МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД
«УЖГОРОДСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ»
ФАКУЛЬТЕТ МАТЕМАТИКИ ТА ЦИФРОВИХ ТЕХНОЛОГІЙ
КАФЕДРА КІБЕРНЕТИКИ І ПРИКЛАДНОЇ МАТЕМАТИКИ

ПОПЕРЕДНЯ ОБРОБКА ТА АНАЛІЗ ДАНИХ

Лабораторний практикум

Ужгород - 2023

Попередня обробка та аналіз даних: лабораторний практикум для студ. спеціальності 113 «Прикладна математика» /Уклад.: Н. Е. Кондрук. Ужгород: УжНУ, 2023. - 41 с.

Укладач:

Кондрук Н. Е., кандидат технічних наук, доцент кафедри кібернетики і прикладної математики.

Рецензент:

Маляр М. М., доктор технічних наук, професор, декан факультету математики і цифрових технологій.

Рекомендовано до друку методичною комісією ФМЦТ ДВНЗ "Ужгородський національний університет", протокол №6 від 24 лютого 2023 року.

Рекомендовано до друку Вченою радою ФМЦТ ДВНЗ "Ужгородський національний університет", протокол №8 від 26 квітня 2023 року.

Зміст

ОСОБЛИВОСТІ РОБОТИ В Google Colaboratory

Google Colaboratory, також відомий як Google Colab (https://colab.research.google.com/) – це безкоштовне інтерактивне хмарне середовище для роботи з кодом від Google для розв'язання задач машинного навчання.

КОРИСТУВАЧІ GOOGLE COLAB:

- аналітики даних (щоб сортувати дані за тривалий період, робити візуалізацію чи вибудовувати закономірності);
- дослідники даних (щоб розробляти та тестувати нові моделі машинного навчання, складати прогнози);
- інженери даних (щоб розробляти програмне забезпечення, системи зберігання великих даних).

В основі «Колабораторії» лежить блокнот Jupyter для роботи на Python, тільки з базою на Google Диску, а не на комп'ютері. Тобто можна програмувати на Python і не завантажувати бібліотеки перевантажуючи жорсткий диск свого комп'ютера. Єдина умова – потрібний обліковий запис Google.

Головна особливість «Колабораторії» – безкоштовні потужні графічні процесори GPU та TPU, завдяки яким можна займатися не лише базовою аналітикою даних, а й складнішими дослідженнями в галузі машинного навчання. Що CPU обчислює годинами, GPU або TPU справляються за хвилини або секунди.

CPU – центральний процесор – мозок комп'ютера, який виконує операції з даними. GPU – графічний процесор. Обробляє дані швидше, оскільки завдання виконує паралельно, а не послідовно, як CPU. TPU – тензорний процесор, розробка Google. Він призначений для тренування нейромереж. У нього в рази вища продуктивність при великих обсягах обчислювальних завдань.

Процесори коштують дорого, і не кожен може їх собі дозволити. Google Colaboratory дає можливість безкоштовно та безперервно користуватися ними протягом 12 годин. Увага: як тільки цей час сплине, Colab зітре всі дані і доведеться починати спочатку.

Крім того, Google відключає блокноти після приблизно 30 хвилин бездіяльності, щоб не перевантажувати процесори. Дуже активним учасникам ненадовго можуть обмежити доступ до GPU, щоб надати можливість використовувати процесор іншим.

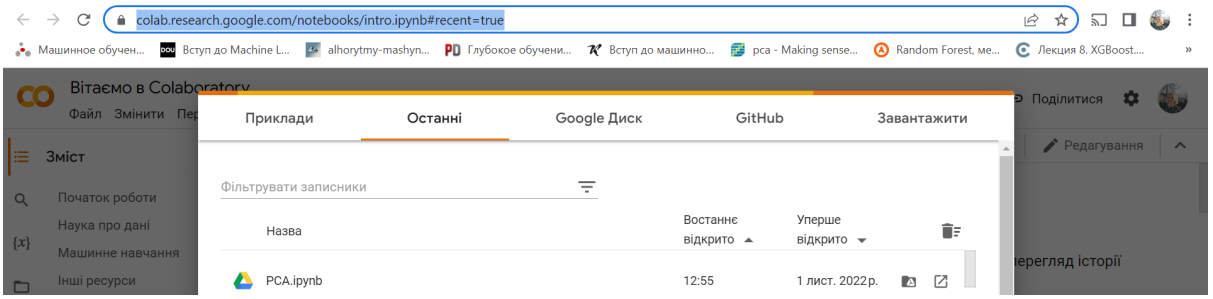
GOOGLE COLAB використовується для:

- знайомства з TensorFlow – відкритою бібліотекою для машинного навчання;
- розробкою нейронних мереж;
- експериментами з TPU;
- поширення досліджень у галузі штучного інтелекту;
- створення посібників.

Декілька таких прикладів є у відкритому доступі прямо в Colab.

ПОЧАТОК РОБОТИ

Зайти в свій обліковий запис від Гугл, та перейти на [сайт](https://colab.research.google.com/notebooks/intro.ipynb#recent=true) (https://colab.research.google.com/notebooks/intro.ipynb#recent=true) сервісу, одразу з'являється екран із доступними блокнотами. Можна створювати новий або завантажувати вже розроблений Python-код із Google Диску.



Google Colab максимально спростив усі процеси: у ньому вбудовано і базові бібліотеки (NumPy , scikit-learn, Pandas), і більш складні (на зразок Keras, TensorFlow або PyTorch), їх не потрібно завантажувати самостійно, можна просто одразу писати код. Якщо базових бібліотек недостатньо, завжди можна додати необхідні за допомогою установника PIP і працювати далі.

```
%pip install emoji
```

У Colab можна ділитися роботою з іншими, залишати коментарі, редакторські нотатки та загалом робити все, що є в тих же Google Документах. Тому при загальному доступі до блокноту весь його вміст буде доступним іншим користувачам (текст, код, коментарі, вихідні дані).

Google має великий репозиторій SeedBank (https://aihub.cloud.google.com/s?category=notebook), в якому можна дослідити блокноти з Data Science або глибокого навчання, просто клікнувши мишкою.

ЗАВАНТАЖЕННЯ ФАЙЛІВ

Для завантаження баз даних (датасетів) найзручніше користуватись наступним алгоритмом.

1. Завантажити датасет на комп'ютер у csv-форматі.
2. В кодовій комірці набрати наступний код і запустити її на виконання.

```
#Reading CSV files
from google.colab import files
files.upload()
```

3. З'явиться кнопка “Вибрати файл”, клікаємо її, знаходимо на жорсткому диску завантажений файл та очікуємо звіту про успішне завантаження.
4. Імпортуємо датасет.

```
import pandas as np
# importing or loading the dataset
dataset = pd.read_csv('імя_файлу.csv')
```

5. У параметрі dataset буде міститися завантажений датасет.

Також в бібліотеку scikit-learn вже вбудовано підбірку датасетів. Щоб завантажити та переглянути опис одного з них скористайтесь кодом.

```
from sklearn import datasets
diabetes=datasets.load_diabetes()
print(diabetes.DESCR)
```

Більше про утиліти завантаження баз даних із даної бібліотеки можна ознайомитись в Dataset loading utilities.

