо дані зображень на масив *numPy* для обробки.

```
def load_image_into_numpy_array(image):

  (im_width, im_height) = image.size

  return np.array(image.getdata()).reshape(

      (im_height, im_width, 3)).astype(np.uint8)
```

Шлях до зображень для цілей тестування визначається тут. У нас є угода про імена «*image* [*i*]» для *i* (від 1 до *n* + 1), де *n* – кількість наданих зображень.

```
PATH_TO_TEST_IMAGES_DIR = 'test_images'

TEST_IMAGE_PATHS = [ os.path.join(PATH_TO_TEST_IMAGES_DIR,
'image{}.jpg'.format(i)) for i in range(1, 8) ]
```

Цей код виконує висновок для одного зображення, де він виявляє об’єкти, створює блоки та надає клас та оцінку класу цього конкретного об’єкта.

```
def run_inference_for_single_image(image, graph):

  with graph.as_default():

    with tf.Session() as sess:

    # Get handles to input and output tensors
```

```
      ops = tf.get_default_graph().get_operations()

      all_tensor_names = {output.name for op in ops for output in
op.outputs}

      tensor_dict = {}

      for key in [

          'num_detections', 'detection_boxes', 'detection_scores',

          'detection_classes', 'detection_masks'

     ]:

        tensor_name = key + ':0'

        if tensor_name in all_tensor_names:

          tensor_dict[key] = tf.get_default_graph().get_tensor_by_name(

            tensor_name)

      if 'detection_masks' in tensor_dict:

        # The following processing is only for single image

        detection_boxes = tf.squeeze(tensor_dict['detection_boxes'], [0])

        detection_masks = tf.squeeze(tensor_dict['detection_masks'], [0])

        # Reframe is required to translate mask from box coordinates to
image coordinates and fit the image size.

        real_num_detection = tf.cast(tensor_dict['num_detections'][0],
tf.int32)

        detection_boxes = tf.slice(detection_boxes, [0, 0],
[real_num_detection, -1])

        detection_masks = tf.slice(detection_masks, [0, 0, 0],
[real_num_detection, -1, -1])

        detection_masks_reframed =
utils_ops.reframe_box_masks_to_image_masks(

            detection_masks, detection_boxes, image.shape[0],
image.shape[1])

        detection_masks_reframed = tf.cast(

            tf.greater(detection_masks_reframed, 0.5), tf.uint8)

        # Follow the convention by adding back the batch dimension

        tensor_dict['detection_masks'] = tf.expand_dims(

            detection_masks_reframed, 0)
```

```
        image_tensor =
tf.get_default_graph().get_tensor_by_name('image_tensor:0')



        # Run inference

        output_dict = sess.run(tensor_dict,

            feed_dict={image_tensor: np.expand_dims(image, 0)})



        # all outputs are float32 numpy arrays, so convert types as
appropriate

        output_dict['num_detections'] =
int(output_dict['num_detections'][0])

        output_dict['detection_classes'] = output_dict[

          'detection_classes'][0].astype(np.uint8)

        output_dict['detection_boxes'] = output_dict['detection_boxes'][0]

        output_dict['detection_scores'] = output_dict['detection_scores'][0]

        if 'detection_masks' in output_dict:

          output_dict['detection_masks'] = output_dict['detection_masks'][0]

return output_dict
```

Наш останній цикл, який буде викликати всі функції, визначені вище, буде запускати висновок по всіх вхідних зображень по одному, що забезпечить нам виведення зображень, в яких об’єкти виявлені з мітками, та відсоток/оцінка цього об’єкта, що є аналогічно до тренувальних даних.

```
for image_path in TEST_IMAGE_PATHS:

  image = Image.open(image_path)

  # the array based representation of the image will be used later in order
to prepare the

  # result image with boxes and labels on it.

  image_np = load_image_into_numpy_array(image)

  # Expand dimensions since the model expects images to have shape: [1,
None, None, 3]

  image_np_expanded = np.expand_dims(image_np, axis=0)
```

```
  # Actual detection.
  output_dict = run_inference_for_single_image(image_np, detection_graph)
  # Visualization of the results of a detection.
  vis_util.visualize_boxes_and_labels_on_image_array(
      image_np,
      output_dict['detection_boxes'],
      output_dict['detection_classes'],
      output_dict['detection_scores'],
      category_index,
      instance_masks=output_dict.get('detection_masks'),
      use_normalized_coordinates=True,
      line_thickness=8)
plt.figure(figsize=IMAGE_SIZE)
plt.imshow(image_np)
```

Тепер давайте подивимося, як ми можемо виявляти об’єкти у прямому ефірі відео.

Виявлення живих об’єктів за допомогою Tensorflow

Для цієї демонстрації ми використовуватимемо той самий код, але трохи підправимо. Ми збираємось використовувати *OpenCV*

та модуль камери, щоб використовувати пряму трансляцію веб-камери для виявлення об’єктів.

Додайте бібліотеку *OpenCV* та камеру, яка використовується для захоплення зображень. Просто додайте наступні рядки до бібліотеки імпорту.

```
import cv2

cap = cv2.VideoCapture(0)
```

Нам не потрібно завантажувати зображення з каталогу та конвертувати його в масив *numPy*, оскільки *OpenCV* подбає про це за нас. Лиш замініть це:

```
for image_path in TEST_IMAGE_PATHS:

image = Image.open(image_path)

# the array based representation of the image will be used later in order
to prepare the

# result image with boxes and labels on it.

image_np = load_image_into_numpy_array(image)
```

Даним кодом:

```
while True:

ret, image_np = cap.read()
```

Ми не будемо використовувати *matplotlib* для остаточного показу зображення. Натомість ми будемо використовувати *OpenCV* і для цього. Тому видаляємо:

```
cv2.imshow('object detection', cv2.resize(image_np, (800,600)))

if cv2.waitKey(25) & 0xFF == ord('q'):

  cv2.destroyAllWindows()

  break
```

Цей код використовуватиме *OpenCV*, який, у свою чергу, використовуватиме об’єкт камери, ініціалізований раніше, щоб

відкрити нове вікно з ім'ям *Object_Detection* розміром 800 × 600. Камера показуватиме зображення протягом 25 мілісекунд, інакше вона закриє вікно.

