Московский государственный технический университет им. Н. Э. Баумана

Курс «Технологии машинного обучения»

Отчёт по лабораторной работе №3

Выполнил:
Здобняков Ф. А.
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:

```python
from sklearn.datasets import load_iris
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

iris = load_iris()
X = pd.DataFrame(iris.data, columns=iris.feature_names)
y = pd.Series(iris.target, name='Species')

print(X.isnull().sum())

# Разделение данных
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Масштабирование признаков
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

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

```python
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, classification_report

# Обучение модели
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train, y_train)

# Предсказание и оценка
y_pred = knn.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
```

[7]

```
Accuracy: 1.0
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        19
           1       1.00      1.00      1.00        13
           2       1.00      1.00      1.00        13

    accuracy                           1.00        45
   macro avg       1.00      1.00      1.00        45
weighted avg       1.00      1.00      1.00        45
```

```python
from sklearn.model_selection import GridSearchCV, RandomizedSearchCV

# Определение параметров для поиска
param_grid = {'n_neighbors': range(1, 31)}

# GridSearchCV
grid_search = GridSearchCV(KNeighborsClassifier(), param_grid, cv=5)
grid_search.fit(X_train, y_train)
print("Best parameters (GridSearchCV):", grid_search.best_params_)

# RandomizedSearchCV
random_search = RandomizedSearchCV(KNeighborsClassifier(), param_grid, n_iter=10, cv=5)
random_search.fit(X_train, y_train)
print("Best parameters (RandomizedSearchCV):", random_search.best_params_)

# Оценка качества оптимальной модели
best_knn = grid_search.best_estimator_
y_pred_best = best_knn.predict(X_test)
print("Optimized Accuracy:", accuracy_score(y_test, y_pred_best))
print("Optimized Classification Report:\n", classification_report(y_test, y_pred_best))
```

Python

```
Best parameters (GridSearchCV): {'n_neighbors': 3}
Best parameters (RandomizedSearchCV): {'n_neighbors': 8}
Optimized Accuracy: 1.0
Optimized Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        19
           1       1.00      1.00      1.00        13
           2       1.00      1.00      1.00        13

    accuracy                           1.00        45
   macro avg       1.00      1.00      1.00        45
weighted avg       1.00      1.00      1.00        45
```

ОБЛЕМЫ 8 ВЫХОДНЫЕ ДАННЫЕ ТЕРМИНАЛ JUPYTER

ТЕРМИНАЛ

```
Fedor@Air-Anna 3 %
```

Строка 16, столбец 26 Spaces: 4 Ячейка 3 из 6 Go Live

```python
from sklearn.model_selection import KFold, StratifiedKFold

# Стратегия 1: K-Fold Cross-Validation
kfold = KFold(n_splits=5, shuffle=True, random_state=42)
grid_search_kfold = GridSearchCV(KNeighborsClassifier(), param_grid, cv=kfold)
grid_search_kfold.fit(X_train, y_train)
print("Best parameters (GridSearchCV with KFold):", grid_search_kfold.best_params_)

# Стратегия 2: Stratified K-Fold Cross-Validation
stratified_kfold = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
grid_search_stratified = GridSearchCV(KNeighborsClassifier(), param_grid, cv=stratified_kfold)
grid_search_stratified.fit(X_train, y_train)
print("Best parameters (GridSearchCV with StratifiedKFold):", grid_search_stratified.best_params_)

# Оценка качества оптимальной модели с использованием KFold
best_knn_kfold = grid_search_kfold.best_estimator_
y_pred_kfold = best_knn_kfold.predict(X_test)
print("Optimized Accuracy (KFold):", accuracy_score(y_test, y_pred_kfold))
print("Optimized Classification Report (KFold):\n", classification_report(y_test, y_pred_kfold))

# Оценка качества оптимальной модели с использованием StratifiedKFold
best_knn_stratified = grid_search_stratified.best_estimator_
y_pred_stratified = best_knn_stratified.predict(X_test)
print("Optimized Accuracy (StratifiedKFold):", accuracy_score(y_test, y_pred_stratified))
print("Optimized Classification Report (StratifiedKFold):\n", classification_report(y_test, y_pred_stratified))
```

```
Best parameters (GridSearchCV with KFold): {'n_neighbors': 4}
Best parameters (GridSearchCV with StratifiedKFold): {'n_neighbors': 6}
Optimized Accuracy (KFold): 0.9777777777777777
Optimized Classification Report (KFold):
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        19
           1       0.93      1.00      0.96        13
           2       1.00      0.92      0.96        13

    accuracy                           0.98        45
   macro avg       0.98      0.97      0.97        45
weighted avg       0.98      0.98      0.98        45

Optimized Accuracy (StratifiedKFold): 1.0
Optimized Classification Report (StratifiedKFold):
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        19
           1       1.00      1.00      1.00        13
           2       1.00      1.00      1.00        13

    accuracy                           1.00        45
   macro avg       1.00      1.00      1.00        45
weighted avg       1.00      1.00      1.00        45
```

```python
# Сравнение accuracy
original_accuracy = accuracy_score(y_test, y_pred)
optimized_accuracy_kfold = accuracy_score(y_test, y_pred_kfold)
optimized_accuracy_stratified = accuracy_score(y_test, y_pred_stratified)

print(f"Original Model Accuracy: {original_accuracy}")
print(f"Optimized Model Accuracy (KFold): {optimized_accuracy_kfold}")
print(f"Optimized Model Accuracy (StratifiedKFold): {optimized_accuracy_stratified}")

# Сравнение classification report
print("Original Classification Report:\n", classification_report(y_test, y_pred))
print("Optimized Classification Report (KFold):\n", classification_report(y_test, y_pred_kfold))
print("Optimized Classification Report (StratifiedKFold):\n", classification_report(y_test, y_pred_stratified))
```

```
Original Model Accuracy: 1.0
Optimized Model Accuracy (KFold): 0.9777777777777777
Optimized Model Accuracy (StratifiedKFold): 1.0
Original Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        19
           1       1.00      1.00      1.00        13
           2       1.00      1.00      1.00        13

    accuracy                           1.00        45
   macro avg       1.00      1.00      1.00        45
weighted avg       1.00      1.00      1.00        45
```

```
Optimized Classification Report (KFold):
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        19
           1       0.93      1.00      0.96        13
           2       1.00      0.92      0.96        13

    accuracy                           0.98        45
   macro avg       0.98      0.97      0.97        45
weighted avg       0.98      0.98      0.98        45
...
    accuracy                           1.00        45
   macro avg       1.00      1.00      1.00        45
weighted avg       1.00      1.00      1.00        45
```

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...