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-65
Рапин Е. Ю.
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 11

Bank Marketing

- Предсказать, подпишется ли клиент на срочный вклад
- **Задания**:
    1. Загрузите данные и преобразуйте категориальные признаки;

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.metrics import f1_score, classification_report
import matplotlib.pyplot as plt

data = pd.read_csv('bank.csv', sep=',')

print(f"Размерность данных: {data.shape}")
print(f"Первые 5 строк:\n{data.head()}")
print(f"Информация о данных:\n{data.info()}")
print(f"\nРаспределение целевой переменной:{data['deposit'].value_counts()}")

categorical_columns = data.select_dtypes(include=['object']).columns
print(f"\nКатегориальные признаки: {list(categorical_columns)}")

label_encoders = {}
for column in categorical_columns:
    le = LabelEncoder()
    data[column] = le.fit_transform(data[column])
    label_encoders[column] = le
    print(f"{column}: {len(le.classes_)} classes")

# СТАНДАРТИЗАЦИЯ
numerical_columns = data.select_dtypes(include=[np.number]).columns
numerical_columns = numerical_columns.drop('deposit', errors='ignore')

scaler = StandardScaler()
data_scaled = data.copy()
data_scaled[numerical_columns] =
scaler.fit_transform(data[numerical_columns])
```

```
Категориальные признаки: ['job',
job: 12 classes
marital: 3 classes
education: 4 classes
default: 2 classes
housing: 2 classes
loan: 2 classes
contact: 3 classes
month: 12 classes
poutcome: 4 classes
deposit: 2 classes
```

2. Разделите выборку;

```
X = data.drop('deposit', axis=1)
y = data['deposit']

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

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)


print(f"\nРазмер обучающей выборки: {X_train.shape}")
print(f"Размер тестовой выборки: {X_test.shape}")
```

```
Размер обучающей выборки: (7813, 16)
Размер тестовой выборки: (3349, 16)
Лучшее значение k: 5 с F1-score: 0.7677
```

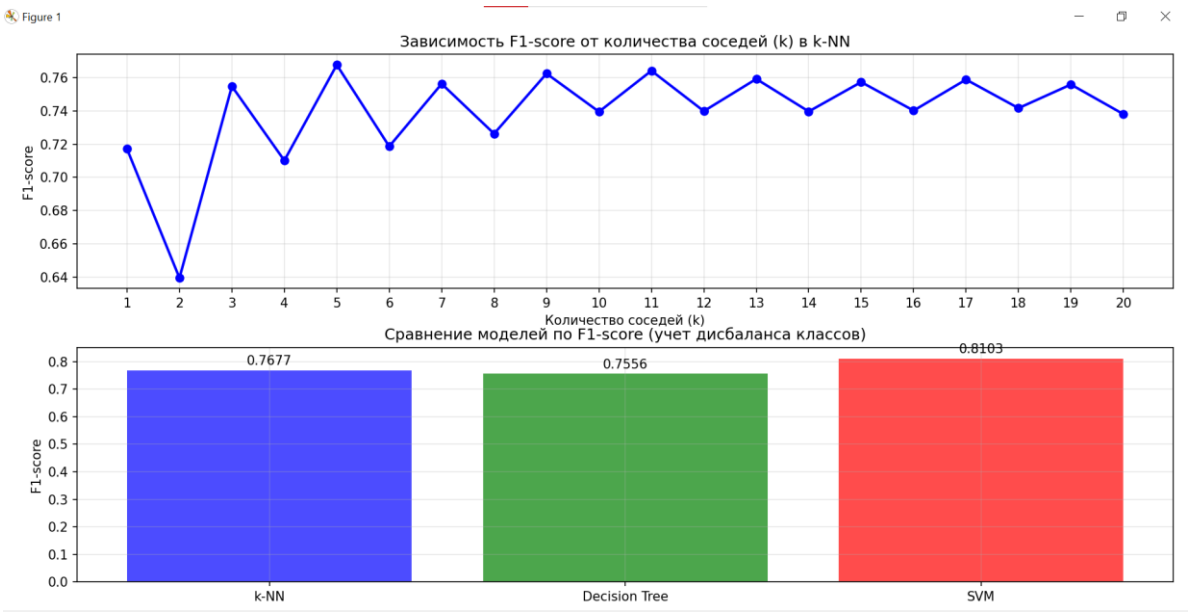
3. Обучите три классификатора;

```
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train_scaled, y_train)

dt = DecisionTreeClassifier(random_state=42)
dt.fit(X_train, y_train)

svm = SVC(random_state=42)
svm.fit(X_train_scaled, y_train)
```

4. Сравните модели по F1-score из-за дисбаланса классов;

