

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

МЕТОДИЧНІ ВКАЗІВКИ

до лабораторних робіт з дисципліни «Основи машинного навчання мовою Python»

для магістрів спеціальності 123 Комп'ютерна інженерія
денної форми навчання

Частина I

2024

Методичні вказівки до лабораторних робіт з дисципліни «Основи машинного навчання мовою Python» для магістрів спеціальності 123 Комп'ютерна інженерія, денної форми навчання. Частина I / Укл.: М.Б. Ільяшенко, Н.А. Куликовська – Запоріжжя: НУ «Запорізька політехніка», 2024. – 39 с.

Укладач: М.Б. Ільяшенко, к.т.н., доцент
Н.А. Куликовська, старший викладач

Рецензент: М.Ю. Тягунова, к.т.н., доцент

Відповідальний
за випуск: М.Б. Ільяшенко, к.т.н., доцент

Затверджено:
На засіданні кафедри
«Комп'ютерні системи та мережі»
Протокол № 7
від 22 березня 2024р.

Рекомендовано до видання
НМК факультету КНТ
Протокол № 8
від 27 березня 2024 р

ЗМІСТ

1	ЛАБОРАТОРНА РОБОТА №1 ОГЛЯДОВИЙ АНАЛІЗ ДАНИХ	5
1.1	Теоретичні відомості	5
1.1.1	Створення датафрейму	5
1.1.2	Базові структури даних: серії та датафрейми	6
1.1.3	Імпорт даних	7
1.1.4	Вибір стовпців	7
1.1.5	Огляд завантажених даних	8
1.1.6	Імпорт та індексація дат, вибір рядків.....	8
1.1.7	Вставлення стовпців.....	10
1.1.8	Вставлення рядків	10
1.1.10	Видалення стовпців і рядків.....	11
1.1.11	Вибірка і фільтрація	12
1.1.12	Вибір даних з iloc	12
1.1.13	Вибір даних з loc	13
1.1.14	Спосіб вибору loc за значенням міток.....	13
1.1.15	Логічне індексування в loc	14
1.1.16	Збереження таблиць датафрейму.....	15
1.2	Хід роботи.....	16
1.3	Зміст звіту	17
1.4	Контрольні питання	17
2	ЛАБОРАТОРНА РОБОТА №2 МОДЕЛЬ КЛАСИФІКАЦІЇ.....	19
2.1	Теоретичні відомості	19
2.2	Інструменти та методи розвідувального аналізу даних (EDA) .	20
2.2.1	Візуалізація даних	20
2.2.2	Зведені статистики та заходи центральної тенденції.....	23
2.2.3	Кореляційний аналіз.....	23
2.2.4	Перетворення даних	25
2.2.5	Аналіз викидів та аномалій	26
2.3	Кроки розвідувального аналізу даних	27
2.3.1	Завантаження та первинний огляд даних.....	27
2.3.2	Обробка пропущених значень.....	28
2.3.3	Аналіз розподілу змінних	28
2.3.4	Дослідження кореляцій між змінними	29
2.3.5	Виявлення викидів та аномалій	29
2.3.6	Вивчення категоріальних змінних	29

2.3.7 Візуалізація результатів EDA.....	30
2.4 Побудова та оцінка моделей класифікації.....	30
2.5 Хід роботи.....	33
2.6 Зміст звіту	34
2.7 Контрольні питання	34
ЛІТЕРАТУРА.....	36
ДОДАТОК А Інструкції з роботи з інструментами	37
ДОДАТОК Б Типові помилки мови python.....	38
ДОДАТОК В Пріоритет операторів.....	39

1 ЛАБОРАТОРНА РОБОТА №1

ОГЛЯДОВИЙ АНАЛІЗ ДАНИХ

Мета роботи: необхідно провести оглядовий аналіз даних (exploratory data analysis) з використанням датасету, що описує різні марки автомобілів, що випускалися в США.

1.1 Теоретичні відомості

Бібліотека Pandas є однією з ключових бібліотек [1, 2, 3], яка дозволяє зробити повноцінний процес аналізу даних, а саме:

- завантаження даних. Бібліотека Pandas дозволяє завантажувати дані різних форматів. Бібліотека підтримує завантаження текстових файлів, бінарних, а також є можливість підключення до баз даних і роботи з ними напяму;

- подання даних. Pandas оперує двома основними структурами даних: Series і DataFrame. Series – індексований масив деякого типу: числовий, бінарний або категоріальний. DataFrame – двовимірна структура даних (сукупність Series) або – це таблиця;

- обробка даних. Після завантаження та формування Series або DataFrame виконання фільтрації, індексування, об'єднання різних DataFrame один з одним, є можливість написання власних обчислювальних методів і застосовування їх на DataFrame;

- побудова графіків. За обраними даними з DataFrame можна будувати гістограми значень, можна подивитися як значення в даному стовпці змінюється в часі. Візуалізація можлива завдяки інтеграції Pandas з іншою бібліотекою – Matplotlib. Загалом Pandas інтегрується й з такими бібліотеками, як NumPy або SciPy.

1.1.1 Створення датафрейму

Приклад створення базового об'єкта Pandas — датафрейм зі словників і списків:

```
>>> months = ['jan', 'feb', 'mar', 'apr']
>>> sales = {
...     'revenue':      [100, 200, 300, 400],
...     'items_sold':  [23, 43, 55, 65],
...     'new_clients': [10, 20, 30, 40]
... }
```

```
>>> sales_df = pd.DataFrame(data=sales, index=months)
>>> sales_df
```

	revenue	items_sold	new_clients
jan	100	23	10
feb	200	43	20
mar	300	55	30
apr	400	65	40

1.1.2 Базові структури даних: серії та датафрейми

Серії — одномірні масиви даних. Вони дуже схожі на списки, але відрізняються поведінкою — наприклад, операції застосовуються до списку цілком, а в серіях — поелементно.

Тобто, якщо список помножити на 2, отримаєте той же список, повторений двічі:

```
>>> vector = [1, 2, 3]
>>> vector * 2 [1, 2, 3, 1, 2, 3]
```

А якщо помножити серію, її довжина не зміниться, а ось елементи подвоються.

```
>>> series = pd.Series([1, 2, 3])
>>> series * 2
```

0	2
1	4
2	6

```
dtype: int64
```

Зверніть увагу на перший стовпчик виведення. Це індекс, в якому зберігаються адреси кожного елемента серії. Кожен елемент потім можна отримувати, звернувшись за потрібною адресою:

```
>>> series = pd.Series(['foo', 'bar'])
>>> series[0] 'foo'
```

Ще одна відмінність серій від списків — у якості індексів можна використовувати довільні значення:

```
>>> months = ['jan', 'feb', 'mar', 'apr']
>>> sales = [100, 200, 300, 400]
>>> data = pd.Series(data=sales, index=months)
>>> data
```

jan	100
feb	200
mar	300
apr	400

```
dtype: int64
```

Приклад. як отримувати значення кожного місяця:

```
>>> data['feb']
200
```

У підсумку: серії — це одномірні масиви, датафрейми — це масиви.