Остаточний код з усіма змінами:

```
import numpy as np

import os

import six.moves.urllib as urllib

import sys

import tarfile

import tensorflow as tf

import zipfile

from collections import defaultdict

from io import StringIO

from matplotlib import pyplot as plt

from PIL import Image

import cv2

cap = cv2.VideoCapture(0)

sys.path.append("..")

from utils import label_map_util

from utils import visualization_utils as vis_util


MODEL_NAME = 'ssd_mobilenet_v1_coco_11_06_2017'

MODEL_FILE = MODEL_NAME + '.tar.gz'

DOWNLOAD_BASE = 'http://download.tensorflow.org/models/object_detection/'

# Path to frozen detection graph. This is the actual model that is used for
the object detection.

PATH_TO_CKPT = MODEL_NAME + '/frozen_inference_graph.pb'

# List of the strings that is used to add correct label for each box.

PATH_TO_LABELS = os.path.join('data', 'mscoco_label_map.pbtxt')

NUM_CLASSES = 90
```

```
opener = urllib.request.URLopener()

opener.retrieve(DOWNLOAD_BASE + MODEL_FILE, MODEL_FILE)

tar_file = tarfile.open(MODEL_FILE)

for file in tar_file.getmembers():

  file_name = os.path.basename(file.name)

  if 'frozen_inference_graph.pb' in file_name:

    tar_file.extract(file, os.getcwd())

detection_graph = tf.Graph()

with detection_graph.as_default():

  od_graph_def = tf.GraphDef()

  with tf.gfile.GFile(PATH_TO_CKPT, 'rb') as fid:

    serialized_graph = fid.read()

    od_graph_def.ParseFromString(serialized_graph)

    tf.import_graph_def(od_graph_def, name='')

label_map = label_map_util.load_labelmap(PATH_TO_LABELS)

categories = label_map_util.convert_label_map_to_categories(label_map,
max_num_classes=NUM_CLASSES, use_display_name=True)

category_index = label_map_util.create_category_index(categories)

with detection_graph.as_default():

  with tf.Session(graph=detection_graph) as sess:

    while True:

    ret, image_np = cap.read()

    # Expand dimensions since the model expects images to have shape: [1,
None, None, 3]

    image_np_expanded = np.expand_dims(image_np, axis=0)

    image_tensor = detection_graph.get_tensor_by_name('image_tensor:0')

    # Each box represents a part of the image where a particular object was
detected.

    boxes = detection_graph.get_tensor_by_name('detection_boxes:0')

    # Each score represent how level of confidence for each of the objects.

    # Score is shown on the result image, together with the class label.
```

```
    scores = detection_graph.get_tensor_by_name('detection_scores:0')
    classes = detection_graph.get_tensor_by_name('detection_classes:0')
    num_detections = detection_graph.get_tensor_by_name('num_detections:0')
    # Actual detection.
    (boxes, scores, classes, num_detections) = sess.run(
      [boxes, scores, classes, num_detections],
      feed_dict={image_tensor: image_np_expanded})
    # Visualization of the results of a detection.
    vis_util.visualize_boxes_and_labels_on_image_array(
        image_np,
        np.squeeze(boxes),
        np.squeeze(classes).astype(np.int32),
        np.squeeze(scores),
        category_index,
        use_normalized_coordinates=True,
        line_thickness=8)
    cv2.imshow('object detection', cv2.resize(image_np, (800,600)))
    if cv2.waitKey(25) 0xFF == ord('q'):
      cv2.destroyAllWindows()
      break
```

13.4. Контрольні запитання

1. Наведіть приклади програм виявлення об’єктів.
2. Для чого використовується *Protobuf*?
3. Що таке *COCO*?
4. Яка функція проводить перетворення даних зображення на масив *numPy* для обробки?
5. Опишіть зміни, щ обули введені у програмі для виявлення об’єктів у прямому ефірі.

Список використаних джерел:

1. Нейронні мережі: теорія та практика : навч. посіб. / С. О. Субботін. Житомир : Вид. О. О. Євенок, 2020. 184 с.
2. Машинне навчання : навчальний посібник. Львів : Видавництво «Новий Світ - 2000», 2021. 315 с.
3. Artificial Intelligence: A Modern Approach / Stuart Russell and Peter Norvig. Upper Saddle River, New Jersey 07458, 2010. https://people.engr.tamu.edu/guni/csce421/files/AI_Russell_Norvig.pdf
4. MIT Deep Learning Book in PDF format (complete and parts) by Ian Goodfellow, Yoshua Bengio and Aaron Courville. https://github.com/janishar/mit-deep-learning-book-pdf
5. Neural Networks and Deep Learning by Michael Nielsen. https://github.com/antonvladyka/neuralnetworksanddeeplearning.com.pdf
6. Reinforcement Learning: An Introduction" by Richard S. Sutton and Andrew G. Barto. https://web.stanford.edu/class/psych209/Readings/SuttonBartoIPRLBook2ndEd.pdf
7. Computer Vision: Algorithms and Applications by Richard Szeliski. https://szeliski.org/Book/