БЕЗКОШТОВНІ ХМАРНІ СЕРЕДОВИЩА, ПОДІБНІ ДО GOOGLE COLAB

Kaggle Kernels – окрім Python, сервіс Kaggle підтримує R, інтегрується з Google Cloud Storage, BigQuery та AutoML. При цьому час користування процесорами – 9 годин. Також на платформі можна безоплатно пройти сертифіковані курси по машинному навчанню, переглянути дослідження інших користувачів та завантажити датасети, які є у вільному доступі. Крім того, організатори постійно проводять різні конкурси із грошовими нагородами для переможців досліджень.

CoCalc – пропонує і безкоштовний, і платний періоди. У розширеній версії більше пам'яті та часу простою, пріоритетний доступ до процесорів та техпідтримки.

ПОСИЛАННЯ НА ВІДКРИТІ СХОВИЩА ДАТАСЕТІВ

1. DATASET SEARCH
2. DATA.WORLD
3. UCI
4. fivethirtyeight/data
5. KAGGLE
6. European Data Portal
7. Home - Eurostat
8. Data.Gov
9. Datahub.io
10. Earth Data
11. CERN Open Data Portal
12. Global Health Observatory Data Repository
13. BFI film industry statistics
14. NYC Taxi Trip Data
15. FBI Crime Data Explorer

ПОПЕРЕДНЯ ОБРОБКА ДАНИХ

Попередня обробка даних – це процес перетворення необроблених даних у значущі за допомогою різних методів.

Дані в реальному світі “неочищені, сирі”. Це означає, що вони можуть бути неповні, містити відсутні, непотрібні, повторювані дані, дані з шумом тощо. Використання такого типу необроблених даних у моделях машинного навчання не забезпечує її хорошої продуктивності. Щоб отримати хорошу та точну продуктивність, необхідно попередньо обробити дані, тобто перетворити необроблені дані на значущі.

ОСНОВНІ ЕТАПИ ПОПЕРЕДНЬОЇ ОБРОБКИ ДАНИХ

ОЧИЩЕННЯ ДАНИХ

1. Видаліть повторювані значення.

2. Перетворіть у відповідний тип даних відповідно до ознаки.

3. Заповніть пропущені значення або видаліть стовпці, рядки чи комірки, які містять пропущені значення.

Для виявлення пропусків в даних використовують різні підходи.

```
# сумування нульових елементів по ознаках
df.isnull().sum()
# вивід інформації про дані, в т.ч. і нульових значень
print(df.info())
```

У процесі очищення даних потрібно:

1. Заповнити відсутні значення, якщо бракує кілька елементів.

Використовуйте глобальні константи. Ця техніка може бути причиною поганої точності.

2. Видалити стовпець або рядок, який містить забагато пропущених значень. Якщо відсутня мала кількість даних, видаліть тільки їх. Наприклад, нехай, є стовпець, який містить 80% пропущених значень. У цьому випадку доцільно видалити цей стовпець. Нехай є 1 мільйон даних і лише 100 відсутніх. У цьому випадку видаляємо тільки відсутні значення.

Приклад даних з відсутніми значеннями:

Id	Name	Department
1	A	S
2	B	C
2	nan	A
4	D	S
5	nan	S
6	nan	A
7	D	C
8	nan	C

Якщо заповнити відсутні значення некоректно, то це може створити велику проблему в подільшій роботі моделі. Для усунення цієї проблеми існують різні техніки та підходи.

Видалення рядка для очищення даних: якщо заданий набір даних містить забагато пропущених значень у рядку, тому можна його видалити.

Id	Name	Department
1	A	S
2	B	C
4	D	S
7	D	C

Видалення стовпця для очищення даних: якщо у наборі даних забагато пропущених значень у стовпці, то його можна видалити.

Id	Department
1	S
2	C
2	A
4	S
5	S
6	A
7	C
8	C

Використовування глобальних констант: заповнимо всі відсутні дані за допомогою одного значення, наприклад null, missing, sorry, ignore тощо.

3. Заповнення відсутніх значень, використовуючи середнє значення, медіану або моду. Для категоріальної ознаки можна використовувати лише моду. Для числової – середнє значення, медіану або моду. Якщо дані розподілені нормально, краще використовувати середнє значення, а якщо мають спотворений розподіл, потрібно використовувати медіану.

```
import pandas as np
df=pd.read_csv("practice2.csv")
print(df)
```

	total_bill	sex	time	tip	size	day	smoker
0	500.0	male	lunch	30	2	sat	no
1	648.0	male	dinner	35	3	sat	yes
2	75.0	female	NaN	10	1	sat	no
3	159.0	female	NaN	12	4	sat	NaN
4	250.0	male	lunch	3	3	NaN	no
5	222.0	female	lunch	25	3	mon	yes
6	356.0	male	dinner	30	4	mon	NaN
7	99.0	female	NaN	5	1	mon	no
8	150.0	male	dinner	NaN	4	mon	NaN
9	NaN	female	lunch	38	4	mon	no
10	478.0	female	lunch	28	3	tue	yes
11	320.0	nan	NaN	14	3	tue	yes
12	NaN	male	lunch	32	2	tue	NaN
13	520.0	male	lunch	33	3	NaN	yes
14	367.0	male	dinner	NaN	3	tue	yes

Використовуючи функцію replace(), можна замінити одні значення іншими. Перший параметр – це значення, яке потрібно замінити, а другий – нове значення. Також можлива заміна значень у конкретній ознаці.

```
r1=df.replace("lunch","Lunch new")
print(r1)
r2=df.replace(["female","sat"],["New female","New
sat"])
print(r2)
r3=df.replace(["female","sat"],"Value change")
print(r3)
# заміна значення 'male' на 'm' ознаки 'sex'
df['sex'] = df['sex'].replace(['male'], 'm')
```

Результати.

r1

	total_bill	sex	time	tip	size	day	smoker
0	500.0	male	Lunch new	30	2	sat	no
1	648.0	male	dinner	35	3	sat	yes
2	75.0	female	NaN	10	1	sat	no
3	159.0	female	NaN	12	4	sat	NaN
4	250.0	NaN	Lunch new	40	3	NaN	no
5	NaN	male	Lunch new	25	3	mon	yes
6	356.0	male	dinner	30	4	mon	NaN
7	99.0	female	NaN	5	1	mon	no
8	150.0	female	dinner	15	4	mon	NaN
9	NaN	NaN	Lunch new	38	4	mon	no
10	478.0	NaN	Lunch new	28	3	NaN	yes
11	320.0	female	NaN	14	3	tue	yes
12	NaN	male	Lunch new	32	2	tue	NaN
13	520.0	male	Lunch new	33	3	NaN	yes
14	367.0	male	dinner	40	3	NaN	yes
15	NaN	NaN	NaN	40	4	wed	yes
16	100.0	female	dinner	6	1	wed	NaN
17	NaN	female	NaN	40	1	wed	yes
18	199.0	NaN	Lunch new	35	3	NaN	no
19	288.0	male	dinner	21	2	wed	NaN
20	120.0	female	dinner	18	1	thu	yes
21	168.0	male	Lunch new	15	4	NaN	yes
22	NaN	female	dinner	10	3	thu	NaN
23	284.0	male	NaN	35	2	NaN	yes