1.1.3 Імпорт даних

Приклад створення датафрейму:

```
>>> data = pd.read_csv('datafile.csv',
sep=';', decimal=',')
```

За замовчанням роздільники CSV-файлу: між значеннями (стовпцями) — кома , і десятковий — крапка

Приклад, як перегланути три перших і два останніх рядки:

```
>>> data.head(3)
>>> data.tail(2)
```

що у підсумку матиме такий вигляд:

```
>>> data.head(3)
```

	DateTime	Category	ValueInt	ValueDb1
0	12.06.2021	First	100	21.512670
1	13.06.2021	Second	200	58.209348
2	14.06.2021	First	500	88.335696

```
>>> data.tail(3)
```

	DateTime	Category	ValueInt	ValueDb1
19	01.07.2021	Second	4050	29.839563
20	02.07.2021	First	4260	99.097263

У разі потреби дані можна імпортувати з Excel-таблиць:

```
pd.read_excel('datafile.xlsx', sheet_name='Аркуш1')
```

1.1.4 Вибір стовпців

Для виведення стовпця потрібно укласти в квадратні дужки його назву і поставити після назви датафрейму:

```
>>> data['Category']
0      First
1      Second
...
19     Second
20      First
```

Name: Category, dtype: object

Зверніть увагу, що результат команди — це новий датафрейм з тим самим індексом.

Якщо потрібно вивести декілька стовпців, в квадратні дужки потрібно вставити список з їхніми назвами: `data[['Category', 'ValueInt']]`. **Зверніть увагу:** квадратні дужки стали подвійними. Перші - від датафрейма, другі - від списку:

1.1.5 Огляд завантажених даних

Приклад завантажених даних з допомогою наступних методів:

- `.shape` - кількість в датафрейме рядків і колонок;
- `.columns, .dtypes, .info()` - яка інформація і якого типу міститься в кожному стовпчику;
- `.index` - назви рядків таблиці;
- `.describe()` - основні статистичні параметри кожного стовпця.

Наприклад:

```
>>> data.shape (21, 4)
```

```
>>> data.dtypes
DateTime      object
Category      object
ValueInt      int64
ValueDbl      float64
dtype: object
```

```
>>> data.describe()
          ValueInt  ValueDbl
count      21.00000  21.000000
mean      2160.00000  58.092186
std       1303.15003  27.785915
min        100.00000   1.970310
25%       1110.00000  33.708401
50%       2160.00000  63.030305
75%       3210.00000  84.424289
max       4260.00000  99.097263
```

1.1.6 Імпорт та індексація дат, вибір рядків

Приклад імпорту дат, що зберігаються у CSV-файлі у формі текстових рядків.

Наведено найпростіший приклад, коли дата представлена як-то “ден.місяць.рік” або, наприклад, “12.06.2021”:

```
>>> data = pd.read_csv('datafile.csv', sep=';',
decimal=',',
                        parse_dates=['DateTime'],
                        dayfirst=True)
```

У цьому прикладі це стовпець DateTime. Наступний аргумент dayfirst відзначає, що перша група цифр в рядку дат це дні.

Приклад дати з точним форматом запису. Наприклад:

```
>>> import pandas as pd
>>> from datetime import datetime
>>> custom_date_parser = lambda x: datetime.strptime(x,
'%d.%m.%Y')
>>> data = pd.read_csv('datafile.csv',
sep=';', decimal=',',
                        index_col='DateTime',
                        parse_dates=['DateTime'],
                        date_parser=custom_date_parser)
```

В даному випадку аргумент date_parser визначає власну функцію custom_date_parser для форматного перетворення рядка у числовий тип дати

Іншим важливим аргументом функції зчитування CSV-файлу read_csv у наведеному прикладі є index_col. Цей аргумент вказує на індексний стовпець (або стовпці), які слід використовувати як мітки рядків датафрейму.

В результаті такого визначення індексного стовпця з’являється ефективна можливість вибору одного або декількох рядків з таблиці датафрейму. Наприклад:

```
>>> data['2021-7']
  Category  ValueInt  ValueDb1  DateTime
2021-07-01   Second    4050  29.839563
2021-07-02   First    4260  99.097263
```

Інший спосіб використовувати frame.loc[string]. Наприклад:

```
>>> show_these_dates = ['2021-06-12', '2021-06-
16', '2021-07-01']
>>> data.loc[show_these_dates]
  Category  ValueInt  ValueDb1  DateTime
2021-06-12   First     100  21.512670
2021-06-16   First     900  73.331767
2021-07-01   Second   4050  29.839563
```

1.1.7 Вставлення стовпців

Новий стовпець можна вставити за допомогою функції `insert`:

```
>>> a_dataframe = pd.DataFrame({"A": [1, 2, 3], "C":
[7, 8, 9]})
>>> a_dataframe.insert(loc=1, column="B", value=[4, 5,
6])
>>> a_dataframe
```

	A	B	C
0	1	4	7
1	2	5	8
2	3	6	9

Аргументи функції `insert` мають таке призначення:

- `loc` визначає індекс колонки, куди буде вставлена колонка (пам'ятаємо, що індекси у Python починаються з 0, тобто 1 - це друга колонка);
- `column` визначає назву нового стовпця - "B";
- `value` передає значення нового стовпця у типі Series або списку.

1.1.8 Вставлення рядків

Приклад для вставлення рядка в кінці таблиці виглядає таким чином:

```
>>> df = pd.DataFrame([[1, 2], [3, 4]],
columns=['A', 'B'], index=['a', 'b'])
>>> df
```

	A	B
a	1	2
b	3	4

```
>>> df2 = pd.DataFrame([[5, 6], [7, 8]],
columns=['A', 'B'], index=['c', 'd'])
>>> df2
```

	A	B
c	5	6
d	7	8

```
>>> df.append(df2)
```

	A	B
a	1	2
b	3	4
c	5	6
d	7	8

1.1.10 Видалення стовпців і рядків

Для видалення рядків та стовпців з датафрейму Pandas використовує функцію `drop`.

```
# Видалити стовпець "Area" з датафрейму
data = data.drop("Area", axis=1)
```

```
# альтернативно, стовпець можна видалити,
використовуючи
# аргумент "columns" функції "drop"
data = data.drop(columns="area")
```

```
# Видалити стовпець "Area" з датафрейму в тому ж
об'єкті
# з допомогою значення аргументу "inplace=True"
data.drop("Area", axis=1, inplace=True).
```

```
# Видалити декілька стовпців з таблиці
data = data.drop(["Y2001", "Y2002", "Y2003"], axis=1)
```

Рядки можна видалити з допомогою функції `drop`, вказавши `axis = 0`. `Drop()` видаляє рядки на основі “міток”, а не числового індексування. Тому, щоб видалити рядки на основі їхнього числового положення / індексу, використовують метод `iloc`.

```
# Видалити рядки з мітками 0,1,5
data = data.drop([0,1,5], axis=0)
```

```
# Видалити всі рядки з міткою "Ireland"
# Для цього спочатку встановіть відповідний індекс для
датафрейму
data = data.set_index("Country")
data = data.drop("Ireland", axis=0)
```

```
# Видалити перші 5 рядків таблиці з допомогою метода
"iloc"
data = data.iloc[5:,]
```

