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 «Системы обработки информации и управления»

Отчет по лабораторной работе №2

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

по дисциплине «Технологии машинного обучения»

Выполнил:
студент группы ИУ5Ц-84Б
Тихонова Д.Д.
подпись, дата

Проверил:
к.т.н., доц., Ю.Е. Гапанюк
подпись, дата

2025 г.

1. Текст программы

```
+ Code + Markdown

import pandas as pd
from sklearn.model_selection import train_test_split, GridSearchCV, RandomizedSearchCV, KFold, StratifiedKFold
from sklearn.neighbors import KNeighborsClassifier, KNeighborsRegressor
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, mean_squared_error, r2_score, classification_report
import numpy as np
[2]  ✓ 8.4s


# 1. Выбор и загрузка датасета (Breast Cancer Wisconsin (Diagnostic))
# Этот датасет подходит для задачи классификации.
try:
    from sklearn.datasets import load_breast_cancer
    cancer = load_breast_cancer()
    df = pd.DataFrame(cancer.data, columns=cancer.feature_names)
    df['target'] = cancer.target  # Целевая переменная
    print("Загружен встроенный датасет breast_cancer.")

except ImportError:
    print("Датасет breast_cancer не найден. Замените код загрузки.")
[3]  ✓ 0.4s
...  Загружен встроенный датасет breast_cancer.


# 2. Предварительная обработка данных (заполнение пропусков, кодирование)
#  В breast cancer dataset нет пропусков или категориальных признаков, так что этот шаг можно пропустить.
#  В реальных данных вам может потребоваться:
#  * Заполнение пропусков: df.fillna(df.mean(), inplace=True)
#  * Кодирование категориальных признаков: pd.get_dummies(df, columns=['категориальный_столбец'])
```

```python
try:
    from sklearn.datasets import load_breast_cancer
    cancer = load_breast_cancer()
    df = pd.DataFrame(cancer.data, columns=cancer.feature_names)
    df['target'] = cancer.target  # Целевая переменная
    print("Загружен встроенный датасет breast_cancer.")

except ImportError:
    print("Датасет breast_cancer не найден. Замените код загрузки.")
```

[3] ✓ 0.4s

```
Загружен встроенный датасет breast_cancer.
```

```python
# 2. Предварительная обработка данных (заполнение пропусков, кодирование)
#  В breast cancer dataset нет пропусков или категориальных признаков, так что этот шаг можно пропустить.
#  В реальных данных вам может потребоваться:
#  * Заполнение пропусков: df.fillna(df.mean(), inplace=True)
#  * Кодирование категориальных признаков: pd.get_dummies(df, columns=['категориальный_столбец'])

# 3. Разделение выборки на обучающую и тестовую
X = df.drop('target', axis=1)  # Признаки
y = df['target']  # Целевая переменная

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42, stratify=y) # stratify=y для сохранения про
```

[4] ✓ 0.0s

```python
# 4. Масштабирование признаков
# Важно для KNeighbors, так как он чувствителен к масштабу.
```

```python
# 4. Масштабирование признаков
# Важно для KNeighbors, так как он чувствителен к масштабу.
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

[5] ✓ 0.0s

```python
# 5. Обучение модели KNN с произвольным K и оценка качества
k_arbitrary = 5  # Пример значения K
knn_arbitrary = KNeighborsClassifier(n_neighbors=k_arbitrary)
knn_arbitrary.fit(X_train, y_train)
y_pred_arbitrary = knn_arbitrary.predict(X_test)

print(f"KNN с K={k_arbitrary}:")
print(classification_report(y_test, y_pred_arbitrary))
#print("Accuracy:", accuracy_score(y_test, y_pred_arbitrary)) #Из classification report всё видно,  accuracy тоже есть.
```

[7] ✓ 0.0s

```
KNN с K=5:
              precision    recall  f1-score   support

           0       1.00      0.89      0.94        64
           1       0.94      1.00      0.97       107

    accuracy                           0.96       171
   macro avg       0.97      0.95      0.96       171
weighted avg       0.96      0.96      0.96       171
```

