Міністерство освіти і науки України

Національний технічний університет України

“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Лабораторна робота №7

Аналіз даних з використанням мови Python

Тема: Кластеризація та регресія в scikit-learn

Варіант: 1

Виконав

студент групи ІП-11:

Панченко С. В.

Перевірив:

Тимофєєва Ю. С

Київ 2023

ЗМІСТ

1 МЕТА ЛАБОРАТОРНОЇ РОБОТИ

Ознайомитись з побудовою моделей для вирішення задач регресії та кластеризації в scikit-learn, визначити основні оцінки цих моделей.

2 ЗАВДАННЯ

Написати програму,яка здійснює попередню обробку даних, навчає та тестує(при потребі) модель, що виконує завдання відповідно до варіанту, оцінити модель за допомогою відповідних метрик.

Оформити звіт. Звіт повинен містити:

—титульний лист;

—код програми;

—результати виконання коду.

Продемонструвати роботу програми та відповісти на питання стосовно теоретичних відомостей та роботи програми.

Варіант 1: lab2/Crime.csv. Використати будь-яку комбінацію незалежних ознак (не менше двох), щоб спрогнозувати рівень злочинності в даний час.

3 ВИКОНАННЯ

3.1 Перетворення даних

Для початку імпортуємо модулі pandas, numpy, seaborn, matplotlib. Завантажимо датафрейм.

```
In [70]:
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import math
df = pd.read_csv('data/Crime.csv')
df.head()
```

Out[70]:

	CrimeRate	Youth	Southern	Education	ExpenditureYear0	LabourForce	Males	MoreMales	StateSize	YouthUne
0	45.5	135	0	12.4	69	540	965	0	6	
1	52.3	140	0	10.9	55	535	1045	1	6	
2	56.6	157	1	11.2	47	512	962	0	22	
3	60.3	139	1	11.9	46	480	968	0	19	
4	64.2	126	0	12.2	106	599	989	0	40	

5 rows × 27 columns

Рисунок 3.1 - Завантаження датафрейму

Переставимо колонку у "Southern" на перше місце, оскільки розташування штату не змінювався.

```
In [71]:
southern_name = 'Southern'
southern = df.pop(southern_name)
df.insert(0, southern_name, southern)
df.head()
```

Out[71]:

	Southern	CrimeRate	Youth	Education	ExpenditureYear0	LabourForce	Males	MoreMales	StateSize	YouthUne
0	0	45.5	135	12.4	69	540	965	0	6	
1	0	52.3	140	10.9	55	535	1045	1	6	
2	1	56.6	157	11.2	47	512	962	0	22	
3	1	60.3	139	11.9	46	480	968	0	19	
4	0	64.2	126	12.2	106	599	989	0	40	

5 rows × 27 columns

Рисунок 3.2 - Переставлення колонки

Переставимо показники через 10 років у звичайні колонки, розділивши датафрейм надвоє та з'єднавши частини вертикально. Таким чином ми збільшуємо розмір вибірки у два рази, тому модель матиме більше даних, аніж у тому випадку якби ми знаходили, наприклад, середнє арифметичне між колонками за 10 років.

```
In [72]: df10 = df.loc[:, 'CrimeRate10':]
         df.drop(df.columns[14:], axis=1, inplace=True)
         df10.insert(0, southern_name, southern)
         df10.columns = df.columns
         df = pd.concat([df, df10], axis=0, ignore_index=True)
         df.rename(columns={'ExpenditureYear0': 'ExpenditureYear'}, inplace=True)
         rate = 'CrimeRate'
         rate_s = df.pop(rate)
         df.insert(0, rate, rate_s)

         df
```

Out[72]:

	CrimeRate	Southern	Youth	Education	ExpenditureYear	LabourForce	Males	MoreMales	StateSize	YouthUne
0	45.5	0	135	12.4	69	540	965	0	6	

Рисунок 3.3 - Перетворення даних

Розділимо дані на аргументи та значення.

```
In [73]: df_x = df.iloc[:, 1:]
         df_y = df.iloc[:, 0]
```

Рисунок 3.4 - Розділення даних на аргументи та значення

Зробимо масштабування даних за допомогою MinMaxScaler.

```
In [74]: from sklearn.preprocessing import MinMaxScaler
         scaler = MinMaxScaler()
         df_x = scaler.fit_transform(df_x)
         df
```