r3

	total_bill	sex	time	tip	size	day	smoker
0	500.0	male	lunch	30	2	Value change	no
1	648.0	male	dinner	35	3	Value change	yes
2	75.0	Value change	NaN	10	1	Value change	no
3	159.0	Value change	NaN	12	4	Value change	NaN
4	250.0	NaN	lunch	40	3	NaN	no
5	NaN	male	lunch	25	3	mon	yes
6	356.0	male	dinner	30	4	mon	NaN
7	99.0	Value change	NaN	5	1	mon	no
8	150.0	Value change	dinner	15	4	mon	NaN
9	NaN	NaN	lunch	38	4	mon	no
10	478.0	NaN	lunch	28	3	NaN	yes
11	320.0	Value change	NaN	14	3	tue	yes
12	NaN	male	lunch	32	2	tue	NaN
13	520.0	male	lunch	33	3	NaN	yes
14	367.0	male	dinner	40	3	NaN	yes
15	NaN	NaN	NaN	40	4	wed	yes
16	100.0	Value change	dinner	6	1	wed	NaN
17	NaN	Value change	NaN	40	1	wed	yes
18	199.0	NaN	lunch	35	3	NaN	no
19	288.0	male	dinner	21	2	wed	NaN
20	120.0	Value change	dinner	18	1	thu	yes
21	168.0	male	lunch	15	4	NaN	yes
22	NaN	Value change	dinner	10	3	thu	NaN
23	284.0	male	NaN	35	2	NaN	yes

Функція Dropna() використовується для видалення рядків, стовпців, які містять значення “nan”. За замовчуванням axis=0, що означає, що ця функція працюватиме по рядку, і якщо вона знайде будь-який “nan” у рядку, то видалить його; але якщо axis=1, тоді Dropna() видалятиме

стовпці із “nan”. Якщо how="all", тоді функція видалить ті стовпці або рядки, які мають усі значення “nan”. Також можна визначити, скільки значень “nan” потрібно для видалення рядка чи стовпця за допомогою параметра thresh.

```
d1=df.dropna()
print(d1)
d2=df.dropna(axis=1)
print(d2)
d3=df.dropna(how="all")
print(d3)
d4=df.dropna(thresh=4,axis=0)
print(d4)
```

Результати.

d1

	total_bill	sex	time	tip	size	day	smoker
0	500.0	male	lunch	30	2	sat	no
1	648.0	male	dinner	35	3	sat	yes
20	120.0	female	dinner	18	1	thu	yes

d2

	time	tip
0	30	2
1	35	3
2	10	1
3	12	4
4	40	3
5	25	3

ФУНКЦІЯ FILLNA()

Fillna використовується для заповнення відсутніх значень (пропусків). Якщо передати method="ffill" у функції fillna, тоді всі відсутні значення будуть заповнені попереднім значенням. Якщо method="bfill" у функції fillna, усі відсутні значення будуть заповнені наступним значенням.

```
f1=df.fillna("FFFFF")
print(f1)
f2=df.fillna(method="ffill")
print(f2)
f3=df.fillna(method="bfill")
print(f3)
# заміна пропусків у стовпцях  вказаними значеннями
df=df.fillna({"day":"sat","total_bill":0})
```

```
# заміна пропусків у стовпці наступними значеннями
df[['time']] = df[['time']]. fillna (method="bfill")
# заміна пропусків нулями
df[['total_bill']] = df[['total_bill']]. fillna (0)
```

Результати.

f1	f2

	total_bill	sex	time	tip	size	day	smoker
0	500.0	male	lunch	30	2	sat	no
1	648.0	male	dinner	35	3	sat	yes
2	75.0	female	FFFFF	10	1	sat	no
3	159.0	female	FFFFF	12	4	sat	FFFFF
4	250.0	FFFFF	lunch	40	3	FFFFF	no
5	FFFFF	male	lunch	25	3	mon	yes
6	356.0	male	dinner	30	4	mon	FFFFF
7	99.0	female	FFFFF	5	1	mon	no

	total_bill	sex	time	tip	size	day	smoker
0	500.0	male	lunch	30	2	sat	no
1	648.0	male	dinner	35	3	sat	yes
2	75.0	female	dinner	10	1	sat	no
3	159.0	female	dinner	12	4	sat	no
4	250.0	female	lunch	40	3	sat	no
5	250.0	male	lunch	25	3	mon	yes
6	356.0	male	dinner	30	4	mon	yes
7	99.0	female	dinner	5	1	mon	no

ЗАПОВНЕННЯ ПРОПУСКІВ НУЛЯМИ, СЕРЕДНІМ ЗНАЧЕННЯМ, МЕДІАНОЮ АБО МОДОЮ

```
import pandas as pd
# заміна пропусків на середні значення 1-го стовця
df['total_bil']=df['total_bil'].fillna(df['total_bil'].mean())
# заміна пропусків на середні значення по всій таблиці
df = df.fillna(df.mean())

# заміна пропусків на нуль 1-го стовця
df['total_bil']=df['total_bil'].fillna(0)
# заміна пропусків на медіану 1-го стовця
df['total_bil']=df['total_bil'].fillna(df['total_bil'].median())
# заміна пропусків на моду 1-го стовця
df['total_bil']=df['total_bil'].fillna(df['total_bil'].mode()[0])
```

Увага! Для числових ознак можна використовувати заміну пропусків модою, медіаною, середнім значенням. Для категоріальних ознак можна використовувати тільки моду!

КОДУВАННЯ КАТЕГОРІАЛЬНИХ МІТОК

Щоб перетворити категоріальне значення в числовий формат, виконується перекодування, щоб модель машинного навчання могла працювати з ним.

Приклад.

Total_bill	Sex	Day	Time	Size
16.99	Male	Sun	Dinner	3
10.50	Female	Sun	Dinner	2
24.59	Male	Sun	Dinner	4
8.77	Male	Mon	Dinner	2
15.07	Female	Mon	Lunch	2
10.27	Male	Mon	Lunch	4
35.66	Female	Fri	Lunch	4
21.15	Female	Fri	Lunch	2
18.49	Male	Fri	Dinner	3

Датасет містить дані ресторану, які показують загальний рахунок, виставлений клієнтами, їхню стать, назву дня, час їжі та розмір. Тут total_bill буде прогнозованою неперервною міткою, а всі інші ознаки, крім останньої – категоріальні. Але моделі машинного навчання можуть працювати тільки з числовими значеннями. Щоб вирішити цю проблему, виконується перекодування, яке можна реалізувати за допомогою бібліотеки pandas або sklearn.

