Министерство образования Республики Беларусь
Учреждение образования
«Брестский Государственный технический университет»
Кафедра ИИТ

Лабораторная работа №3
По дисциплине: «Основы машинного обучения»
Тема: «Сравнение классических методов классификации»

Выполнила:
Студентка 3 курса
Группы АС-66
Прокурат В. Д.
Проверил:
Крощенко А. А.

Брест 2025

Цель работы: на практике сравнить работу нескольких алгоритмов классификации, таких как метод k-ближайших соседей (k-NN), деревья решений и метод опорных векторов (SVM). Научиться подбирать гиперпараметры моделей и оценивать их влияние на результат.

Вариант 9

• `Glass Identification`

• Классифицировать тип стекла на основе его химического состава

• **Задания:**

1. Загрузите данные и стандартизируйте их;
2. Разделите выборку;

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.metrics import classification_report, accuracy_score

# 1, 2.
df = pd.read_csv("glass.csv")

X = df.drop("Type", axis=1)
y = df["Type"]

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

X_train, X_test, y_train, y_test = train_test_split(
    X_scaled, y, test_size=0.2, random_state=42, stratify=y
)
```

3. Обучите модели k-NN, Decision Tree и SVM;
4. Сравните производительность моделей с помощью `classification_report` (sklearn.metrics);

```
k_values = range(1, 21)
accuracies = []

for k in k_values:
    knn = KNeighborsClassifier(n_neighbors=k)
    knn.fit(X_train, y_train)
    y_pred = knn.predict(X_test)
    accuracies.append(accuracy_score(y_test, y_pred))
best_k = k_values[accuracies.index(max(accuracies))]
print(f"Лучшее значение k из промежутка [1; 21]: {best_k}, точность:
{max(accuracies):.4f}\n")

models = {
    "k-NN": KNeighborsClassifier(n_neighbors=best_k),
    "Decision Tree": DecisionTreeClassifier(random_state=42),
    "SVM": SVC(kernel="rbf", random_state=42)
}
```

```
for name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    print(f"=== {name} ===")
    print(f"Accuracy: {accuracy_score(y_test, y_pred):.4f}")
    print(classification_report(y_test, y_pred, zero_division=0))
    print()
```

```
Лучшее значение k из промежутка [1; 21]: 2, точность: 0.7907
```

Мы определили лучшее k для модели k-NN.
При сравнении производительности моделей, модель k-NN будет использоваться с лучшем количеством соседей (k=2).

```
=== k-NN ===
Accuracy: 0.7907
              precision    recall  f1-score   support

           1       0.76      0.93      0.84        14
           2       0.75      0.80      0.77        15
           3       1.00      0.33      0.50         3
           5       0.67      0.67      0.67         3
           6       1.00      0.50      0.67         2
           7       1.00      0.83      0.91         6

    accuracy                           0.79        43
   macro avg       0.86      0.68      0.73        43
weighted avg       0.81      0.79      0.78        43
```

```
=== Decision Tree ===
Accuracy: 0.6744
              precision    recall  f1-score   support

           1       0.77      0.71      0.74        14
           2       0.70      0.47      0.56        15
           3       0.40      0.67      0.50         3
           5       1.00      0.67      0.80         3
           6       0.40      1.00      0.57         2
           7       0.75      1.00      0.86         6

    accuracy                           0.67        43
   macro avg       0.67      0.75      0.67        43
weighted avg       0.72      0.67      0.67        43
```

```
=== SVM ===
Accuracy: 0.7209
              precision    recall  f1-score   support

           1       0.69      0.79      0.73        14
           2       0.69      0.73      0.71        15
           3       0.00      0.00      0.00         3
           5       1.00      0.67      0.80         3
           6       0.50      0.50      0.50         2
           7       0.86      1.00      0.92         6

    accuracy                           0.72        43
   macro avg       0.62      0.61      0.61        43
weighted avg       0.68      0.72      0.69        43
```

Лучше всего работает метод k ближайших соседей при k=2 - он показывает наивысшую точность и сбалансированные значения метрик precision и recall. Методы Decision Tree и SVM немного хуже, вероятно из-за малого количества наблюдений для некоторых классов и пересечения признаков между типами стекла.

5. Укажите, какой класс модели определяют хуже всего, и предположите, почему.

```
for name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    report = classification_report(y_test, y_pred, output_dict=True, zero_division=0)
    recalls = {cls: metrics["recall"] for cls, metrics in report.items() if
cls.isdigit()}
    worst_class = min(recalls, key=recalls.get)
    print(f"{name}: худший класс - {worst_class}
(recall={recalls[worst_class]:.2f})")
```

```
k-NN: худший класс — 3 (recall=0.33)
Decision Tree: худший класс — 2 (recall=0.47)
SVM: худший класс — 3 (recall=0.00)
```

Хуже всего классифицируется класс 3, поскольку у него минимальные значения recall (0.33 у k-NN, 0.0 у SVM).
Также класс 6 имеет малое количество объектов (всего 2), из-за чего его определение нестабильно: небольшое изменение выборки может резко ухудшить качество.
Основная причина - небаланс данных и пересечение признаков между редкими классами.

Вывод: на практике сравнила работу нескольких алгоритмов классификации, таких как метод k-ближайших соседей (k-NN), деревья решений и метод опорных векторов (SVM). Научилась подбирать гиперпараметры моделей и оценивать их влияние на результат.