Московский государственный технический университет им. Н. Э. Баумана

Курс «Технологии машинного обучения»

Отчёт по лабораторной работе №3

Выполнил:
Каятский П. Е.
группа ИУ5-64Б

Проверил:
Гапанюк Ю.Е.

Дата: 07.04.25

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:

Подготовка обучающей и тестовой выборки, кросс-валидация и подбор гиперпараметров на примере метода ближайших соседей.

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

```
Executed at 2025.03.20 12:38:24 in 4ms

Загрузка датасета diabetes

[15]
```
dia = load_wine()
x = dia.data
y = dia.target
```
Executed at 2025.03.20 12:42:57 in 8ms

Преобразуем в DataFrame для удобства

[16]
```
df = pd.DataFrame(x, columns=dia.feature_names)
df['target'] = y
```
Executed at 2025.03.20 12:43:00 in 5ms

[17]
```
df.head()
```
Executed at 2025.03.20 12:43:01 in 16ms

5 rows | 5 rows × 14 columns | Static Output

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nc
0	14.23	1.71	2.43	15.6	127.0	2.80	3.06	
1	13.20	1.78	2.14	11.2	100.0	2.65	2.76	
2	13.16	2.36	2.67	18.6	101.0	2.80	3.24	
3	14.37	1.95	2.50	16.8	113.0	3.85	3.49	
4	13.24	2.59	2.87	21.0	118.0	2.80	2.69	

[18]
```
df.isnull().sum()
```
Executed at 2025.03.20 12:43:03 in 9ms

```
magnesium                     0
total_phenols                 0
flavanoids                    0
nonflavanoid_phenols          0
proanthocyanins               0
color_intensity               0
hue                           0
od280/od315_of_diluted_wines  0
proline                       0
target                        0
```

106

Масштабирование данных

```
[19] 1 scaler = StandardScaler()
     2 X_scaled = scaler.fit_transform(x)
```
Executed at 2025.03.20 12:43:06 in 6ms

```
[20] 1 X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.3, random_state=42)
```
Executed at 2025.03.20 12:43:09 in 6ms

```
[21] 1  # Обучаем модель с K=2
     2  knn = KNeighborsClassifier(n_neighbors=2) #создается экземпляр классификатора KNN с параметром n_neighbors=2,
          что означает, что при классификации нового объекта будут рассматриваться 2 ближайших соседа.
     3  knn.fit(X_train, y_train) #модель обучается на обучающей выборке X_train с соответствующими метками y_train.
     4
     5  # Предсказание классов на тестовой выборке
     6  y_pred = knn.predict(X_test)
     7
     8  # Оценка качества модели
     9  accuracy = accuracy_score(y_test, y_pred)
    10  print(f"Accuracy (K=2): {accuracy:.2f}")
    11  print(classification_report(y_test, y_pred))
```
Executed at 2025.03.20 12:43:09 in 20ms

```
Accuracy (K=2): 0.94
              precision    recall  f1-score   support

           0       0.90      1.00      0.95        19
           1       1.00      0.86      0.92        21
           2       0.93      1.00      0.97        14

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

Подбор гиперпараметра K с использованием GridSearchCV и RandomizedSearchCV

```
[27] 1  # Определение параметров для поиска
     2  param_grid = {'n_neighbors': np.arange(2, 20)}
     3
     4  # GridSearchCV -  Полный перебор
     5  grid_search = GridSearchCV(KNeighborsClassifier(), param_grid, cv=3, scoring='accuracy')
     6  grid_search.fit(X_train, y_train)
     7  print(f"Лучший параметр K (GridSearchCV): {grid_search.best_params_}")
     8  print(f"Лучшая точность (GridSearchCV): {grid_search.best_score_:.2f}")
     9
    10  # RandomizedSearchCV - Случайный выбор
    11  random_search = RandomizedSearchCV(KNeighborsClassifier(), param_grid, cv=5, n_iter=10, scoring='accuracy',
          random_state=42)
    12  random_search.fit(X_train, y_train)
    13  print(f"Лучший параметр K (RandomizedSearchCV): {random_search.best_params_}")
    14  print(f"Лучшая точность (RandomizedSearchCV): {random_search.best_score_:.2f}")
```
Executed at 2025.03.20 12:51:59 in 496ms

```
Лучший параметр K (GridSearchCV): {'n_neighbors': np.int64(16)}
Лучшая точность (GridSearchCV): 0.96
Лучший параметр K (RandomizedSearchCV): {'n_neighbors': np.int64(17)}
Лучшая точность (RandomizedSearchCV): 0.96
```

Оценка качества оптимальной модели

Оценка качества оптимальной модели

```
[23]
# Используем лучший параметр K из GridSearchCV
best_knn = grid_search.best_estimator_
y_pred_best = best_knn.predict(X_test)

# Оценка качества оптимальной модели
accuracy_best = accuracy_score(y_test, y_pred_best)
print(f"Accuracy (оптимальная модель): {accuracy_best:.2f}")
print(classification_report(y_test, y_pred_best))
```

Executed at 2025.03.20 12:44:18 in 15ms

```
Accuracy (оптимальная модель): 0.96
              precision    recall  f1-score   support

           0       0.95      1.00      0.97        19
           1       1.00      0.90      0.95        21
           2       0.93      1.00      0.97        14

    accuracy                           0.96        54
   macro avg       0.96      0.97      0.96        54
weighted avg       0.97      0.96      0.96        54
```

Сравнение метрик качества исходной и оптимальной моделей

106

```
[24]
print(f"Accuracy исходной модели (K=5): {accuracy:.2f}")
print(f"Accuracy оптимальной модели (K={grid_search.best_params_['n_neighbors']}): {accuracy_best:.2f}")
```

Executed at 2025.03.20 12:44:28 in 8ms

```
Accuracy исходной модели (K=5): 0.94
Accuracy оптимальной модели (K=16): 0.96
```

```
[25]
# Стратегия 1: KFold (по умолчанию в GridSearchCV) - разбивает весь набор данных на K равных частей (фолдов).
cv_scores = cross_val_score(best_knn, X_scaled, y, cv=5, scoring='accuracy')
print(f"Точность кросс-валидации (KFold): {np.mean(cv_scores):.2f}")

# Стратегия 2: StratifiedKFold -  для несбалансированных классов, где мы учитываем пропорции
from sklearn.model_selection import StratifiedKFold
stratified_cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
cv_scores_stratified = cross_val_score(best_knn, X_scaled, y, cv=stratified_cv, scoring='accuracy')
print(f"Точность кросс-валидации (StratifiedKFold): {np.mean(cv_scores_stratified):.2f}")
```

Executed at 2025.03.20 12:44:31 in 59ms

```
Точность кросс-валидации (KFold): 0.97
Точность кросс-валидации (StratifiedKFold): 0.98
```