КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
імені ТАРАСА ШЕВЧЕНКА

ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра прикладних інформаційних систем

Лабораторна робота №2

з курсу

«Технології обробки даних»

Студента 5 курсу
групи ІС-11
спеціальності 122 «Комп'ютерні науки»
ОП «Інформаційні системи»
Іванова Віктора Миколайовича

Київ – 2023

Пайплайни в scikit-learn як засіб автоматизації дослідження

1. Завантажте датасет (наприклад, Iris, Wine або Breast Cancer) і поділіть його на тренувальну та тестову вибірки. Для цього використайте функцію train_test_split з бібліотеки scikit-learn.
2. Створіть пайплайн, який включає предобработку даних (наприклад, масштабування за допомогою StandardScaler) та відбір ознак (наприклад, використовуючи метод головних компонент, PCA).
3. Додайте до пайплайну модель класифікації або регресії (залежно від обраного датасету), таку як LogisticRegression, RandomForestClassifier чи LinearRegression.
4. Використайте GridSearchCV або RandomizedSearchCV для налаштування гіперпараметрів моделі та відбору ознак. Визначте список гіперпараметрів, які ви хочете налаштувати, та відповідні значення, які ви хочете перевірити.
5. Навчіть пайплайн на тренувальній вибірці та оцініть його якість на тестовій вибірці, використовуючи метрики, такі як точність (accuracy), точність (precision), повнота (recall) або коефіцієнт детермінації (R^2), залежно від типу задачі.
6. Порівняйте результати пайплайну з базовою моделлю, яка не використовує пайплайн. Це допоможе вам оцінити ефективність використання пайплайнів у вашому дослідженні.
7. Збережіть навчений пайплайн за допомогою бібліотеки joblib або pickle та завантажте його для подальшого використання.

Хід роботи

Встановимо бібліотеку scikit-learn за допомогою наступної команди:

```
pip install scikit-learn
```

1. Завантажимо датасет та поділимо його на тренувальну та тестову вибірки за допомогою функції **train_test_split** з бібліотеки scikit-learn:

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split

# Завантаження датасету Iris
iris = load_iris()

# Поділ датасету на тренувальну та тестову вибірки
X_train, X_test, y_train, y_test = train_test_split(iris.data, iris.target, test_size=0.2,
random_state=42)
```

2. Створимо пайплайн, який включає предобработку даних та відбір ознак. Наприклад, використовуємо масштабування за допомогою **StandardScaler** та метод головних компонент (PCA):

```
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.pipeline import Pipeline

# Створення пайплайну
pipeline = Pipeline([
   ('scaler', StandardScaler()),  # Масштабування даних
   ('pca', PCA(n_components=2)),  # Відбір 2 головних компонент
])
```

3. Додамо до пайплайну модель класифікації або регресії, залежно від обраного датасету. Наприклад, використовуємо **LogisticRegression** для класифікації:

```
from sklearn.linear_model import LogisticRegression

# Додавання моделі класифікації до пайплайну
pipeline.steps.append(('classifier', LogisticRegression()))
```

4. Використовуємо **GridSearchCV** або **RandomizedSearchCV** для налаштування гіперпараметрів моделі та відбору ознак. Визначимо список гіперпараметрів та значень, які ви хочете перевірити. Наприклад, для **LogisticRegression** можемо налаштувати параметр **C** та значення **[0.1, 1, 10]**:

```
from sklearn.model_selection import GridSearchCV

# Список гіперпараметрів та значень для перевірки
param_grid = {
    'classifier__C': [0.1, 1, 10],
}

# GridSearchCV для налаштування гіперпараметрів
grid_search = GridSearchCV(pipeline, param_grid, cv=5)
grid_search.fit(X_train, y_train)
```

5. Навчимо пайплайн на тренувальній вибірці та оцінимо його якість на тестовій вибірці за допомогою вибраних метрик. Наприклад, для класифікації використаємо метрику точності (accuracy):

```
from sklearn.metrics import accuracy_score
```

```
# Передбачення на тестовій вибірці
y_pred = grid_search.predict(X_test)

# Оцінка якості моделі
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

6. Порівняємо результати пайплайну з базовою моделлю, яка не використовує пайплайн. Це допоможе нам оцінити ефективність використання пайплайнів у вашому дослідженні.
7. Збереж навчений пайплайн за допомогою бібліотеки **joblib** або **pickle** для подальшого використання:

```
import joblib

# Збереження пайплайну
joblib.dump(grid_search, 'pipeline.pkl')

# Завантаження пайплайну
loaded_pipeline = joblib.load('pipeline.pkl')
```