1.1.11 Вибірка і фільтрація

Для подальшого прикладу треба скористатися набором даних uk-500 з www.briandunning.com [2]. Ці дані містять вигадані імена, адреси, компанії та номери телефонів у Великобританії.

```
>>> import pandas as pd
>>> import random

>>> # зчитування CSV-файлу
>>> data = pd.read_csv('https://s3-eu-west-
1.amazonaws.com/shanebucket/downloads/uk-500.csv')
>>> # встановити значення 'id' як індексу в прикладах
>>> data['id'] = [random.randint(0,1000) for x in
range(data.shape[0])]

>>> data.head(5)

first_name last_name ...      web      id
0 Aleshia Tomkiewicz ...
http://www.alandrosenburgcpapc.co.uk 136
1 Evan Zigomalas ...
http://www.capgeminiamerica.co.uk 829
2 France Andrade ...
http://www.elliottjohnwesq.co.uk 483
3 Ulysses Mcwalters ...
http://www.mcmahanbenl.co.uk 231
4 Tyisha Veness ...
http://www.champagneroom.co.uk 873
```

```
[5 rows x 12 columns]
```

Ефективним і характерним для Python є використання конструкції `for` у квадратних дужках.

1.1.12 Вибір даних з `iloc`

Індикатор `iloc` використовується для датафреймів Pandas на основі цілочисельного позиціонування або вибору за положенням.

Синтаксис індикатора такий — `data.iloc[<номери рядків>, <номери стовпців>]`. Кожен рядок має номер рядка від 0 до загальної кількості рядків `data.shape[0]`. Так само, це стосується номерів стовпців - від 0 до `data.shape[1]`.

Вибір одного рядка або стовпця виглядає так:

```
# Рядки:
data.iloc[0] # 1-й рядок датафрейму - зауважте,
повертається тип даних Series
data.iloc[1] # 2-й рядок датафрейму
data.iloc[-1] # останній рядок датафрейму
# Стовпці:
data.iloc[:,0] # 1-й стовпець 'first_name'
data.iloc[:,1] # 2-й стовпець 'last_name'
data.iloc[:, -1] # останній стовпець - 'id'
    вибрати кілька стовпців і рядків разом з допомогою
індексатора iloc:
data.iloc[0:5] # перші п'ять рядків датафрейму 0, 1, 2,
3, 4
data.iloc[:, 0:2] # перші два стовпці з усіма рядками
data.iloc[[0,3,6,24], [0,5,6]] # рядки 1, 4, 7, 25 і
стовпці 1, 6, 7
```

Зверніть увагу, якщо обрано один рядок або стовпець, то тип повернутих даних — Series, в іншому випадку — DataFrame. Щоб запобігти несподіваній зміні типу, аргумент номеру навіть одного рядка або стовпця треба передавати як список — тобто у квадратних дужках:

```
>>> print(type(data.iloc[100])) # 1 рядок заданий
числом - 100, тому повертається тип 'Series'
>>> print(type(data.iloc[[100]])) # 1 рядок заданий
списком - [100], тому повертається тип 'DataFrame'
```

```
<class 'pandas.core.series.Series'>
<class 'pandas.core.frame.DataFrame'>
```

1.1.13 Вибір даних з loc

За допомогою індексатора loc вибір рядків датафрейму можна здійснювати у два різних способи:

1. за значеннями міток;
2. за логічним (умовним) пошуком.

Синтаксис застосування індексатору loc той самий, що і для iloc: data.loc[<виділення рядків>, <виділення стовпців>].

1.1.14 Спосіб вибору loc за значенням міток

Вибір з допомогою loc базується на індексі датафрейму (якщо такий встановлений). Якщо ні, то індекс встановлюється у датафреймі

з допомогою `df.set_index()`. Наприклад, встановивши індекс тестового датафрейму на `last_name`:

```
data.set_index("last_name", inplace=True)
```

Далі зі встановленим у датафреймі індексом треба безпосередньо вибирати рядки для різних значень `last_name` для одного або для декількох значень. Наприклад:

```
data.loc['Andrade']
data.loc[['Andrade', 'Veness']]
```

також, треба обрати певні стовпці для виведення:

```
data.loc[['Andrade', 'Veness'], ['first_name',
'address', 'city']]
```

Можна обрати діапазони значень рядків та/або стовпців:

```
data.loc['Andrade':'Veness', ['first_name', 'address',
'city']]
data.loc[['Andrade', 'Veness'], 'city':'email']
```

1.1.15 Логічне індексування в loc

Умовний вибір з допомогою логічних масивів `data.loc[<умова вибору>]` - це найпотужніший метод використання Pandas в роботі з датафреймами. Завдяки булевому індексуванню або логічному вибору ви передаєте масив або ряд `True / False` значень індексатору `loc`, щоб вибрати ті рядки, де елементи логічного ряду мають значення `True`.

Наприклад, вираз `data['first_name'] == 'Antonio'` продукує ряд `True / False` значень, де, значення `True` відповідають умові відповідності елемента стовпця `first_name` значенню `'Antonio'`:

```
data.loc[data['first_name'] == 'Antonio']
```

Другий аргумент може бути переданий до `loc` для вибору конкретних стовпців датафрейму. В даному випадку, стовпці можуть або називатися іменем або бути окремим списком стовпців або визначатись як діапазон з допомогою оператора `:`.

```
data.loc[data['first_name'] == 'Erasmus',
['company_name', 'email', 'phone1']]
```

Приклад логічним виразів:

```
# Виберіть рядки з іменем 'Antonio' та всі стовпці
# між 'city' і 'email'
data.loc[data['first_name'] == 'Antonio',
'city':'email']

# Виберіть рядки, де стовпець 'email' закінчується
```

```

# на 'hotmail.com', включає усі стовпці
data.loc[data['email'].str.endswith('hotmail.com')]

# Виберіть рядки з 'first_name', що дорівнюють певним
# значенням + усі стовпці
data.loc[data['first_name'].isin(['France', 'Tyisha',
'Eric'])]

# Виберіть рядки з е-поштою 'gmail.com' та іменем
'Antonio'
data.loc[data['email'].str.endswith("gmail.com") &
(data['first_name'] == 'Antonio')]

# виділіть рядки зі стовпцем 'id' між 100 і 200 і
просто
# поверніть стовпці 'postal' і 'web'
data.loc[(data['id'] > 100) & (data['id'] <= 200),
['postal', 'web']]

# Також може бути використана лямбда-функція, що
повертає
# значення True / False. Виділіть рядки, де назва
компанії
# містить 4 слова.
data.loc[data['company_name'].apply(lambda x:
len(x.split(' ')) == 4)]

# Вибір можна зробити за межами основного .loc:
# Створіть окрему змінну 'idx' з вашим вибором:
idx = data['company_name'].apply(lambda x:
len(x.split(' ')) == 4)
# Виберіть лише значення True у 'idx' і лише 3
зазначені стовпці:
data.loc[idx, ['email', 'first_name', 'company_name']]

```

1.1.16 Збереження таблиць датафрейму

Після маніпуляцій або обчислень, завершальним кроком є збереження даних у CSV. Для цього є дві функції `to_csv`, щоб записати датафрейм у файл CSV, та `to_excel`, щоб записати датафрейм у файл Excel.