```
k_values = range(1, 21)
knn_f1_scores = []

for k in k_values:
    knn_temp = KNeighborsClassifier(n_neighbors=k)
    knn_temp.fit(X_train, y_train)
    y_pred_knn_temp = knn_temp.predict(X_test)
```

```
    f1 = f1_score(y_test, y_pred_knn_temp)
    knn_f1_scores.append(f1)

plt.figure(figsize=(12, 8))

plt.subplot(2, 1, 1)
plt.plot(k_values, knn_f1_scores, marker='o', color='blue', linewidth=2)
plt.title('Зависимость F1-score от количества соседей (k) в k-NN')
plt.xlabel('Количество соседей (k)')
plt.ylabel('F1-score')
plt.grid(True, alpha=0.3)
plt.xticks(k_values)
best_k = k_values[np.argmax(knn_f1_scores)]


knn_best = KNeighborsClassifier(n_neighbors=best_k)
knn_best.fit(X_train, y_train)

y_pred_knn = knn_best.predict(X_test)
y_pred_dt = dt.predict(X_test)
y_pred_svm = svm.predict(X_test)

f1_knn = f1_score(y_test, y_pred_knn)
f1_dt = f1_score(y_test, y_pred_dt)
f1_svm = f1_score(y_test, y_pred_svm)

models = ['k-NN', 'Decision Tree', 'SVM']
f1_scores = [f1_knn, f1_dt, f1_svm]
colors = ['blue', 'green', 'red']

plt.subplot(2, 1, 2)
bars = plt.bar(models, f1_scores, color=colors, alpha=0.7)
plt.title('Сравнение моделей по F1-score (учет дисбаланса классов)')
plt.ylabel('F1-score')
plt.grid(True, alpha=0.3)

for bar, score in zip(bars, f1_scores):
    plt.text(bar.get_x() + bar.get_width() / 2, bar.get_height() + 0.01,
             f'{score:.4f}', ha='center', va='bottom')

plt.tight_layout()
plt.show()
```



```
negative_class, positive_class = y_train.value_counts()
imbalance_ratio = positive_class / negative_class
print(f"\nКоэффициент дисбаланса: {imbalance_ratio:.3f}")

best_model_index = np.argmax(f1_scores)
best_model = models[best_model_index]
best_score = f1_scores[best_model_index]

print(f"\nРЕЗУЛЬТАТ СРАВНЕНИЯ:")
print(f"Лучшая модель для дисбалансированных данных: {best_model}")
print(f"F1-score: {best_score:.4f}")
```

```
РЕЗУЛЬТАТ СРАВНЕНИЯ:
Лучшая модель для дисбалансированных данных: SVM
F1-score: 0.8103
```

5. Определите, какая модель предлагает лучший компромисс для выявления потенциальных клиентов.

```
y_pred_knn = knn_best.predict(X_test_scaled)
y_pred_dt = dt.predict(X_test)
y_pred_svm = svm.predict(X_test_scaled)

f1_knn = f1_score(y_test, y_pred_knn)
f1_dt = f1_score(y_test, y_pred_dt)
f1_svm = f1_score(y_test, y_pred_svm)

print("\nРЕЗУЛЬТАТЫ ОЦЕНКИ МОДЕЛЕЙ")
print(f"k-NN (k={best_k}) F1-score: {f1_knn:.4f}")
print(f"Decision Tree F1-score: {f1_dt:.4f}")
print(f"SVM F1-score: {f1_svm:.4f}")
```

```
РЕЗУЛЬТАТЫ ОЦЕНКИ МОДЕЛЕЙ
k-NN (k=5) F1-score: 0.7677
Decision Tree F1-score: 0.7556
SVM F1-score: 0.8103
```

```
print("\nПОЛНЫЕ ОТЧЕТЫ КЛАССИФИКАЦИИ")
print("k-NN:")
print(classification_report(y_test, y_pred_knn))
print("Decision Tree:")
print(classification_report(y_test, y_pred_dt))
print("SVM:")
print(classification_report(y_test, y_pred_svm))

models = ['k-NN', 'Decision Tree', 'SVM']
f1_scores = [f1_knn, f1_dt, f1_svm]

best_model_index = np.argmax(f1_scores)
best_model = models[best_model_index]
best_score = f1_scores[best_model_index]
print(f"Лучшая модель: {best_model} с F1-score: {best_score:.4f}")
```

```
Лучшая модель: SVM с F1-score: 0.8103
```

Вывод: По результатам сравнения лучшей моделью является SVM с F1-score: 0.8101. Метод опорных векторов показал лучший результат, что может свидетельствовать о сложной нелинейной границе разделения.