Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования «Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)» (МГТУ им. Н.Э. Баумана)

Факультет «Информатика и системы управления»

Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Технологии машинного обучения»

Лабораторная работа №3

Выполнил:
студент группы ИУ5-64Б
Коваленко Г. В.

Проверил:
Гапанюк Ю. Е.

2024 г.

Качество Вина

Задание:

- Выберите набор данных (датасет) для решения задачи классификации или регрессии.
- В случае необходимости проведите удаление или заполнение пропусков и кодирование категориальных признаков.
- С использованием метода train_test_split разделите выборку на обучающую и тестовую.
- Обучите модель ближайших соседей для произвольно заданного гиперпараметра K. Оцените качество модели с помощью подходящих для задачи метрик.
- Произведите подбор гиперпараметра K с использованием GridSearchCV и RandomizedSearchCV и кросс-валидации, оцените качество оптимальной модели. Используйте не менее двух стратегий кросс-валидации.
- Сравните метрики качества исходной и оптимальной моделей.

Ссылка на датасет:
https://www.kaggle.com/datasets/uciml/red-wine-quality-cortez-et-al-2009

Описание

```
!unzip "archive(3).zip"
```

```
Archive:  archive(3).zip
  inflating: winequality-red.csv
```

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

from sklearn.model_selection import train_test_split, GridSearchCV,
RandomizedSearchCV, cross_val_score
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report, accuracy_score

import warnings
warnings.filterwarnings('ignore')

df = pd.read_csv('winequality-red.csv')

df.head()
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8	5
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8	5
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9.8	6
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5

```
df.describe()
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
count	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000
mean	8.319637	0.527821	0.270976	2.538806	0.087467	15.874922	46.467792	0.996747	3.311113	0.658149	10.422983	5.636023
std	1.741096	0.179060	0.194801	1.409928	0.047065	10.460157	32.895324	0.001887	0.154386	0.169507	1.065668	0.807569
min	4.600000	0.120000	0.000000	0.900000	0.012000	1.000000	6.000000	0.990070	2.740000	0.330000	8.400000	3.000000
25%	7.100000	0.390000	0.090000	1.900000	0.070000	7.000000	22.000000	0.995600	3.210000	0.550000	9.500000	5.000000
50%	7.900000	0.520000	0.260000	2.200000	0.079000	14.000000	38.000000	0.996750	3.310000	0.620000	10.200000	6.000000
75%	9.200000	0.640000	0.420000	2.600000	0.090000	21.000000	62.000000	0.997835	3.400000	0.730000	11.100000	6.000000
max	15.900000	1.580000	1.000000	15.500000	0.611000	72.000000	289.000000	1.003690	4.010000	2.000000	14.900000	8.000000

```
df.shape, df.quality.shape
```

```
((1599, 12), (1599,))
```

```
X = df.drop("quality", axis=1)
y = df["quality"]
```

```
from sklearn.model_selection import train_test_split
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=1)
```

```
np.unique(y_test)
```

```
array([3, 4, 5, 6, 7, 8])
```

```
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train, y_train)
```

```
KNeighborsClassifier(n_neighbors=3)
```

```
y_pred = knn.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
Accuracy: 0.5275
              precision    recall  f1-score   support

           3       0.00      0.00      0.00         1
           4       0.08      0.06      0.07        16
           5       0.57      0.71      0.63       171
           6       0.53      0.44      0.48       167
           7       0.46      0.38      0.42        42
           8       0.00      0.00      0.00         3
```

```
    accuracy                           0.53       400
   macro avg       0.27      0.26      0.27       400
weighted avg       0.52      0.53      0.52       400
```

```
# Подбор гиперпараметра K с использованием GridSearchCV
param_grid = {'n_neighbors': range(1, 31)}
grid_search = GridSearchCV(KNeighborsClassifier(), param_grid, cv=5)
grid_search.fit(X_train, y_train)
print("Best parameters (GridSearchCV):", grid_search.best_params_)
print("Best cross-validation score (GridSearchCV):", grid_search.best_score_)
```

```
Best parameters (GridSearchCV): {'n_neighbors': 1}
Best cross-validation score (GridSearchCV): 0.5462935843793584
```

```
# Подбор гиперпараметра K с использованием RandomizedSearchCV
random_search = RandomizedSearchCV(KNeighborsClassifier(), param_grid,
n_iter=10, cv=5, random_state=42)
random_search.fit(X_train, y_train)
print("Best parameters (RandomizedSearchCV):", random_search.best_params_)
print("Best cross-validation score (RandomizedSearchCV):",
random_search.best_score_)
```

```
Best parameters (RandomizedSearchCV): {'n_neighbors': 1}
Best cross-validation score (RandomizedSearchCV): 0.5462935843793584
```

```
# Оценка оптимальной модели
best_knn = grid_search.best_estimator_
y_pred_opt = best_knn.predict(X_test)
print("Accuracy (optimized):", accuracy_score(y_test, y_pred_opt))
print(classification_report(y_test, y_pred_opt))
```

```
Accuracy (optimized): 0.5525
              precision    recall  f1-score   support

           3       0.00      0.00      0.00         1
           4       0.05      0.06      0.06        16
           5       0.62      0.67      0.64       171
           6       0.58      0.53      0.55       167
           7       0.47      0.40      0.44        42
           8       0.11      0.33      0.17         3

    accuracy                           0.55       400
   macro avg       0.31      0.33      0.31       400
weighted avg       0.56      0.55      0.55       400
```