Зазвичай номери рядків не потрібні у вихідному файлі,

```
# отже index = False. Щоб уникнути проблем із
символами,
# зазвичай використовуємо кодування utf8.

data.to_csv("output_filename.csv", index=False,
encoding='utf8')

# Щоб вихідний результат Excel працював, можливо, вам
можливо
# доведеться встановити пакет "xlsxwriter".

data.to_csv("output_excel_file.xlsx", sheet_name="Аркуш
1", index=False)
```

1.2Хід роботи

Завдання. Необхідно реалізувати наступні етапи:

1. Завантажити дані в pandas.DataFrame.
2. Вивести загальну статистику за датасетом та за кожною з колонок.
3. Прибрати не інформативні колонки (пояснити чому ці колонки було обрано).
4. Переіменувати колонки в нижньому регістрі без пропусків.
5. Видалити пропущені або null-значення або заповнити їх відповідними значеннями.
6. Видалити дублікати рядків.
7. Визначити та видалити аутлаєрів
8. Провести розгорнуту візуалізацію всієї значущої інформації в даних (визначити якісь колонки або їх комбінації або похідні значення які з них отримані і як наочно візуалізувати, пояснити що демонструється кожним з графіків)
9. Зробити висновки (текстом) за результатами оглядового аналізу даних (що вдалося дізнатися, які явні закономірності чи залежності простежуються, що у даних привертає увагу)

Аналітика даних це слабо формалізований етап, багато в чому залежить від досвіду аналітика та цілей, які перед ним стоять. Тому аналіз даних можна не обмежувати пунктами вище, а запропонувати свої етапи передобробки, аналізу та/або візуалізації, що застосовуються в датасету в завданні.

Для виконання завдання використати датасет, що описує різні марки автомобілів, що випускалися в США.

Значення полів:

Make - виробник

Model – модель

Year - рік виробництва

Engine Fuel Type – тип палива

Engine HP - потужність двигуна в к.с.

Engine Cylinders - кількість циліндрів двигуна

Transmission Type – тип трансмісії

Driven_Wheels - кількість провідних коліс

Number of Doors - кількість дверей

Market Category – категорія на ринку

Vehicle Size - розмір машини

Vehicle Style - стиль машини

highway MPG - споживання палива на трасі

city mpg - споживання палива у місті

Popularity – популярність

MSRP (manufacturer's suggested retail price) - рекомендована виробником роздрібна ціна

Результатом роботи має бути Jupyter Notebook із коментарями по ходу виконання роботи.

1.3 Міст звіту

Звіт повинен мати такі складові:

- титульний лист;
- мета роботи;
- результат виконання завдань (програмний код та скріншоти);
- відповіді на контрольні питання.

1.4 Контрольні питання

1. Опишіть, як бібліотеку Pandas можна використовувати для аналізу даних у Python?

2. Поясніть різницю між серією та DataFrame у Pandas.

3. Як ви можете завантажити дані з файлу CSV у DataFrame у Pandas?
4. Які методи можна використовувати для перевірки кількох перших і кількох останніх рядків DataFrame?
5. Опишіть важливість і методи обробки відсутніх значень у наборі даних.
6. Як можна перейменувати стовпці DataFrame у Pandas?
7. Опишіть процес ідентифікації та видалення повторюваних рядків у DataFrame?
8. Як візуалізація даних може сприяти дослідницькому аналізу даних і які бібліотеки зазвичай використовуються для цієї мети в Python?
9. Подумайте про важливість попередньої обробки даних і етапи підготовки даних для аналізу?

2 ЛАБОРАТОРНА РОБОТА №2

МОДЕЛЬ КЛАСИФІКАЦІЇ

Мета роботи: необхідно провести EDA (exploratory data analysis) та побудувати модель класифікації, яка б змогла максимально точно класифікувати пацієнтів із проблемами печінки та без них за результатами аналізів. Провести оптимізацію параметрів моделі. Навести всі метрики якості для моделі класифікації.

2.1 Теоретичні відомості

Цілі розвідувального аналізу даних:

- розуміння структури та характеристик набору даних. На цьому етапі EDA ми уважно вивчаємо структуру та характеристики даних, щоб повністю зрозуміти, з чим маємо справу. Це включає огляд розміру набору даних, типів змінних, наявності пропущених значень, дублікатів та інших важливих аспектів. Розвідка на цьому рівні дозволяє нам сформуванати чітке уявлення про дані, що стає фундаментом подальшого аналізу;

- виявлення аномалій та викидів. Іноді дані можуть зберігати сюрпризи, які можуть суттєво вплинути на наші висновки. Використовуючи методи EDA, ми виявляємо аномалії та викиди в наших даних. Ці значення, що відхиляються від загального патерну, можуть виникати через помилки введення, випадкових подій або навіть вказувати на систематичні проблеми в зборі даних. Визначення та усунення таких аномалій стає критичною частиною нашого аналізу, щоб гарантувати надійність результатів;

- ідентифікація зв'язків та кореляцій між змінними. Використовуючи статистичні заходи, EDA дозволяє ідентифікувати взаємозв'язки між змінними, що допомагає зрозуміти, як одні фактори впливають на інші. Це може стати відправною точкою для формулювання гіпотез та прийняття важливих бізнес-рішень;

- підготовка даних для подальших етапів аналізу. На цьому етапі EDA ми опрацьовуємо дані для того, щоб підготувати їх до більш складних аналітичних методів. Ми чистимо дані від шуму, заповнюємо пропущені значення, проводимо масштабування чи перетворення змінних, щоб забезпечити їхню якісну та інтерпретовану структуру.

Тільки з належним чином підготовленими даними ми можемо досягти точних і надійних результатів на наступних етапах аналізу.

2.2 Інструменти та методи розвідувального аналізу даних (EDA)

2.2.1. Візуалізація даних

Візуалізація даних дозволяє побачити та зрозуміти патерни, тренди та взаємозв'язки у даних через графіку та діаграми.

Гістограма – це графічне уявлення розподілу даних з різних інтервалів. Вона дозволяє оцінити, як часто значення потрапляють у певні діапазони і які ми маємо піки чи провали в даних.

Діаграма розсіювання - це графік, в якому кожна точка є окремим спостереженням і показує взаємозв'язок між двома змінними. Це може допомогти визначити, чи є якась залежність чи кореляція між ними.

