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

Лабораторная работа №3
По дисциплине: «ОМО»

Выполнил:
Студент 3-го курса
Группы АС-66
Николова М.С
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

Вариант 6

```
import pandas as pd
import numpy as np
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 confusion_matrix, recall_score, precision_score, f1_score, accuracy_score
import matplotlib.pyplot as plt

# 1. Загрузка и предварительная обработка данных
df = pd.read_csv("pima-indians-diabetes.csv", comment="#", header=None)

df.columns = [
    'Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness',
    'Insulin', 'BMI', 'DiabetesPedigreeFunction', 'Age', 'Class'
]

print("Пропущенные значения по столбцам:")
print(df.isnull().sum())

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

# 2. Разделение выборки с стратификацией
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42, stratify=y
)

# 3. Стандартизация признаков
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# 4. Обучение моделей с k=1 для k-NN
knn = KNeighborsClassifier(n_neighbors=1)
dt = DecisionTreeClassifier(max_depth=4, random_state=42)
svm = SVC(kernel='linear', random_state=42)

knn.fit(X_train_scaled, y_train)
dt.fit(X_train, y_train)
svm.fit(X_train_scaled, y_train)

# 5. Предсказания
y_pred_knn = knn.predict(X_test_scaled)
y_pred_dt = dt.predict(X_test)
y_pred_svm = svm.predict(X_test_scaled)

# 6. Метрики
models = {
    'k-NN (k=1)': y_pred_knn,
    'Decision Tree (max_depth=4)': y_pred_dt,
    'SVM (linear kernel)': y_pred_svm
}
```

```
print("\nМЕТРИКИ ДЛЯ НАЛИЧИЯ ДИАБЕТА (положительный класс = 1):")
for name, y_pred in models.items():
    cm = confusion_matrix(y_test, y_pred)
    recall = recall_score(y_test, y_pred)
    precision = precision_score(y_test, y_pred)
    f1 = f1_score(y_test, y_pred)
    accuracy = accuracy_score(y_test, y_pred)

    print(f"\n{name}:")
    print(f"Матрица ошибок:\n{cm}")
    print(f"Accuracy: {accuracy:.4f}")
    print(f"Precision: {precision:.4f}")
    print(f"Recall: {recall:.4f}")
    print(f"F1: {f1:.4f}")

# 7. Построение графика Recall(k) для k-NN
recall_scores_k = []
k_values = range(1, 21)

for k in k_values:
    knn_k = KNeighborsClassifier(n_neighbors=k)
    knn_k.fit(X_train_scaled, y_train)
    y_pred_k = knn_k.predict(X_test_scaled)
    recall_scores_k.append(recall_score(y_test, y_pred_k))

# Визуализация
plt.figure(figsize=(8,5))
plt.plot(k_values, recall_scores_k, marker='o')
plt.axvline(x=1, color='red', linestyle='--', label='Используемый k=1')
plt.xlabel('Количество соседей (k)')
plt.ylabel('Recall')
plt.title('Зависимость Recall от k для k-NN')
plt.grid(True)
plt.legend()
plt.show()

# 8. Сравнение моделей по Recall
print("\nСРАВНЕНИЕ ПО RECALL (важно для медицинской задачи):")
recall_scores_models = {name: recall_score(y_test, y_pred) for name, y_pred in models.items()}
for name, score in sorted(recall_scores_models.items(), key=lambda x: x[1], reverse=True):
    print(f"{name}: Recall = {score:.4f}")

best_model = max(recall_scores_models.items(), key=lambda x: x[1])
print(f"\nЛУЧШАЯ МОДЕЛЬ: {best_model[0]} (Recall = {best_model[1]:.4f})")

# 9. Обоснование выбора
print("""
ОБОСНОВАНИЕ:
В медицинских задачах важно минимизировать количество ложноотрицательных (False Negative) —
то есть случаев, когда болезнь есть, но модель не выявила её.
Поэтому основная метрика — Recall для положительного класса.
Модель с наибольшим Recall считается предпочтительной, даже если её точность (Accuracy) чуть ниже.
""")
```

Результат:

```
Пропущенные значения по столбцам:
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
```

```
BMI                       0
DiabetesPedigreeFunction  0
Age                       0
Class                     0
dtype: int64

МЕТРИКИ ДЛЯ НАЛИЧИЯ ДИАБЕТА (положительный класс = 1):
k-NN (k=1):
Матрица ошибок:
[[121  29]
 [ 33  48]]
Accuracy: 0.7316
Precision: 0.6234
Recall: 0.5926
F1: 0.6076

Decision Tree (max_depth=4):
Матрица ошибок:
[[128  22]
 [ 32  49]]
Accuracy: 0.7662
Precision: 0.6901
Recall: 0.6049
F1: 0.6447

SVM (linear kernel):
Матрица ошибок:
[[128  22]
 [ 41  40]]
Accuracy: 0.7273
Precision: 0.6452
Recall: 0.4938
F1: 0.5594

СРАВНЕНИЕ ПО RECALL (важно для медицинской задачи):
Decision Tree (max_depth=4): Recall = 0.6049
k-NN (k=1): Recall = 0.5926
SVM (linear kernel): Recall = 0.4938

ЛУЧШАЯ МОДЕЛЬ: Decision Tree (max_depth=4) (Recall = 0.6049)
ОБОСНОВАНИЕ:
В медицинских задачах важно минимизировать количество ложноотрицательных (False Negative) —
то есть случаев, когда болезнь есть, но модель не выявила её.
Поэтому основная метрика — Recall для положительного класса.
Модель с наибольшим Recall считается предпочтительной, даже если её точность (Accuracy) чуть ниже.
```