КОДУВАННЯ PANDAS DUMMY_VARIABLE

```
import pandas as pd
df=pd.read_csv("my_bill.csv")
encoding=pd.get_dummies(df)
print(encoding)
```

Після кодування отримаємо

Total_bill	Sex_male	Sex_female	Day_sun	Day_mon	Day_fri	Time_dinner	Time_lunch	Size
16.99	1	0	1	0	0	1	0	3
10.50	0	1	1	0	0	1	0	2
24.59	1	0	1	0	0	1	0	4
8.77	1	0	0	1	0	1	0	2
15.07	0	1	0	1	0	0	1	3
10.27	1	0	0	1	0	0	1	4
35.66	0	1	0	0	1	0	1	4
21.15	0	1	0	0	1	0	1	2
18.49	1	0	0	0	1	1	0	3

Після кодування унікальне значення кожної категоріальної ознаки стає окремим стовпцем, які містять лише значення 0 і 1. Тут 1 означає “так”, а 0 означає “ні”. Зокрема, два стовпці створюються із ознаки Sex, три стовпці утворились із Day, два – із Time. Це відбувається тому, що стовпець «Стать» має два унікальних значення: чоловіча і жіноча, ознака Day має три унікальні значення, а ознака Time – два унікальні значення. Тому для кожного унікального значення буде створено окремий стовпець. Наприклад, ви отримуєте стовпець sex_female зі стовпця sex, де 1 відповідає “жінка”, а 0 – “чоловік”. У стовпці sex_male 1 це “чоловік”, а 0 – жінка. Отже, 1 означає “так”, це значення комірки присутнє в головному стовпці, а 0 означає, що його немає.

```
# перекодування тільки однієї ознаки
pd.get_dummies(df,columns=['Sex'], drop_first= True )
```

Параметр DROP_FIRST

```
import pandas as pd
df=pd.read_csv("my_bill.csv")
'''
dataset link:
```

https://www.kaggle.com/code/yashsharmabharatpur/tips-collected-at-a-restaurant/data
'''

```
encoding=pd.get_dummies(df,drop_first=True)
print(encoding)
```

Total_bill	Sex_female	Day_mon	Day_fri	Time_lunch	Size
16.99	0	0	0	0	3
10.50	1	0	0	0	2
24.59	0	0	0	0	4
8.77	0	1	0	0	2
15.07	1	1	0	1	3
10.27	0	1	0	1	4
35.66	1	0	1	1	4
21.15	1	0	1	1	2
18.49	0	0	1	0	3

Параметр drop_first має два значення: 1-False (за замовчуванням); 2-True.

Параметр використовується для видалення перших стовпців серед усіх тих, які утворені однією ознакою після кодування. Нехай зі стовпця Sex, утворяться два стовпці Sex_male і Sex_female. Таким чином, цей параметр видалить перший стовпець – sex_male, оскільки за значеннями інших стовпців (sex_female) можна визначити його значення.

ONE HOT ENCODING

Результат даного підходу буде таким самим, як і у функції pandas dummy_variable. Метод one hot encoding з бібліотеки sklearn, а dummy_variable — з бібліотеки pandas. Також, як і в dummy_variable можна відкинути стовпець, але не обов'язково перший. Є лише одна різниця між dummy_variable і one-hot encoding, а саме: у dummy_variable вихідні дані отримуються у вигляді стовпця data frame, а в one-hot encoding

- у формі масиву. Вже після отримання результату перетворюємо його в data frame.

Проблема 1.

One-Hot Encoding призводить до фіктивних змінних - Dummy Variable Trap. Це означає, що змінні сильно корелюють одна з одною: одну змінну можна легко передбачити за допомогою решти. Dummy Variable Trap може бути причиною мультиколінеарності. Це відбувається, якщо існує залежність між ознаками. Щоб подолати проблему мультиколінеарності, потрібно видалити одну з фіктивних змінних.

Проблема 2.

Якщо в One-Hot Encoding ознака має 10 унікальних значень, то для неї буде створено 10 різних стовпців, що може бути причиною поганої точності.

Припустимо, що серед 10-ти унікальних значень 5 зустрічаються часто, а інші 5 – рідше. У цьому випадку ті значення ознаки, які зустрічаються рідко доцільно перетворити в одне значення (“інше”). Після даного перетворення отримаємо 6 унікальних значень (5 унікальних значень, які зустрічаються найчастіше + перетворення решти п'яти в одне). Тож тепер застосуємо one-hot encoding лише до цих 6 значень. Цією технікою можна трохи зменшувати розмірність простору ознак.

Коли використовувати One-hot encoding?

1. Якщо дані не впорядковані, тобто їх неможливо проранжувати за певною перевагою.

2. Якщо кількість категоріальних ознак невелика.

3. Якщо категоріальні ознаки мають більше двох значень.

```
import pandas as pd
from sklearn.preprocessing import OneHotEncoder
df=pd.read_csv("my_bill.csv")
'''
dataset link:
https://www.kaggle.com/code/yashsharmabharatpur/tips-collected-at-a-restaurant/data
'''
encoding= OneHotEncoder(sparse=False, drop="first")
'''
Якщо ви хочете кодувати дані у формі розрідженої матриці, скористайтеся sparse=True, а щоб отримати масив numpy, використовуйте sparse= False
'''
'''
Тепер підберіть і трансформуйте ті стовпці, які потрібно закодувати
'''
result_encod=encoding.fit_transform(df[["sex","day","time"]])

print(result_encod)

'''
Після кодування ми отримуємо дані у вигляді масиву numpy. Для подальшої роботи нам потрібно перетворити цей масив у dataframe.
'''
'''
Отримання назв стовпців, після кодування. Тут ми використали фіктивну змінну, щоб отримати назви стовпців.
'''
print(pd.get_dummies(df[["sex","day","time"]],drop_first=True).keys())

'''
Створимо фрейм даних з тих стовпців, які отримано після кодування. Тут потрібно назви стовпців. Скопіюйте назви стовпців, які ми отримуємо після використання порожніх елементів, і вставте сюди.
'''
dataFrame_encode_col=pd.DataFrame(result_encod,columns=['sex_male', 'day_sat', 'day_thu', 'day_tue',
```

```
'day_wed', 'time_lunch'])
'''
Відфільтруємо числовий стовпець із набору даних
'''
numeric_col=df[["total_bill","tip","size"]]
#Create dataset
final_dataset=pd.merge(dataFrame_encode_col,numeric_c
ol,left_index=True,right_index=True)
print(final_dataset)
```

LABEL ENCODING

У кодуванні dummy_variable та OneHotEncoder створюється кілька стовпців із однієї ознаки відповідно до унікальних значень. Label encoding кодує категоріальні значення в одному стовпці. Припустимо, є 3 значення ознаки: сорочка, штани та годинник. Кодувальник міток перетворить сорочку на 0, штани на 1 і годинник на 2. Він не створюватиме жодного нового стовпця, а кодуватиме в тому самому.

Column without encode	-->	Column after encode
shirt	-->	0
watch	-->	2
watch	-->	2
shirt	-->	0
watch	-->	2
shirt	-->	0
pant	-->	1
pant	-->	1

Проблеми.