Приклад Python-коду для побудови гістограми та діаграми розсіювання:

```
import matplotlib.pyplot as plt
import seaborn as sns

# генеруємо випадкові дані
data = [23, 34, 45, 67, 23, 56, 78, 89, 43, 65, 34, 56, 76]

# побудова гістограми
plt.hist(data, bins=10, color='blue', alpha=0.7)
plt.xlabel('Значення')
plt.ylabel('Частота')
plt.title('Гістограма даних')
plt.show()

# побудова діаграми розсіювання
x = [1, 2, 3, 4, 5, 6, 7, 8]
y = [15, 25, 30, 40, 50, 60, 70, 80]

plt.scatter(x, y, color='red')
plt.xlabel('Зміна X')
plt.ylabel('Зміна Y')
plt.title('Діаграма розсіювання')
plt.show()
```

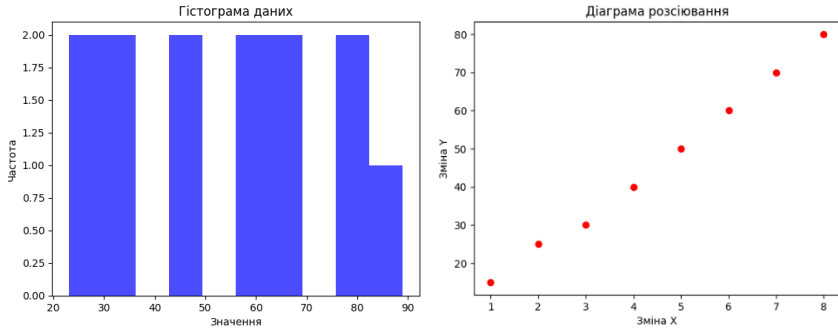


Рисунок 2.1 – Гістограма та діаграма даних

Скринька з вусами (box plot) – це візуалізація статистичних характеристик розподілу даних, таких як медіана, кватилі та викиди. Вона допомагає швидко оцінити розкид та симетрію даних, а також виявити потенційні аномалії.

Приклад Python-коду для побудови ящика з вусами:

```
import matplotlib.pyplot as plt
import seaborn as sns

# Випадкові дані
data = [23, 34, 45, 67, 23, 56, 78, 89, 43, 65, 34, 56,
76]

# Побудова скриньки з вусами
plt.boxplot(data)
plt.ylabel('Значення')
plt.title('box plot')
plt.show()
```

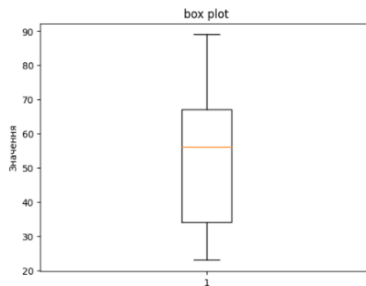


Рисунок 2.2 – Box plot

Теплова карта (heatmap) – це графічне уявлення матриці даних, де колірна шкала показує рівень взаємозв'язку між змінними. Це допомагає виявити патерни та залежності у великих наборах даних.

Приклад Python-коду для побудови теплової карти:

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Випадкові числа
data = np.random.rand(5, 5)

# побудова теплової карти
sns.heatmap(data, annot=True, cmap='YlGnBu')
plt.title('Теплова карта')
plt.show()
```

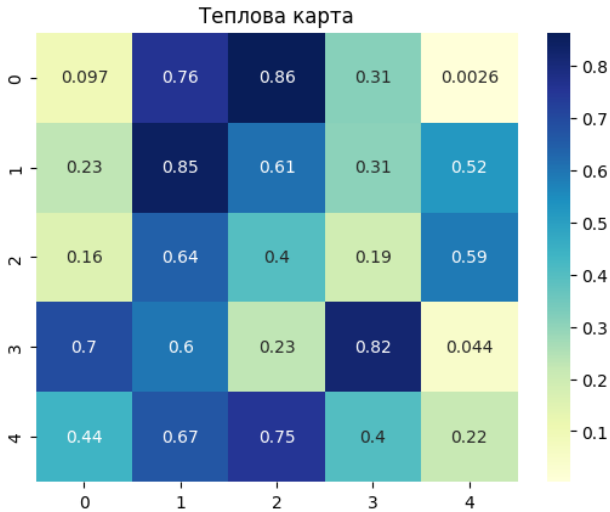


Рисунок 2.3 – Теплова карта

Більше прикладів візуалізації даних можна подивитись за посиланням:

<https://github.com/itsmeale/pandas-plotting-examples>

<https://github.com/grexor/python-plotting-tutorial>

<https://github.com/pb111/Data-Visualization-with-Seaborn>

Візуалізація даних у Python надає потужні інструменти для дослідження, аналізу та візуалізації інформації з набору даних.

2.2.2 Зведені статистики та заходи центральної тенденції

Зведені статистики та заходи центральної тенденції дозволяють отримати узагальнене уявлення про розподіл даних та основні характеристики. Це ключові числові метрики, які допомагають нам зрозуміти типові та найбільш значущі значення у наборі даних:

– Середнє (Mean) : Це сума всіх значень, поділена на кількість значень. Воно представляє загальну "середню" величину даних.

– Медіана (Median) : Це середнє значення двох середніх значень, якщо кількість значень парне, або середнє значення центрального числа, якщо кількість значень непарне. Медіана надає більш стійку міру центральної тенденції у присутності викидів.

– Мода (Mode) : Це значення, яке найчастіше зустрічається в наборі даних. Мода може бути корисною для визначення найбільш типового значення.

Приклад Python-коду для розрахунку середнього, медіани та моди:

```
import numpy as np
from scipy import stats

# Генеруємо випадкові дані
data = np.random.randint(1, 100, 20)

# Розрахунок середнього
mean = np.mean(data)
print("Середнє:", mean)

# Розрахунок медіани
median = np.median(data)
print("Медіана:", median)

# Розрахунок моди
mode = stats.mode(data)
print("Мода:", mode)
```

2.2.3 Кореляційний аналіз

Кореляційний аналіз допомагає нам зрозуміти, які змінні взаємопов'язані між собою та наскільки сильний цей зв'язок.

Коефіцієнт кореляції вимірює ступінь лінійної залежності між двома змінними:

- Позитивна кореляція Якщо одна змінна збільшується, інша також збільшується. Коефіцієнт кореляції знаходиться у діапазоні від 0 до 1.

- Негативна кореляція : Якщо одна змінна збільшується, інша зменшується. Коефіцієнт кореляції знаходиться у діапазоні від 0 до -1.

- Нульова кореляція : Відсутність лінійної залежності між змінними. Коефіцієнт кореляції близький до 0.

Приклад Python-коду для розрахунку коефіцієнта кореляції:

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

# Генеруємо випадкові дані
x = np.array([1, 2, 3, 4, 5, 6])
y = np.array([2, 3, 4, 4, 5, 7])

# Розрахунок коефіцієнта кореляції Пірсона
correlation = np.corrcoef(x, y)[0, 1]
print("Коефіцієнт кореляції:", correlation)

# Візуалізація даних
plt.scatter(x, y)
plt.xlabel('Змінна X')
plt.ylabel('Змінна Y')
plt.title('Діаграма розсіювання та кореляція')
plt.show()
```

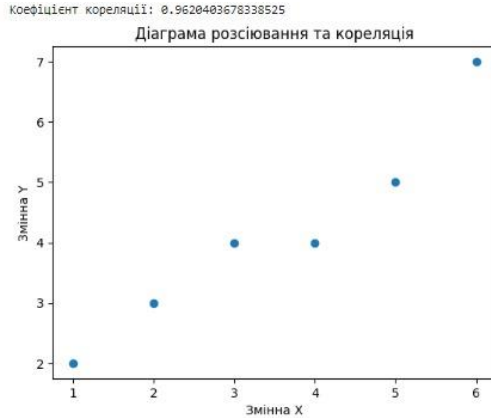


Рисунок 2.4 – Кореляційний аналіз

2.2.4 Перетворення даних

Перетворення даних – це процес зміни шкали або розподілу змінних, щоб зробити їх більш підходящими для аналізу чи моделювання. Це важливий етап EDA, який допомагає згладити відмінності між змінними та створити більш стійкі та інтерпретовані дані:

- Нормалізація (Normalization) : Цей метод масштабує значення змінних так, щоб вони знаходилися в діапазоні від 0 до 1. Це особливо корисно, коли є змінні з різними одиницями вимірювання та масштабами.