Out[74]:

	CrimeRate	Southern	Youth	Education	ExpenditureYear	LabourForce	Males	MoreMales	StateSize	YouthUne
0	45.5	0	135	12.4	69	540	965	0	6	
1	52.3	0	140	10.9	55	535	1045	1	6	
2	56.6	1	157	11.2	47	512	962	0	22	
3	60.3	1	139	11.9	46	480	968	0	19	
4	64.2	0	126	12.2	106	599	989	0	40	
...	...	...	...	...	...	...	...	...	...	
89	157.3	1	131	12.1	109	548	976	0	52	
90	162.7	0	142	12.2	95	612	1003	1	13	
91	169.6	0	134	12.2	116	580	987	0	104	
92	177.2	0	140	15.2	141	578	995	0	160	
93	178.2	0	132	13.2	143	632	1058	1	4	

94 rows × 14 columns

Рисунок 3.5 - Масштабування даних

3.2 Навчання та тестування моделі

Зараз побудуємо модель, де будемо використовувати одночасно декілька алгоритмів для того, щоб мати кращий результат. До прикладу, оберемо SVR, LinearRegression, DecisionTree та RandomForest. Для цього імпортуємо з пакету sklearn.ensemble StackingClassifier та RandomForestRegressor, з sklearn.svr - SVR, з sklearn.linear_model - LinearRegression, з sklearn.tree - DecisionTreeRegressor.

Напишемо функцію model_factory та проведемо тренування n-кількість разів, зберігатимемо найкращу з моделей. Також будемо всередині використовувати функцію train_test_split, яка ділить датасет на 75% тренувальних та 25% тестових даних за замовчуванням.

Оцінювання буде проводитися параметр R2, або коефіцієнт детермінації, який кількісно визначає частку дисперсії залежної змінної, яку можна передбачити на основі незалежних змінних. Також використаємо середню квадратичну похибку та середню абсолютну.

Проведемо певну кількість тренувань моделі та оберемо найкращу.

In [75]:
```
from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error
from sklearn.svm import SVR
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import StackingRegressor, RandomForestRegressor
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split

def model_factory(df_x, df_y):
    samples = train_test_split(df_x, df_y)
    x_train, x_test, y_train, y_test = samples
    estimators = [
        ('svr', SVR()),
        ('lin', LinearRegression()),
        ('dt', DecisionTreeRegressor()),
        ('rf', RandomForestRegressor()),
    ]
    model = StackingRegressor(estimators=estimators)
    model.fit(x_train, y_train)
    y_train_pred = model.predict(x_train)
    y_test_pred = model.predict(x_test)
    scores = {
        'r2_train': r2_score(y_train, y_train_pred),
        'r2_test': r2_score(y_test, y_test_pred),
        'mse_train': mean_squared_error(y_train, y_train_pred),
        'mse_test': mean_squared_error(y_test, y_test_pred),
        'mae_train': mean_absolute_error(y_train, y_train_pred),
        'mae_test': mean_absolute_error(y_test, y_test_pred)
    }
    return scores

def give_best(df_x, df_y, n_times):
    results = []
    for i in range(n_times):
        results.append(model_factory(df_x, df_y))
    results = sorted(results, key=lambda x: x['r2_test'])
    for k, v in results[-1].items():
        print(f'{k}: {v}')

give_best(df_x, df_y, 50)
```

```
r2_train: 0.8901163649854281
r2_test: 0.7863732036052896
mse_train: 130.68975505900463
mse_test: 232.12945457748356
mae_train: 9.051193435279407
mae_test: 11.301232010635758
```

Рисунок 3.6 - Тренування моделей

3.3 Зменшення вимірів (Dimensionality reduction)

Однак дана модель має один мінус. У ній забагато даних обробляється. І, можливо, результати можна покращити, зробивши dimesionality reduction. Для початку побудуємо матрицю кореляцій.

```
In [76]: def corr_map(df, figsize):
             fig, axis = plt.subplots(figsize=figsize)
             axis.set_title('Кореляція між факторами')
             sns.heatmap(df.corr(), ax=axis, annot=True)
         corr_map(df, (10, 6))
```