Припустимо, що дані не є порядковими, наприклад деякі назви країн (Бангладеш, США, Канада, Великобританія). Немає порядку чи рангу між

назвами цих країн. Label encoding виконуватиме кодування в одному стовпці та може перетворювати Бангладеш в 1, США в 2, Канаду в 3 і Великобританію в 0. Тоді модель машинного навчання може вважати, що Канада має вищий ранг, ніж США (Канада(3)>США(2)>Бангладеш(1)>Великобританія(0)), але насправді це не так, вони рівнозначні.

Коли використовувати label encoding?

1. Якщо в категоріальному стовпці лише два значення, як-от “чоловіча” чи “жіноча”, “так” чи “ні” тощо.

2. У задачах бінарної класифікації.

3. Рекомендовано використовувати кодування міток у цільовому (класовому) стовпці або залежній предикторній змінній (Y), а не в незалежних змінних (описувальних ознаках X).

```
import pandas as pd
from sklearn.preprocessing import LabelEncoder
df=pd.read_csv("my_bill.csv")
'''
dataset link:
https://www.kaggle.com/code/yashsharmabharatpur/tips-collected-at-a-restaurant/data
'''
encoding=LabelEncoder()
df["sex"]=encoding.fit_transform(df["sex"])
df
```

ORDINAL ENCODING

Іноді необхідно перекодувати ознаку, яка приймає ранжовані значення. Для цих типів даних використовується порядкове кодування. Припустимо, у вас є стовпець, який містить такі оцінки A, B, і C. Причому

A переважніше за B, B переважніше за C. Отже, якщо використовувати кодування one-hot або dummy_variable, тоді всі значення будуть перетворені в 0 або 1. Це означає, що значення всіх оцінок стануть однорідними. В той же час, якщо студент отримує A з математики, а інший - B, то це означає, що перший студент знає математику краще. Тому, якщо застосувати dummy_variable або OneHotEncoder, то рівні A і B урівнюються, але насправді це не відповідає дійсності. Отже, доцільніше використати мітку 1 для A і 2 для B, тоді машина зрозуміє, що A краще, ніж B. Для виконання цього типу кодування використовується порядкове кодування.

```
import pandas as pd
df=pd.read_csv("my_bill.csv")
'''
dataset link:
https://www.kaggle.com/code/yashsharmabharatpur/tips-collected-at-a-restaurant/data
'''
#Тут ми визначаємо ранг наших значень, задаєм мапувальню
Order_day={"sat":1,"mon":2,"tue":3,"wed":4,"thu":5}
df["day"]=df["day"].map(Order_day)
df
```

ІНЖЕНЕРІЯ ОЗНАК

Іноді в даних може бути багато стовпців або ознак, але не всі вони пов’язані з цільовою ознакою. Потрібно залишити тільки корельовані (зв'язані) та видалити зайві, такий підхід називається редукцією простору ознак. В окремих випадках доцільним є створення нових ознак на основі вхідних, щоб отримати кращу точність моделі. Це можна реалізувати шляхом поділу або об’єднання кількох ознак або зміною типів. Отже, якщо наявні ознаки не дають належної точності результату або потрібно отримати високу продуктивність моделі іноді доцільним є створення нових ознак.

Незалежна ознака означає ту в наборі даних, яка не залежить від інших. Припустимо, є чотири характеристики одягу: ціна, назва, колір, країна. Ціна є залежною змінною, тому що вона залежить від інших характеристик, таких як назва, колір, країна. Назва, колір і країна є незалежними змінними, оскільки вони не залежать від жодних інших ознак.

Етапи формування нової ознаки:

1. Тестування ознак. Спочатку перевірте наявні ознаки, якщо вони не ефективні, перейдіть до нових.

2. Вирішіть, які ознаки слід створити.

3. Створення ознак.

4. Перевірка ефективності нових ознак щодо моделі.

5. Удосконаліть ознаки, якщо потрібно.

Приклад:

Нехай у датасеті є дві ознаки: одна – це час посадки, а інша – час прибуття до станції.

Час посадки в літак	Літак досяг станції
01:00	01:00
03:00	3:15
04:00	04:10

Після їх перегляду можна створити дві нові ознаки: Затримка і Час затримки в хвилинах.

Час посадки в літак	Літак досяг станції	затримка	Час затримки в хвилинах
01:00	01:00	Вчасно	00:00
03:00	3:15	Затримується	00:15
04:00	04:10	Затримується	00:10

ТЕХНІКИ ВІДБОРУ ОЗНАК

Щоб отримати хорошу точність моделі машинного навчання, потрібно обрати ті з них, які відіграють важливу роль.

Припустимо, що є три незалежні ознаки та одна залежна або цільова. Використовуючи методи вибору ознак, спробуємо знайти кореляцію між залежною та цільовою змінною. Якщо незалежна фіча корелюється з цільовою, то залишаємо цю ознаку, а якщо ні, то видаляємо.

Корельований означає, що якщо значення незалежної ознаки збільшується, то значення цільової ознаки також збільшиться (зменшиться).

ВІДБІР ОЗНАК НА ОСНОВІ ДИСПЕРСІЇ

За допомогою цієї техніки будуть видалені ті, які містять сталі значення або, можна сказати, ознаки з низькою дисперсією. Вони не

важливі для алгоритму машинного навчання, так як рахується, що містять мало інформації.

Високий показник дисперсії, як правило необхідний для побудови продуктивної моделі.

Отже, якщо є сталі ознаки з низькою дисперсією, то їх доцільно видалити.

Приклад.

```
import pandas as pd
data=pd.DataFrame({"A":[1,2,3,4],
                                "B":[5,6,7,8],
                                "C":[0,0,0,0],
                                "D":[1,1,1,1]})

"""
Тут стовпці C і D мають сталі значення, тому їх
потрібно видалити. Для цього з бібліотеки sklearn
буде використано клас під назвою VarianceThreshold.
Постійні ознаки завжди мають низьку дисперсію. Поріг
дисперсії можна задати за допомогою параметра
threshold. Алгоритм вибору ознак завжди потрібно
використовувати лише для незалежних ознак, а не для
цільових. Стовпці C і D містять однакові значення,
отже, дисперсія дорівнюватиме 0. Тому, алгоритм
видалить їх. При цьому параметр threshold= 0.
"""
from sklearn.feature_selection import
VarianceThreshold
vari_threshold=VarianceThreshold(threshold=0)
vari_threshold.fit(data)
constant_columns=[column for column in data.columns
if column not in
          data.columns[vari_threshold.get_support()]]
for feature in constant_columns:
    print(feature)
"""
Друкуються ті назви стовпців, які мають низьку
дисперсію та повинні бути видалені.
"""
data.drop(constant_columns,axis=1,inplace=True)
data
```

```
#Видалення стовпця.
```