- Стандартизація (Standardization) : Цей метод перетворює значення змінних так, щоб їхнє середнє було рівне 0, а стандартне відхилення – 1. Він робить розподіл більш "стандартним" і симетричним.

Приклад Python-коду для нормалізації та стандартизації даних:

```
import numpy as np
from sklearn.preprocessing import MinMaxScaler,
StandardScaler

# Generate random data
data = np.array([[1, 2],
                 [2, 3],
                 [3, 4],
```

```

[4, 5]])

# Data normalization
scaler = MinMaxScaler()
normalized_data = scaler.fit_transform(data)
print("Normalized data:")
print(normalized_data)

# Data standardization
scaler = StandardScaler()
standardized_data = scaler.fit_transform(data)
print("Standardized data:")
print(standardized_data)

```

Нормалізовані дані:

```

[[0.      0.      ]
 [0.33333333 0.33333333]
 [0.66666667 0.66666667]
 [1.      1.      ]]

```

Стандартизовані дані:

```

[[-1.34164079 -1.34164079]
 [-0.4472136  -0.4472136 ]
 [ 0.4472136   0.4472136 ]
 [ 1.34164079  1.34164079]]

```

Рисунок 2.5 – Нормалізація

2.2.5 Аналіз викидів та аномалій

Аналіз викидів та аномалій – це процес виявлення та дослідження значень даних, які істотно відрізняються від інших спостережень. Викиди та аномалії можуть виникнути через помилки даних, випадкових подій або вказувати на особливості досліджуваного явища.

Основні кроки аналізу викидів та аномалій:

- Візуалізація даних : Використовувати графіки, такі як ящик із вусами (box plot) або діаграми розсіювання, щоб візуально виявити потенційні викиди.

- Статистичний аналіз : Використовувати статистичні методи, щоб визначити, які значення вважаються викидами на основі критеріїв, таких як інтерквартильний розмах або Z-оцінка.

- Прийняття рішення : Вирішити, які дії потрібно зробити з викидами, наприклад, видалити їх, замінити на інші значення або залишити без змін.

Приклад Python-коду для аналізу викидів з використанням ящика з вусами:

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

# Генеруємо випадкові дані з викидами
data = np.array([10, 15, 20, 25, 30, 500])

# Побудова ящика з вусами
plt.boxplot(data)
plt.ylabel('Значення')
plt.title('Аналіз викидів')
plt.show()
```



Рисунок 2.6 – Аналіз аномалій

2.3 Кроки розвідувального аналізу даних

2.3.1 Завантаження та первинний огляд даних

Перший крок – завантаження даних та їх первинний огляд. Для цього використовують бібліотеку Pandas, яка дозволяє працювати з табличними даними.

Наприклад, є CSV-файл `sales_data.csv` з даними про продаж. Завантажити дані з файлу та вивести перші п'ять рядків таблиці на екран.

```
import pandas as pd

# Завантаження даних
data = pd.read_csv('sales_data.csv')

# Виведення перших 5 рядків таблиці
print(data.head())
```

2.3.2 Обробка пропущених значень

Пропущені значення можуть вплинути на аналіз, тому важливо їх обробити. У цьому кроці треба заповнити пропущені значення середніми або медіанними значеннями.

Наприклад, є пропущені значення у стовпці Ціна. Вирішити цю проблему заповнивши пропущені значення середньою ціною продуктів.

```
mean_price = data['Ціна'].mean()
data['Ціна'].fillna(mean_price, inplace=True)
```

2.3.3 Аналіз розподілу змінних

У цьому кроці показано, як визначити розподіл числових змінних. Побудувати гістограми та діаграми розсіювання для кращого розуміння даних.

Наприклад, треба дізнатись розподіл цін на продукти. Гістограму для цієї змінної.

```
import matplotlib.pyplot as plt

plt.hist(data['Ціна'], bins=20, color='blue',
alpha=0.7)
plt.xlabel('Ціна')
plt.ylabel('Частота')
plt.title('Розподіл цін на продукти')
plt.show()
```

2.3.4 Дослідження кореляцій між змінними

У цьому кроці розглядається кореляції між числовими змінними. Треба використати коефіцієнт кореляції, щоб визначити, які змінні взаємопов'язані.

Наприклад, треба подивитись чи існує зв'язок між ціною продукту та його кількістю продажів. Розрахунок коефіцієнту кореляції між цими змінними наступний:

```
correlation = data['Ціна'].corr(data['Кількість'])
print("Кореляція між ціною та кількістю продажів:",
correlation)
```

2.3.5 Виявлення викидів та аномалій

У цьому кроці показано, як подивитись викиди та аномалії в даних. Для цього використовується візуалізація ящик з вусами (box plot).

Наприклад, чи є викиди у стовпці Кількість (кількість продажів товару). Побудувати ящик із вусами для цієї змінної.

```
plt.boxplot(data['Кількість'])
plt.ylabel('Кількість')
plt.title('Аналіз викидів у кількості продажів')
plt.show()
```

2.3.6 Вивчення категоріальних змінних

У цьому досліджується розподіл та частоти категоріальних змінних.

Наприклад, які продукти є найпопулярнішими серед покупців. Побудувати графік частоти продажів для кожного продукту.

```
product_counts = data['Продукт'].value_counts()
product_counts.plot(kind='bar')
plt.xlabel('Продукт')
plt.ylabel('Частота продажів')
plt.title('Частота продажів продуктів')
plt.xticks(rotation=45)
plt.show()
```

2.3.7 Візуалізація результатів EDA

Останній крок – візуалізація результатів EDA.

Наприклад, створити узагальнену візуалізацію результатів EDA, включаючи гістограму, діаграму розсіювання та ящик з вусами.

