Московский государственный технический университет им. Н. Э. Баумана

Курс «Технологии машинного обучения»

Отчёт по лабораторной работе №3

Выполнил:
Фролов М. К.
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.

```
from sklearn.datasets import load_iris
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
```

Загрузка датасета Iris

```
iris = load_iris()
X = iris.data
y = iris.target
```

Преобразуем в DataFrame

```
df = pd.DataFrame(X, columns=iris.feature_names)
df['target'] = y
```

```
df.head()
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0

```
[13]
df.isnull().sum()
```

```
sepal length (cm)    0
sepal width (cm)     0
petal length (cm)    0
petal width (cm)     0
target               0
dtype: int64
```

Масштабирование данных

```
[41]
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

```
[42]
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.5, random_state=42)
```

```
[44]
# Обучаем модель с K=2
knn = KNeighborsClassifier(n_neighbors=2)
knn.fit(X_train, y_train)

# Предсказание на тестовой выборке
y_pred = knn.predict(X_test)

# Оценка качества модели
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy (K=2): {accuracy:.2f}")
print(classification_report(y_test, y_pred))
```

```
Accuracy (K=2): 0.91
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        29
           1       0.77      1.00      0.87        23
           2       1.00      0.70      0.82        23

    accuracy                           0.91        75
   macro avg       0.92      0.90      0.90        75
weighted avg       0.93      0.91      0.90        75
```

Подбор гиперпараметра K с использованием GridSearchCV и RandomizedSearchCV

```
param_grid = {'n_neighbors': np.arange(2, 20)}          # Определение параметров для поиска

grid_search = GridSearchCV(KNeighborsClassifier(), param_grid, cv=3, scoring='accuracy')
grid_search.fit(X_train, y_train)
print(f"Лучший параметр K (GridSearchCV): {grid_search.best_params_}")
print(f"Лучшая точность (GridSearchCV): {grid_search.best_score_:.2f}")

random_search = RandomizedSearchCV(KNeighborsClassifier(), param_grid, cv=5, n_iter=10, scoring='accuracy', random_state=42)
random_search.fit(X_train, y_train)
print(f"Лучший параметр K (RandomizedSearchCV): {random_search.best_params_}")
print(f"Лучшая точность (RandomizedSearchCV): {random_search.best_score_:.2f}")
```

```
Лучший параметр K (GridSearchCV): {'n_neighbors': 3}
Лучшая точность (GridSearchCV): 0.93
Лучший параметр K (RandomizedSearchCV): {'n_neighbors': 3}
Лучшая точность (RandomizedSearchCV): 0.92
```

Оценка качества оптимальной модели

```
[51]
# 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))
```

```
Accuracy (оптимальная модель): 0.97
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        29
           1       0.92      1.00      0.96        23
           2       1.00      0.91      0.95        23

    accuracy                           0.97        75
   macro avg       0.97      0.97      0.97        75
weighted avg       0.98      0.97      0.97        75
```

Сравнение метрик качества исходной и оптимальной моделей

```
[52]
print(f"Accuracy исходной модели (K=5): {accuracy:.2f}")
print(f"Accuracy оптимальной модели (K={grid_search.best_params_['n_neighbors']}): {accuracy_best:.2f}")
```

```
Accuracy исходной модели (K=5): 0.91
Accuracy оптимальной модели (K=3): 0.97
```

```
[53]
# Стратегия 1: KFold (по умолчанию в GridSearchCV)
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
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}")
```

```
Точность кросс-валидации (KFold): 0.95
Точность кросс-валидации (StratifiedKFold): 0.94
```