Застосування даної техніки на реальному прикладі.

```
import pandas as pd
from sklearn.feature_selection import
VarianceThreshold

fdf=pd.read_csv("Santander.csv")
fdf.head()

'''
Занесемо всі числові незалежні ознаки в одну змінну
'''
independent_feature=fdf.drop("TARGET",axis=1)
independent_feature.head(2)

'''
Імпортуємо VarianceThreshold з sklearn і використаємо
його. Порогове значення - 0.
'''
vari_threshold=VarianceThreshold(threshold=0)
vari_threshold.fit(independent_feature)

constant_columns=[column for column in
independent_feature.columns if column not in
independent_feature.columns[vari_threshold.get_suppor
t()]]
for feature in constant_columns:
     print(feature)
'''
У змінну constant_columns занесено всі стовпці із
сталими значеннями.
'''
#Їх видалення з independent_feature
independent_feature.drop(constant_columns,axis=1)

X=independent_feature
Y=df["TARGET"]
"""
У змінній X, занесено всі незалежні ознаки, а в Y -
```

```
цільову або залежну.
"""
```

ВІДБІР ОЗНАК НА ОСНОВІ КОРЕЛЯЦІЇ

Предикторні ознаки повинні бути корельовані із залежними та некорельовані одна з одною. Оскільки, якщо предикторна ознака сильно корелює з іншою, то її можна передбачити за іншою. Припустимо, три ознаки сильно корелюють одна з одною. Отже, у модель потрібно додати лише одну з них, яка більше корелює з цільовою ознакою. Інакше вони поводитимуться як дублікати.

Щоб знайти кореляцію між предикторними ознаками, використовуємо порогові значення, наприклад 0,7 чи 0,8. Якщо кілька незалежних ознак співвідносяться між собою на більше, ніж порогове значення, тоді залишаємо лише одну з них.

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


# Завантаженн набору даних
fdf=pd.read_csv("D:/USa_house_price.csv")
fdf.head()

'''
Відбір незалежних ознак
'''
independent_feature=fdf.drop("Price",axis=1)
independent_feature.head(2)

'''
Побудова теплової карти кореляції між стовпцями
'''
```

```
plt.figure(figsize=(12,10))
cor=independent_feature.corr()
sns.heatmap(cor,annot=True,cmap=plt.cm.CMRmap_r)
plt.show()

#Отримання стовпців кореляції
'''
Створимо функцію для вибору сильно корельованих
ознак,порівнюючи їх кореляцію за пороговим значенням.
Якщо кореляція ознак його перевищує, позначимо їх для
видалення.
'''
def correlation(dataset,threshold):
    col_corr=set()
    #встановлюємо всі назви корельованих стовпців
    corr_matrix=dataset.corr()
    for i in range(len(corr_matrix.columns)):
        for j in range(i):
            if abs(corr_matrix.iloc[i,j])>threshold:
                #Визначаємо абсолютне значення
                colname=corr_matrix.columns[i]
                #Отримуємо назви стовпців
                col_corr.add(colname)
    return col_corr
"""
За допомогою цієї функції буде вибрано
висококорельовану ознаку. Перша корельована  ознака
видаляється. У параметрі threshold - поріг кореляції,
наприклад, 80%. Зазвичай використовується 85% як
порогове значення.
"""

corr_features=correlation(independent_feature,0.7)
corr_features

independent_feature.drop(corr_features,axis=1,inplace
=True)
independent_feature.head(2)
```

ІНФОРМАЦІЙНИЙ ПРИРІСТ ВІДБОРУ ОЗНАК

В інженерії ознак потрібно знайти залежність між предикторними та цільовими ознаками. Якщо інформаційний приріст між предикторною та цільовою ознаками дорівнює 0, тоді їх відношення є не корельованим. Інформаційний приріст розраховується на основі ентропії.

Формула: G(X;Y)=H(X)-H(X|Y), де

G(X;Y)=інформаційний приріст для X і Y.

H(X) – ентропія для X

H(X|Y) – умовна ентропія для X при Y.

Одиниці результату вимірюються в бітах.

Таким чином можна отримати інформацію про більш важливіші ознаки.

Практичний приклад коду отримання інформації в задачі класифікації.

```
import pandas as pd
from sklearn.feature_selection import
mutual_info_classif
import matplotlib.pyplot as plt
from sklearn.feature_selection import SelectKBest
df=pd.read_csv("D:/xyz.csv")
df.head()

"""
Всі предикторні ознаки знаходяться в
independent_feature, а цільова в dependent_feature
"""
independent_feature=fdf.drop("target",axis=1)
dependent_feature=fdf["target"]

#Розрахунок mutual info інформаційного приросту
mutual_info=mutual_info_classif(independent_feature,de
pendent_feature)

'''
    Перетворимо дані на ряди та відсортуємо їх у
```

```
порядку спадання.
'''
mutual_info=pd.Series(mutual_info)
mutual_info.index=independent_feature.columns
mutual_info.sort_values(ascending=False)

Створимо гістограму зі спільною інформацією про
функції, щоб побачити, які стовпці важливіші
mutual_info.sort_values(ascending=False).plot.bar(figs
ize=(20,8))

"""
Тепер буде взято ті предикторні ознаки, які сильно
корелюють із залежними. Для цього скористаємось класом
SelectKBest із sklearn. Тут SelectKBest вибере k
найпопулярніших функцій.
"""
select_best_fea=SelectKBest(mutual_info_classif,k=8)
select_best_fea.fit(independent_feature,dependent_feat
ure)
independent_feature.columns[select_best_fea.get_suppor
t()]

'''
Для кращої точності візьміть ті характеристики, які
важливіші. У списку, який отримуємо після використання
get_support(), є список важливих функцій, вибраних
SelectKBest. Тепер використовуйте їх для навчання
'''

best_feature_data=fdf[['sex', 'cp', 'chol', 'thalach',
'exang', 'oldpeak', 'ca', 'thal']]
```

Практичний приклад коду отримання інформації в задачі регресії

```
import pandas as pd
from sklearn.feature_selection import
mutual_info_regression
import matplotlib.pyplot as plt
from sklearn.feature_sele ction import SelectKBest
```

```
df=pd.read_csv("D:/USa_house_price.csv")
df.head()

independent_feature=fdf.drop("target",axis=1)
dependent_feature=fdf["target"]

mutual_info=mutual_info_regression(independent_feature
,dependent_feature)

mutual_info=pd.Series(mutual_info)
mutual_info.index=independent_feature.columns
mutual_info.sort_values(ascending=False)

mutual_info.sort_values(ascending=False).plot.bar(figs
ize=(20,8))

"""
Буде обрано ті предикторні ознаки, які сильно
корелюють із цільовою. Для цього скористайтеся класом
SelectKBest із sklearn. SelectKBest вибере k найкращих
ознак.
"""
select_best_fea=SelectKBest(mutual_info_classif,k=8)
select_best_fea.fit(independent_feature,dependent_feat
ure)
independent_feature.columns[select_best_fea.get_suppor
t()]

best_feature_data=fdf[["battery_power", "blue",
"clock_speed", "dual_sim", "fc", "four_g",
"int_memory" ]]
```

ТЕСТ ХІ-КВАДРАТ ВІДБОРУ ОЗНАК

Цей підхід використовується для категоріальних ознак у задачі класифікації, при цьому досліджується зв’язок між категоріальними ознаками та цільовою ознакою. Тест хі-квадрат дає два значення: одне – F-оцінка, а інше – p-значення. Більш важливою є та ознака, яка має високу оцінку F та менше p-значення.