```
plt.figure(figsize=(10, 6))

plt.subplot(2, 2, 1)
plt.hist(data['Ціна'], bins=20, color='blue',
alpha=0.7)
plt.xlabel('Ціна')
plt.ylabel('Частота')
plt.title('Розподіл цін на продукти')

plt.subplot(2, 2, 2)
plt.scatter(data['Ціна'], data['Кількість'],
color='green')
plt.xlabel('Ціна')
plt.ylabel('Кількість')
plt.title('Діаграма розсіювання між ціною та кількістю
продажів')

plt.subplot(2, 2, 3)
plt.boxplot(data['Кількість'])
plt.ylabel('Кількість')
plt.title('Аналіз викидів у кількості продажів')

plt.tight_layout()
plt.show()
```

За допомогою цих кроків показано процес від завантаження та огляду даних до візуалізації результатів аналізу.

2.4 Побудова та оцінка моделей класифікації

Коли дані повністю готові і очищені треба створювати модель. Далі будемо використовувати на прикладі моделі поведінки споживача. Де було відокремлено 13 незалежних факторів, що можуть впливати на залежну метрику, якою є здійснення покупки споживачем. Для початку визначемо в нашій моделі залежну і незалежні змінні:

```
y = filtered['paid']
X = filtered.iloc[:,1:13]
```

Зверніть увагу, що як незалежні змінні беруть з першої по тринадцяту колонку, хоча лічення в Python починається з нуля. Це за тією причиною, що ідентифікатор користувача “user_pseudo_id” був потрібен лише задля того, щоб ідентифікувати юзерів в процесі обробки даних та видаленні дублікатів, а далі ми будемо працювати з даними без нього.

Далі, за допомогою функції `train_test_split` дата-фрейм поділяється на тренувальну та тестову групи: одну для того, щоб навчити модель ідентифікувати покупця від не покупця, а іншу - щоб перевірити на вже обучений моделі наскільки добре вона справляється зі своєю задачею:

```
# Split into 70% training and 30% testing set
X, X_test, y, y_test = train_test_split(features,
                                         targets,
                                         test_size = 0.3,
                                         random_state = 42)
```

30% для тестової групи - це стандартний розмір розподілу, коли працюєш з моделями бінарної класифікації.

Далі був створений словник, в який спочатку були записані назви моделей, а далі будуть записані отриманні результати по тій чи іншій моделі:

```
models = {}
models['Logistic Regression'] =
LogisticRegression(max_iter=5000)
models['Support Vector Machines'] =
LinearSVC(max_iter=5000)
models['Decision Trees'] = DecisionTreeClassifier()
models['Random Forest'] = RandomForestClassifier()
models['Naive Bayes'] = GaussianNB()
models['K-Nearest Neighbor'] = KNeighborsClassifier()
models['Gradient Boosting'] =
GradientBoostingClassifier()
```

Для кожного елемента словника було створено об'єкт з однойменною функцією для розрахунку моделі, а для моделей логістичної регресії та машин опорних векторів ми задаємо одразу параметр максимальної кількості ітерацій в 5000, бо при стандартній настройці вони не зможуть обробити таку кількість даних.

Далі було створено ще п'ять словників з критеріями оцінювання моделі, які в наступному циклі буду записувати в словник з назвою

моделі, це буде словник, в якому значенням виступає також словник з результатами за критеріями оцінювання моделі, з якого ми зробимо дата-фрейм для більш приємного сприйняття результатів зором.

```
true_positive, false_positive, true_negative,
false_negative, accuracy = {}, {}, {}, {}, {}
for key in models.keys():
models[key].fit(X_train, y_train)
predictions = models[key].predict(X_test)
true_negative[key], false_positive[key],
false_negative[key],
true_positive[key] = confusion_matrix(y_test,
predictions).ravel()
accuracy[key] = (true_positive[key]+true_negative[key])
/
(true_positive[key]+false_positive[key]
+true_negative[key]+false_negative[key])
```

