Московский государственный технический университет им. Н. Э. Баумана

Курс «Технологии машинного обучения»

Отчёт по лабораторной работе №3

Выполнил:
Файзуллин К. Х.
группа ИУ5-64Б

Проверил:
Гапанюк Ю.Е.

Дата: 21.06.25

Подпись:

Дата:

Подпись:

Москва, 2025 г.

Цель лабораторной работы: изучение способов подготовки выборки и подбора гиперпараметров на примере метода ближайших соседей.

Задание:

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

Ход выполнения:

[11]:
```
import pandas as pd
from sklearn.datasets import load_wine
from sklearn.model_selection import train_test_split, GridSearchCV, RandomizedSearchCV, cross_val_score
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, classification_report
from sklearn.preprocessing import StandardScaler
```

[21]:
```
wine = load_wine()
X = pd.DataFrame(wine.data, columns=wine.feature_names)
y = pd.Series(wine.target)
```

[23]:
```
y.value_counts()
```

[23]:
```
1    71
0    59
2    48
Name: count, dtype: int64
```

[13]:
```
X.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 178 entries, 0 to 177
Data columns (total 13 columns):
 #   Column                        Non-Null Count  Dtype  
---  ------                        --------------  -----  
 0   alcohol                       178 non-null    float64
 1   malic_acid                    178 non-null    float64
 2   ash                           178 non-null    float64
 3   alcalinity_of_ash             178 non-null    float64
 4   magnesium                     178 non-null    float64
 5   total_phenols                 178 non-null    float64
 6   flavanoids                    178 non-null    float64
 7   nonflavanoid_phenols          178 non-null    float64
 8   proanthocyanins               178 non-null    float64
 9   color_intensity               178 non-null    float64
 10  hue                           178 non-null    float64
 11  od280/od315_of_diluted_wines  178 non-null    float64
 12  proline                       178 non-null    float64
dtypes: float64(13)
memory usage: 18.2 KB
```

[15]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=52, shuffle=True)
```

[16]:
```
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

[17]:
```
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")
print(classification_report(y_test, y_pred))
```

```
Accuracy: 0.94
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        18
           1       1.00      0.88      0.93        24
           2       0.80      1.00      0.89        12

    accuracy                           0.94        54
   macro avg       0.93      0.96      0.94        54
weighted avg       0.96      0.94      0.95        54
```

[18]:
```
param_grid = {'n_neighbors': range(1, 15)}
grid_search = GridSearchCV(KNeighborsClassifier(), param_grid, cv=5)
grid_search.fit(X_train, y_train)

print(grid_search.best_params_)
print(grid_search.best_score_)


param_dist = {'n_neighbors': range(1, 15)}
random_search = RandomizedSearchCV(KNeighborsClassifier(), param_dist, n_iter=10, cv=5)
random_search.fit(X_train, y_train)

print(random_search.best_params_)
print(random_search.best_score_)
```

```
{'n_neighbors': 5}
0.9596666666666666
{'n_neighbors': 5}
0.9596666666666666
```

[19]:
```
best_knn = grid_search.best_estimator_
y_pred_best = best_knn.predict(X_test)


accuracy_best = accuracy_score(y_test, y_pred_best)
print(accuracy_best)
print(classification_report(y_test, y_pred_best))
```

```
0.9444444444444444
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        18
           1       1.00      0.88      0.93        24
           2       0.80      1.00      0.89        12

    accuracy                           0.94        54
   macro avg       0.93      0.96      0.94        54
weighted avg       0.96      0.94      0.95        54
```

[20]:
```
print(accuracy)
print(accuracy_best)
```

```
0.9444444444444444
0.9444444444444444
```