```
import pandas as pd
from sklearn.preprocessing import LabelEncoder
from sklearn.feature_selection import chi2
fdf=pd.read_csv("data_set.csv")
fdf.head()

fdf.info()

#Відбір тільки категоріальний ознак

categorical_features=fdf[["sex","embarked","alone"]]
categorical_features
target_features=fdf["survived"]

'''
Перекодовуємо ознак
'''
encoding=LabelEncoder()
categorical_features["sex"]=encoding.fit_transform(ca
tegorical_features["sex"])
categorical_features["embarked"]=encoding.fit_transfo
rm(categorical_features["embarked"])
categorical_features["alone"]=encoding.fit_transform(
categorical_features["alone"])
#Використовуємо chi2 function

chi_values=chi2(categorical_features,target_features)

#Друкуємо p-значення і сортуємо в порядку зростання.

p_values=pd.Series(chi_values[1])
p_values.index=categorical_features.columns
p_values.sort_index(ascending=True)
p_values

#Друк F-оцінки та сортування у порядку зростання.

f_score_values=pd.Series(chi_values[0])
f_score_values.index=categorical_features.columns
f_score_values.sort_index(ascending=False)
f_score_values
'''
Чим більше значення оцінки F, тим важливіша ознака.
Чим менше значення P, тим важливіша ознака.
```

```
Тепер відповідно до оцінки F і значення P
відфільтруйте ознаки для нового фрейму даних і
навчання моделі.
'''
```

ВІДБІР ОЗНАК: ТЕХНІКА ВАЖЛИВОСТІ ОЗНАК

Цей підхід дає оцінку для кожної ознаки наших даних. Якщо оцінка висока, то ознака є важливішою.

```
import pandas as pd
df=pd.read_csv("data_set.csv")
pd.set_option("display.max_columns",None)
df.head()

"""
 X містить всі незалежні змінні, а Y - цільову
"""
X=enco_value.drop("Vote",axis=1)
Y=enco_value["Vote"]

from sklearn.model_selection import train_test_split
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,te
st_size=0.3,random_state=0)

from sklearn.ensemble import ExtraTreesClassifier
import matplotlib.pyplot as plt
model=ExtraTreesClassifier()
model.fit(X,Y)

#Перегляньте оцінку, та нанесіть її на графік

ranked_score=pd.Series(model.feature_importances_,ind
ex=X.columns)
ranked_score.nlargest (5).plot(kind="barh")
print(ranked_score)
plt.show()
```

ВІДБІР ОЗНАК МЕТОДОМ WRAPPER

У техніці Wrapper згортаємо ознаки разом і знаходимо яка комбінація підходить для моделі.

Наприклад, є 5 ознак. Спочатку використовуємо комбінацію 1, 2 і 4 ознак і надсилаємо її моделі. Тоді візьмемо 3, 4 ознаки та надішлемо їх у модель, потім 2, 3 і 5. Отже, таким чином ми створюємо всеможливі комбінації фіч та передаємо їх в модель, намагаючись знайти найкращу підмножину, яка забезпечує кращу точність моделі. Припустимо, що отримуємо ознаки з номерами 1, 4 і 5, отже, інші відкидаємо.

Існує три різні методи пошуку найкращої комбінації ознак.

1. Прямий метод

Крок 1: спочатку обирається рівень значущості.

Крок 2: випадковим чином обираємо ознаки і окремо передамо їх моделі для тестування. Цей метод вибере ту із них, яка має мінімальне значення p (наприклад, №3).

Крок 3: на другій ітерації метод спробує створити комбінацію решти ознак.

Наприклад, ознака № 3 обрана, спочатку буде випадково взято іншу ознаку, наприклад, №1 та передано моделі для пошуку найкращої їхньої комбінації відповідно до мінімального p-значення. Припустимо, що потрібна комбінація ознак із номерами 3, 1.

Крок 4: формуються комбінації трьох ознак. Якщо розглядається лише 4 ознаки, а p-значення попередньої комбінації менше рівня значущості, це і буде останньою ітерацією.

2. Зворотний метод:

Крок 1: спочатку обирається рівень значущості.

Потім починаємо з повної моделі (включаючи всі предикторні ознаки).

Наприклад: F1+F2+F3+F4, F1+F2+F4+F5, F1+F3+F4+F5, F2+F3+F4+F5.

Далі будуть видалені ті комбінації, у яких сума p-значення кожної ознаки більша за рівень значущості.

Припустимо, комбінацію F1+F2+F3+F4 обрано.

Крок 2: розглядаємо різні комбінації із трьох ознак.

Наприклад: F1+F2+F3, F1+F2+F4, F1+F3+F4, F2+F3+F4.

Обираємо ті комбінації, у яких сумарне значення p для кожної ознаки не перевищує рівень значущості.

Таким чином, цей процес відбуватиметься, доки не отримаємо остаточний набір важливих ознак.

3. Двонаправлений метод: це поєднання прямого відбору та зворотного усунення.

Щоб додати ознаку до комбінації, ми використовуємо прямий метод, а щоб вилучити з моделі – зворотній.

Оберіть рівень значущості для елемента або додайте ознаку до комбінації. Якщо p-значення комбінації менше рівня значущості, то вона додається, а якщо більше, то усувається. Цей процес відбуватиметься, поки не отримаємо остаточний оптимальний набір ознак.

```
#Install mlxtend !pip install mlxtend

import pandas as pd
from mlxtend.feature_selection import
SequentialFeatureSelection as SFS
from sklearn.neighbors import KNeighborsClassifier

fdf=pd.read_csv("data_set.csv")
fdf.head()
```

```
'''
Розділення на залежні і незалежні ознаки
'''
independent_feature=fdf.drop("target",axis=1)
dependent_feature=fdf["target"]

#Створення деякої моделі
model=KNeighborsClassifier(n_neighbors=4)

'''
Створення змінних використовуючи
SequentialFeatureSelection
'''
sfs=SFS(model, k_feature=3, forward=False,
floating=False, scoring="accuracy", cv=2)
'''
Параметри:
1. model:
В якості моделі взято метод найближчих сусідів
2. forward and floating:
Можуть бути обрані два значення False or True. Якщо
використати True в forward і False для floating, тоді
комбінація відбуватиметься прямим методом. В
протилежному випадку - зворотнім. Якщо обрати True
для обох параметрів тоді два направления методом.
3. k_feature:
Вказує кількість ознак, які потрібно вибрати. Це може
бути випадкове значення.
4. cv:
кількість папок k-кратної перехресної перевірки.
5. scoring:
Визначає критерій оцінки, який потрібно
використовувати. Для задачі регресії використовуємо
r2 score. Для задач класифікація - accuracy,
precision, recall, f1-score.
'''

feature_names=("sepal length","sepal width","petal
length","petal width" ) '''
Підбір independent_feature, dependent_feature,
feature names у SFS змінній
'''
sfs=sfs.fit(independent_feature,dependent_feature,
```

