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

Отчёт по лабораторной работе №3

Выполнил:
Студент 3 курса
Группы АС-65
Романюк Д. А.
Проверил:
Крощенко А. А.

Брест 2025

Цель: 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 5

• Mushroom Classification
• Определить, является ли гриб ядовитым или съедобным
• Задания:
1. Загрузите данные и преобразуйте все категориальные признаки в числовые (например, с помощью One-Hot Encoding);
2. Разделите данные на обучающую и тестовую части;
3. Обучите классификаторы k-NN, Decision Tree и SVM;
4. Рассчитайте точность и полноту (precision и recall) для класса "ядовитый";
5. Сделайте вывод о том, какой классификатор лучше всего справляется с этой задачей, где цена ошибки очень высока.

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OneHotEncoder
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.metrics import precision_score, recall_score

df = pd.read_csv(r"C:\OMO\3lab\mushrooms.csv")

print(df.head())

X = df.drop('class', axis=1)
y = df['class']

encoder = OneHotEncoder(sparse_output=False)
X_encoded = encoder.fit_transform(X)

X_train, X_test, y_train, y_test = train_test_split(
    X_encoded, y, test_size=0.3, random_state=42
)

models = {
    "k-NN": KNeighborsClassifier(n_neighbors=5),
    "Decision Tree": DecisionTreeClassifier(random_state=42),
    "SVM": SVC(kernel='rbf', random_state=42)
}

results = {}

for name, model in models.items():
    model.fit(X_train, y_train)
```

```
        y_pred = model.predict(X_test)

        precision = precision_score(y_test, y_pred, pos_label='p')
        recall = recall_score(y_test, y_pred, pos_label='p')

        results[name] = {'Precision': precision, 'Recall': recall}

    print("\n=== Результаты классификации ===")
    for model_name, metrics in results.items():
        print(f"{model_name}: Precision = {metrics['Precision']:.4f}, Recall
= {metrics['Recall']:.4f}")

    print("\n=== Вывод ===")
    best_model = max(results.items(), key=lambda x: x[1]['Recall'])
    print(f"Лучший классификатор по полноте (важно при высокой цене ошибки):
{best_model[0]}")
```

```
=== Результаты классификации ===
k-NN: Precision = 1.0000, Recall = 1.0000
Decision Tree: Precision = 1.0000, Recall = 1.0000
SVM: Precision = 1.0000, Recall = 1.0000

=== Вывод ===
[5 rows x 23 columns]

=== Результаты классификации ===
k-NN: Precision = 1.0000, Recall = 1.0000
Decision Tree: Precision = 1.0000, Recall = 1.0000
SVM: Precision = 1.0000, Recall = 1.0000

=== Вывод ===
Лучший классификатор по полноте (важно при высокой цене ошибки): k-NN
PS C:\OMO>
```

Вывод: Все модели показывают почти идеальные результаты, но если выбирать с точки зрения *безопасности* (максимизация Recall для класса “ядовитый”) — лучше взять **k-NN**.