```python
# a) GridSearchCV
param_grid = {'n_neighbors': list(range(1, 31))}  # Диапазон значений K для перебора
grid_search = GridSearchCV(KNeighborsClassifier(), param_grid, cv=KFold(n_splits=5, shuffle=True, random_state=42), scoring='accuracy
grid_search.fit(X_train, y_train)

print("GridSearchCV:")
print("Лучшие параметры:", grid_search.best_params_)
print("Лучшая оценка:", grid_search.best_score_)

best_knn_grid = grid_search.best_estimator_
y_pred_grid = best_knn_grid.predict(X_test)
print(classification_report(y_test, y_pred_grid))
```

[8] ✓ 1.2s

```
GridSearchCV:
Лучшие параметры: {'n_neighbors': 5}
Лучшая оценка: 0.9672784810126581
              precision    recall  f1-score   support

           0       1.00      0.89      0.94        64
           1       0.94      1.00      0.97       107

    accuracy                           0.96       171
   macro avg       0.97      0.95      0.96       171
weighted avg       0.96      0.96      0.96       171
```

```python
# b) RandomizedSearchCV
param_distributions = {'n_neighbors': list(range(1, 31)),
                       'weights': ['uniform', 'distance'],
                       'metric': ['euclidean', 'manhattan']} #Дополнительные параметры для RandomizedSearchCV
```

```python
# b) RandomizedSearchCV
param_distributions = {'n_neighbors': list(range(1, 31)),
                       'weights': ['uniform', 'distance'],
                       'metric': ['euclidean', 'manhattan']} #Дополнительные параметры для RandomizedSearchCV
random_search = RandomizedSearchCV(KNeighborsClassifier(), param_distributions, cv=StratifiedKFold(n_splits=5, shuffle=True, random_s
random_search.fit(X_train, y_train)

print("\nRandomizedSearchCV:")
print("Лучшие параметры:", random_search.best_params_)
print("Лучшая оценка:", random_search.best_score_)

best_knn_random = random_search.best_estimator_
y_pred_random = best_knn_random.predict(X_test)
print(classification_report(y_test, y_pred_random))
```

[9] ✓ 0.6s

```
RandomizedSearchCV:
Лучшие параметры: {'weights': 'distance', 'n_neighbors': 7, 'metric': 'manhattan'}
Лучшая оценка: 0.9723734177215191
              precision    recall  f1-score   support

           0       1.00      0.89      0.94        64
           1       0.94      1.00      0.97       107

    accuracy                           0.96       171
   macro avg       0.97      0.95      0.96       171
weighted avg       0.96      0.96      0.96       171
```

```python
# Этот шаг подразумевает анализ classification reports, чтобы сравнить,
#  какая модель показала лучшие результаты по accuracy, precision, recall, f1-score.
#  Обычно, RandomizedSearchCV дает более хорошие результаты, чем KNeighborsClassifier с K=5.
#  GridSearchCV может показать результаты лучше, чем RandomizedSearchCV, но работает дольше.
#  Вывод о лучших параметрах нужно делать исходя из classification report.

print("\nСравнение метрик:")
print("KNN с K=5 (произвольный):")
print(classification_report(y_test, y_pred_arbitrary))
print("\nKNN с GridSearchCV:")
print(classification_report(y_test, y_pred_grid))
print("\nKNN с RandomizedSearchCV:")
print(classification_report(y_test, y_pred_random))
```

[10] ✓ 0.0s

```
Сравнение метрик:
KNN с K=5 (произвольный):
              precision    recall  f1-score   support

           0       1.00      0.89      0.94        64
           1       0.94      1.00      0.97       107

    accuracy                           0.96       171
   macro avg       0.97      0.95      0.96       171
weighted avg       0.96      0.96      0.96       171


KNN с GridSearchCV:
              precision    recall  f1-score   support

           0       1.00      0.89      0.94        64
           1       0.94      1.00      0.97       107
```