```
custom_feature_names=feature_names)

result_df=pd.DataFrame.from_dict(sfs.get_metric_dict(
))

'''
Оберіть ті функції, комбінація яких забезпечує хорошу
точність, і створіть нові дані для навчання моделі.
'''
```

МАСШТАБУВАННЯ ОЗНАК У МАШИННОМУ НАВЧАННІ

Масштабування ознак – це метод приведення вимірювання різних ознак в одному діапазоні чи масштабі (наприклад, від -1 до 1, від 0 до 1). Ця техніка називається нормалізацією даних.

Шкала необроблених ознак відрізняється залежно від одиниць вимірювання, наприклад, деякі ознаки можуть визначатись в метрах, інші - в кілограмах або футах, але алгоритми машинного навчання не “розуміють” одиниць вимірювання, вони “розуміють” лише числові значення. Отже, якщо одиниці ознак не співрозмірні, це погано вплине на роботу моделі машинного навчання, і, як наслідок, модель машинного навчання не буде продуктивною.

Наприклад, наведено зріст двох людей, зріст першої становить 150 см, а другої – 8,2 фута. Зрозуміло, що вища зі зростом 8,2 фута, але модель машинного навчання обере за більшим числовим значенням – 150 см. Щоб вирішити цю проблему, необхідно звести одиниці вимірювання в одну шкалу або діапазон.

Типи масштабування ознак:

1. Мінімаксне (MinMaxScaler, MaxAbsScaler).
2. Стандартне (Standard scaler).
3. Робастне (Robust scaler).

СТАНДАРТИЗАЦІЯ

Стандартизація змінює масштаб ознаки так, щоб її середнє значення дорівнювало 0, а стандартне відхилення - 1.

Формула: $z=(x- \mu)/\sigma$, де

x- значення, яке потрібно масштабувати з певного стовпця;

μ - середнє значення цього стовпця;

σ - середньоквадратичне відхилення стовпця.

Зазвичай, воно використовується, коли дані відповідають нормальному/гаусовому/дзвоноподібному розподілу.

```
import pandas as pd
from sklearn.preprocessing import StandardScaler

df=pd.read_csv("my_bill.csv")
'''
dataset link:
https://www.kaggle.com/code/yashsharmabharatpur/tips-collected-at-a-restaurant/data
'''
XX=df.drop("Total_bill",axis=1)
sc=StandardScaler()
XX_scaled=sc.fit_transform(XX)
X=XX_scaled
y=df["Total_bill"]
```

НОРМАЛІЗАЦІЯ

Нормалізація (MinMaxScaler, MaxAbsScaler) масштабує ознаку у фіксованому діапазоні від 0 до 1 або від -1 до 1. Нормалізація також називається мінімально-максимальним масштабуванням. Зазвичай, якщо дані не відповідають нормальному/дзвіноподібному/гаусовому розподілу, використовується нормалізація.

Формула: $z=(x-x_{min})/(x_{max}-x_{min})$, де

x - це значення, яке потрібно масштабувати з певного стовпця;

x_{min} - мінімальне значення цього стовпця;

x_{max} - максимальне значення цього стовпця.

```
import pandas as pd
```

```
from sklearn.preprocessing import MinMaxScaler
df=pd.read_csv("my_bill.csv")
'''
dataset link:
https://www.kaggle.com/code/yashsharmabharatpur/tips-collected-at-a-restaurant/data
'''
XX=df.drop("Total_bill",axis=1)
mms=MinMaxScaler()
mms.fit(XX)
XX_scaled=mms.transform(XX)
X=XX_scaled
Y=df["Total_bill"]
```

Стандартизація vs нормалізація

Стандартизація	Нормалізація
Тут середнє завжди буде 0, а стандартне відхилення буде 1	Тут ми завжди масштабуємо дані від 0 до 1 або від -1 до 1
Немає жодного основного правила використання стандартизації	Немає жодного основного правила використання стандартизації
Здебільшого стандартизація використовується для аналізу кластеризації, алгоритмів, пов'язаних із відстанню, тощо, коли наші дані відповідають нормальному/гауссовому/ дзвоноподібному розподілу.	Нормалізація здебільшого використовується для попередньої обробки зображень, оскільки там інтенсивність пікселів становить від 0 до 255, дані алгоритмів нейронної мережі в масштабі 0-1, K-найближчі сусіди, а також коли наші дані не відповідають нормальному/гаусовому/ дзвоноподібному розподілу.

РОБАСТНЕ МАСШТАБУВАННЯ

Це одна з найкращих технік масштабування. Якщо у наборі даних є викиди, використовують цю техніку. Вона масштабує дані відповідно до інтерквартильного діапазону (IQR = 75 Quartile – 25 Quartile). Тут IQR – це діапазон між 1-м квартилем (25-й квартиль) і 3-м квартилем (75-й

квартиль). У цьому методі ми віднімаємо всі значення ознаки від значення медіани, а потім ділимо на значення IQR (міжквартильний діапазон).

```
import pandas as pd
from sklearn.preprocessing import RobustScale

df=pd.read_csv("my_bill.csv")
'''
dataset link:
https://www.kaggle.com/code/yashsharmabharatpur/tips-collected-at-a-restaurant/data
'''
XX=df.drop("Total_bill",axis=1)
rs=RobustScale()
rs.fit(XX)
XX_scaled=rs.transform(XX)

X=XX_scaled
Y=df["Total_bill"]
```

ЛІТЕРАТУРА

1. Штовба С.Д. Machine learning: стартовий курс : електронний навчальний посібник / Штовба С.Д., Козачко О.М. – Вінниця : ВНТУ, 2020. – 81 с. URL: https://www.researchgate.net/publication/338924246_Machine_Learning_startovij_kurs

2. Путівник мовою програмування Python. URL: https://pythonguide.rozh2sch.org.ua/

3. Засоби підготовки та аналізу даних. Лабораторний практикум [Електронний ресурс] : навч. посіб. для студ. спеціальності 113 «Прикладна математика» / А. Ю. Шелестов, Н. М. Куссуль; КПІ ім. Ігоря Сікорського. – Електронні текстові дані (1 файл: 1086 Кбайт). – Київ : КПІ ім. Ігоря Сікорського, 2021. – 31 с. URL: https://ela.kpi.ua/bitstream/123456789/43491/1/Shelestov_Zasoby-pidhotovky-ta-analizu-danykh_LabPrakt.pdf

4. Data Mining and Data Warehousing: Principles and Practical Techniques. Front Cover /Parteek Bhatia. Cambridge University Press, 2019. – 513 p. URL: https://cutt.ly/s2DP9Ds

5. Data Preprocessing in Machine learning. URL: https://www.javatpoint.com/data-preprocessing-machine-learning

6. A Simple Guide to Data Preprocessing in Machine Learning. URL: https://www.v7labs.com/blog/data-preprocessing-guide

7. Introduction to Data Preprocessing in Machine Learning. URL: https://towardsdatascience.com/introduction-to-data-preprocessing-in-machine-learning-a9fa83a5dc9d