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

Лабораторная работа №3
По дисциплине: «ОМО»
Тема:” Сравнение классических методов классификации”

Выполнил:
Студент 3-го курса
Группы АС-66
Осовец А.О.
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 13

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.

• Telco Customer Churn
• Предсказать, откажется ли клиент от услуг телеком-оператора
• Задания:

1. Загрузите данные, обработайте категориальные признаки и пропуски;
2. Разделите данные на обучающую и тестовую выборки;
3. Обучите модели k-NN, Decision Tree и SVM;
4. Сравните модели по метрике F1-score для класса "отток";
5. Сделайте вывод о применимости каждого из методов для решения бизнес-задачи по удержанию клиентов.

```
import pandas as pd
import os
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.metrics import f1_score

# 1. Загрузка датасета
current_dir = os.path.dirname(os.path.abspath(__file__))
project_root = os.path.abspath(os.path.join(current_dir, '..', '..', '..', '..', '..'))
file_path = os.path.join(project_root, 'Telco-Customer-Churn.csv')

try:
    df = pd.read_csv(file_path)
    print(f"    Файл успешно загружен: {file_path}")
except FileNotFoundError:
    print(f"    Файл не найден: {file_path}")
    df = None

if "customerID" in df.columns:
    df.drop("customerID", axis=1, inplace=True)

df["TotalCharges"] = pd.to_numeric(df["TotalCharges"], errors="coerce")
```

```
df = df.dropna()

y = df["Churn"].map({"Yes": 1, "No": 0})
X = df.drop("Churn", axis=1)

num_cols = X.select_dtypes(include=["int64", "float64"]).columns
cat_cols = X.select_dtypes(include=["object"]).columns

# 2. Кодирование и стандартизация
preprocessor = ColumnTransformer([
    ("num", StandardScaler(), num_cols),
    ("cat", OneHotEncoder(handle_unknown="ignore"), cat_cols)
])

# 3. Определение моделей
models = {
    "DecisionTree": DecisionTreeClassifier(random_state=42),
    "SVM": SVC(kernel="rbf", random_state=42),
    "kNN": KNeighborsClassifier(n_neighbors=5)
}

# 2. Разделение данных на обучающую и тестовую выборки
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42, stratify=y
)

# 3. Обучение моделей и оценка точности
results = {}
for name, model in models.items():
    pipe = Pipeline([("prep", preprocessor), ("model", model)])
    pipe.fit(X_train, y_train)
    y_pred = pipe.predict(X_test)
    score = f1_score(y_test, y_pred)
    results[name] = score
    print(f"{name}: F1-score = {score:.4f}")

# 4. Исследование влияния k на k-NN
k_values = range(1, 21)
knn_scores = []
for k in k_values:
    knn = Pipeline([
        ("prep", preprocessor),
        ("model", KNeighborsClassifier(n_neighbors=k))
    ])
    knn.fit(X_train, y_train)
    y_pred = knn.predict(X_test)
    knn_scores.append(f1_score(y_test, y_pred))

best_k = k_values[np.argmax(knn_scores)]
best_knn = max(knn_scores)

print("\nF1-score k-NN при разных k:")
for i in [1, 3, 5, 7, 9, 11, 15]:
    print(f"k={i}: {knn_scores[i-1]:.4f}")

# 5. Оценка точности каждой модели
print("\nРезультаты моделей (F1-score):")
for name, score in results.items():
    print(f"{name}: {score:.4f}")
print(f"kNN (best k={best_k}): {best_knn:.4f}")

best_model_name = max(results, key=results.get)
print(f"\n6. Лучшая модель: {best_model_name} (F1-score = {results[best_model_name]:.4f})")

# 6. Визуализация зависимости F1-score от k
plt.figure(figsize=(6, 4))
```

```
plt.plot(k_values, knn_scores, marker='o')
plt.title("Зависимость F1-score от количества соседей (k)")
plt.xlabel("k")
plt.ylabel("F1-score")
plt.grid(True)
plt.show()
```

Результат:

```
DecisionTree: F1-score = 0.4516
SVM: F1-score = 0.5499
kNN: F1-score = 0.5628

F1-score k-NN при разных k:
k=1: 0.5042
k=3: 0.5398
k=5: 0.5628
k=7: 0.5558
k=9: 0.5657
k=11: 0.5673
k=15: 0.5727

Результаты моделей (F1-score):
DecisionTree: 0.4516
SVM: 0.5499
kNN: 0.5628
kNN (best k=19): 0.6002

6. Лучшая модель: kNN (F1-score = 0.5628)
```

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