```
df_model = pd.DataFrame(index=models.keys(), columns=['True Positive', 'False Positive',
'True Negative', 'False Negative', 'Accuracy'])
df_model['True Positive'] = true_positive.values()
df_model['False Positive'] = false_positive.values()
df_model['True Negative'] = true_negative.values()
df_model['False Negative'] = false_negative.values()
df_model['Accuracy'] = accuracy.values()
df_model
```

	True Positive	False Positive	True Negative	False Negative	Accuracy
Logistic Regression	27	24	12159	827	0.934724
Support Vector Machines	12	5	12178	842	0.935031
Decision Trees	366	648	11535	488	0.912863
Random Forest	323	103	12080	531	0.951369
Naive Bayes	257	874	11309	597	0.887167
K-Nearest Neighbor	77	114	12069	777	0.931656
Gradient Boosting	326	89	12094	528	0.952673

Рисунок 2.7 – Результат класифікації всіх моделей

Ассурасу, це відсоток результатів, які правильно розподілила модель. І з однієї сторони треба обрати модель з найвищим показником точності та застосовуємо її до моделі, але бувають випадки, коли не загальна точність результатів стоїть на першому місці, а точність правильно класифікованих true positive результатів, тобто в цьому випадку це ті люди, що купили, і модель так і сказала, що вони купили.

В даній ситуації відокремлюються три моделі: Random Forest, Decision Trees та Gradient Boosting, за True Positive результатами, які важливі.

В основі оцінки якості класифікаційних моделей лежить статистика результатів класифікації навчальних прикладів. З її допомогою обчислюються метрики якості – показники, які залежать від результатів класифікації та не залежать від внутрішнього стану моделі.

Серед найпопулярніших метрики оцінки якості класифікаторів можна виділити такі:

- Матриця помилок (Confusion matrix).
- Влучність (Accuracy).
- Точність (Precision).
- Повнота (Recall).
- Специфіка (Specificity).
- F1-мера (F1-score).
- Метрика P4.
- Площа під ROC-кривою (Area under ROC-curve, AUC-ROC).
- Площа під кривою повнота-точність (Area under precision-recall curve, AUC-PR).
- Коефіцієнт кореляції Метьюса (MCC).
- Функція втрат логістичної регресії (Logistic Loss Function, Log Loss).

2.5 Хід роботи

Завдання: Необхідно провести EDA (exploratory data analysis) та побудувати модель класифікації, яка б змогла максимально точно класифікувати пацієнтів із проблемами печінки та без них за результатами аналізів. Провести оптимізацію параметрів моделі. Навести всі метрики якості для моделі класифікації.

Для виконання завдання використати датасет про пацієнтів:

Age - вік

Gender - стать

Total_Bilirubin - загальний білірубін (не прямий)

Direct_Bilirubin - прямий білірубін

Alkaline_Phosphotase - алкалайн-фосфати

Alamine_Aminotransferase - алкалайн-амінотрансфераза

Aspartate_Aminotransferase - аспарат-амінотрансфераза

Albumin - альбумін

Albumin_and_Globulin_Ratio - ставлення альбуміну до глобуліну

Total_Protiens - аналіз, який показує концентрацію білків у плазмі крові

Dataset - є чи ні у пацієнта проблеми з печінкою (клас, який потрібно передбачати).

Результатом роботи є Jupyter Notebook із коментарями до виконаної роботи.

2.6 Зміст звіту

Звіт повинен мати такі складові:

- титульний лист;
- мета роботи;
- результат виконання завдань (програмний код та скріншоти);
- відповіді на контрольні питання.

2.7 Контрольні питання

2. Яка мета дослідницького аналізу даних (EDA) у контексті машинного навчання за допомогою Python?

3. Як такі методи візуалізації даних, як гістограми та діаграми розсіювання, можуть допомогти в розумінні даних?

4. Опишіть, як коробкові діаграми можна використовувати для визначення викидів у наборі даних.

5. Яку роль відіграє перетворення даних, наприклад нормалізація та стандартизація, у підготовці даних для аналізу?

6. Поясніть важливість обробки відсутніх значень у наборі даних і методи їх вирішення.

7. Як кореляційний аналіз може допомогти зрозуміти зв'язок між різними змінними в наборі даних?

8. Причини поділу даних на навчальні та тестові набори для оцінки моделі.

9. Які загальні моделі класифікації використовуються в машинному навчанні та які їхні ключові характеристики?

10. Як показники, як точність, точність, запам'ятовуваність і оцінка F1, використовуються для оцінки ефективності моделі класифікації?

11. Які кроки передбачають виконання EDA і чому це важливо перед створенням моделей машинного навчання?

JITEPATYPA

1. Machine Learning and Knowledge Extraction [Electronic resource] / ed. by A. Holzinger [et al.]. – Cham : Springer Nature Switzerland, 2023. – Mode of access: <https://doi.org/10.1007/978-3-031-40837-3>
2. Intelligent Systems and Machine Learning [Electronic resource] / ed. by S. Nandan Mohanty, V. Garcia Diaz, G. A. E. Satish Kumar. – Cham : Springer Nature Switzerland, 2023. – Mode of access: <https://doi.org/10.1007/978-3-031-35078-8>
3. Joshi A. V. Machine Learning and Artificial Intelligence [Electronic resource] / Ameet V. Joshi. – Cham : Springer International Publishing, 2023. – Mode of access: <https://doi.org/10.1007/978-3-031-12282-8>
4. Zollanvari A. Machine Learning with Python [Electronic resource] / Amin Zollanvari. – Cham : Springer International Publishing, 2023. – Mode of access: <https://doi.org/10.1007/978-3-031-33342-2>
5. Hu F. AI, Machine Learning and Deep Learning [Electronic resource] / Fei Hu, Xiali Hei. – Boca Raton : CRC Press, 2023. – Mode of access: <https://doi.org/10.1201/9781003187158>
6. Karpatne A. Knowledge-Guided Machine Learning [Electronic resource] / Anuj Karpatne, Ramakrishnan Kannan, Vipin Kumar. – Boca Raton : Chapman and Hall/CRC, 2022. – Mode of access: <https://doi.org/10.1201/9781003143376>
7. Aggarwal C. C. Machine Learning for Text [Electronic resource] / Charu C. Aggarwal. – Cham : Springer International Publishing, 2022. – Mode of access: <https://doi.org/10.1007/978-3-030-96623-2>
8. Valkenborg D. Mathematics for Machine Learning [Electronic resource] / Dirk Valkenborg, Jeroen Wynen. – Washington, DC, USA : American Chemical Society, 2022. – Mode of access: <https://doi.org/10.1021/acsinfocus.7e5034>
9. Indovina J. Machine Learning Systems: Learn How to Develop Machine-Learning Systems / Johnson Indovina. – [S. l.] : Independently Published, 2022.
10. Applied Machine Learning / Setareh Rafatirad [et al.]. – [S. l.] : Springer International Publishing AG, 2022.

ДОДАТОК А

Інструкції з роботи з інструментами

Програмні середовища:

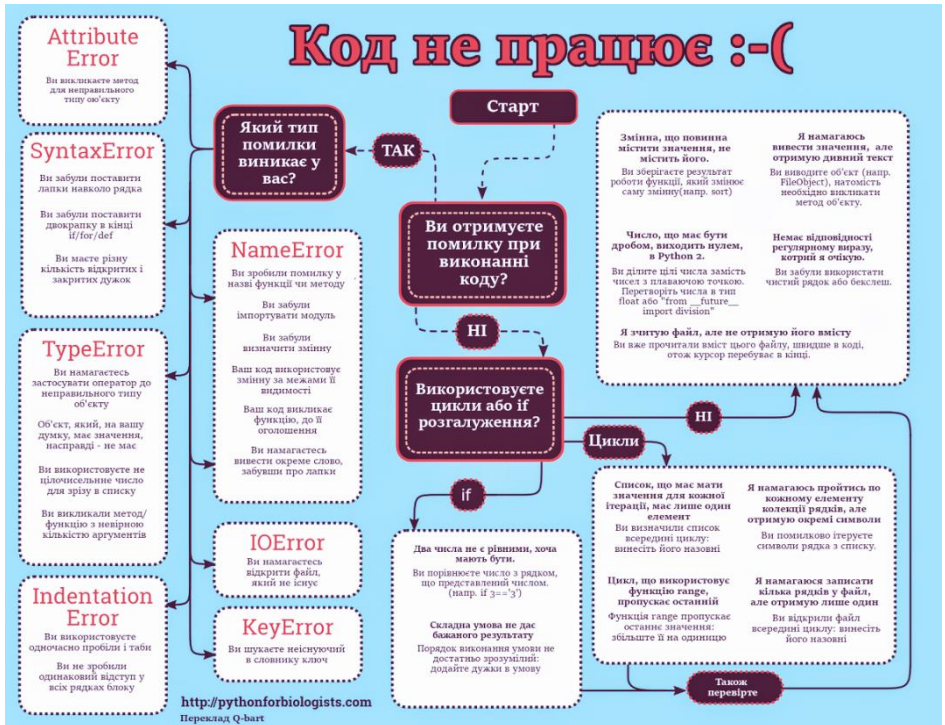
- Google Colab + інструкція по роботі:
<https://colab.research.google.com/notebooks/welcome.ipynb> ;
- встановлення Anaconda на Windows:
<https://docs.anaconda.com/anaconda/install/windows/> ;
- встановлення Jupyter Lab (без Anaconda через pip):
<https://jupyter.org/install> ;
- IntelliJ PyCharm IDE (Community Edition):
<https://www.jetbrains.com/pycharm/download/> ;

Посилання на вільно доступні датасети:

- платформа Kaggle: <https://www.kaggle.com/datasets>
- репозиторій із даними UCI:
<https://archive.ics.uci.edu/ml/datasets.php>
- Papers with code: <https://paperswithcode.com/datasets>
- список датасетів з Wikipedia:
https://en.wikipedia.org/wiki/List_of_datasets_for_machine-learning_research

ДОДАТОК Б

Типові помилки мови python



ДОДАТОК В

Пріоритет операторів

(вирази...), [вирази...], {ключ: значення...}, {вирази...}	Вираз в дужках, створення або включення списку/словника/множин и
x[індекс], x[індекс:індекс], x(аргументи ...), x.атрибут	Індекс, зріз, виклик функції, посилання на атрибут
**	Піднесення до степеня
+x, -x, ~x	Знаки «+» і «-», бітове НІ
*, /, //, %	Множення, ділення з плаваючою крапкою, цілочисельне ділення, остача від ділення
+, -	Додавання, віднімання
<<, >>	Бітові зсуви
&	Бітове І
	Бітове АБО
^	Бітове виключення АБО
in, not in, is, is not, <, <=, >, >=, !=, ==	Перевірка на приналежність і рівність
not x	Булеве (логічне) НІ
and	Булеве (логічне) І
or	Булеве (логічне) АБО
if ... else	Умовний вираз
lambda	Лямбда-вираз