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»

Выполнил:
Студент 2 курса
Группы АС-66
Колбашко А. В.
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.

Ход работы

Задачи:

1. Загрузить датасет по варианту;
2. Разделить данные на обучающую и тестовую выборки;
3. Обучить на обучающей выборке три модели: k-NN, Decision Tree и SVM;
4. Для модели k-NN исследовать, как меняется качество при разном количестве соседей (k);
5. Оценить точность каждой модели на тестовой выборке;
6. Сравнить результаты, сделать выводы о применимости каждого метода для данного набора данных.

Вариант 9

- `Glass Identification`
- Классифицировать тип стекла на основе его химического состава
- **Задания:**
    1. Загрузите данные и стандартизируйте их;
    2. Разделите выборку;
    3. Обучите модели k-NN, Decision Tree и SVM;
    4. Сравните производительность моделей с помощью `classification_report` (sklearn.metrics);
    5. Укажите, какой класс модели определяют хуже всего, и предположите, почему.

Код программы:

```
import pandas as pd
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

data = pd.read_csv("C:\\Users\\wlksm\\OneDrive\\Рабочий
стол\\Уник\\PP\\Parser\\3\\src\\glass.csv")

X = data.drop('Type', axis=1)
y = data['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=0.25, random_state=16, stratify=y
)

knn_best = KNeighborsClassifier(n_neighbors=2)
```

```
knn_best.fit(X_train, y_train)
y_pred_knn = knn_best.predict(X_test)

dt = DecisionTreeClassifier(random_state=16)
dt.fit(X_train, y_train)
y_pred_dt = dt.predict(X_test)

svm = SVC(kernel='rbf', C=1, gamma='scale', random_state=16)
svm.fit(X_train, y_train)
y_pred_svm = svm.predict(X_test)


print("Отчёт по моделям\n")

print("k-NN:")
print(classification_report(y_test, y_pred_knn))
print("-" * 55)

print("Decision Tree:")
print(classification_report(y_test, y_pred_dt))
print("-" * 55)

print("SVM:")
print(classification_report(y_test, y_pred_svm))
print("-" * 55)

acc_knn = accuracy_score(y_test, y_pred_knn)
acc_dt = accuracy_score(y_test, y_pred_dt)
acc_svm = accuracy_score(y_test, y_pred_svm)

print(f"Accuracy k-NN: {acc_knn:.3f}")
print(f"Accuracy Decision Tree: {acc_dt:.3f}")
print(f"Accuracy SVM: {acc_svm:.3f}")
```

Результат выполнения программы:

k-NN:

	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:

	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:

	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

Accuracy k-NN: 0.791

Accuracy Decision Tree: 0.674

Accuracy SVM: 0.721

Вывод: наиболее успешной моделью для данной выборки является k-NN. Хуже всего модели определяют класс 3, скорее всего это связано с похожей структурой данного класса на другие классы и малым количеством элементов этого класса в датасете.