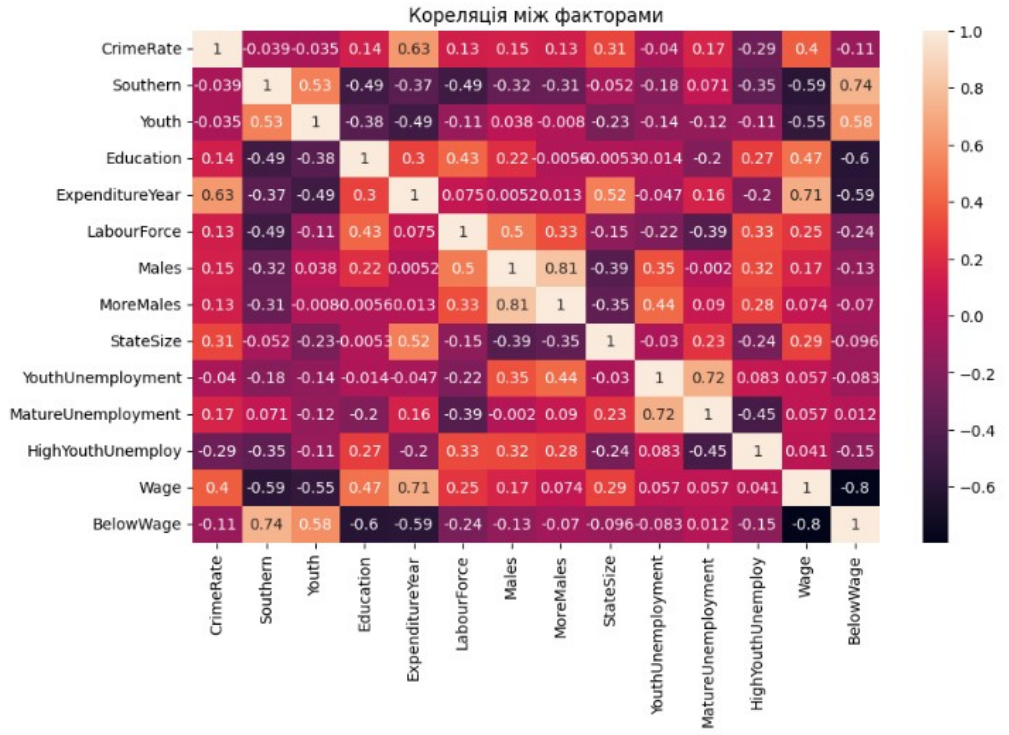

Рисунок 3.7 - Матриця кореляцій

Як бачимо, матриця має доволі багато взаємозалежних величин, тому зменшимо їхню кількість. Для цього використаємо PCA - Principal Component Analysis, або метод головних компонент. В Імпортуємо відповідний пакет PCA з sklearn.decomposition.

```
In [78]: from sklearn.decomposition import PCA
         pca = PCA(n_components=10, whiten=True)
         df_x = pca.fit_transform(df_x)
         give_best(df_x, df_y, 50)

r2_train: 0.9131434605964489
r2_test: 0.7362289389721255
mse_train: 100.57064527417907
mse_test: 310.0304557993231
mae_train: 7.834756072084398
mae_test: 14.493882900141521
```

Рисунок 3.8 - Зменшення залежних величин

Як бачимо, точність для тестових даних погіршилася, а тому сенсу зменшувати далі нема, оскільки величини сильно взаємозалежні.

4 ВИСНОВОК

Під час виконання даної лабораторної роботи я ознайомився з побудовою моделей для вирішення задач регресії та кластеризації в scikit-learn, визначив основні оцінки цих моделей.

По-перше, перетворив дані: перемістив інформацію зі стовпчиків, яка була зафіксована через 10 років, у звичайні стовпчики, таким чином зменшивши їхню кількість.

По-друге, зробив масштабування аргументів за допомогою MinMaxScaler.

По-третє, спроектував багатошарову модель з використанням таких алгоритмів як: SVR, DecisionTreeRegressor, RandomForestRegressor, LinearRegression.

По-четверте, розділив дані на навчальні та тестові у відношенні до основного датасету як 75% і 25% відповідно.

По-п’яте, для оцінки якості моделі використав параметр R2, середню квадратичну та абсолютну похибки.

По-шосте, побудував матрицю кореляцій та спробував зменшити кількість стовпчиків за допомогою PCA. Однак результати виявилися гіршими ніж з використанням усіх даних.