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
Цеван К.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 12

KDD Cup 1999
• Классифицировать сетевые подключения на "нормальные" и "атаки"
• Задания:
1. Загрузите данные, преобразуйте категориальные признаки;
2. Создайте бинарную целевую переменную (normal vs attack);
3. Обучите k-NN, Decision Tree и SVM на небольшой подвыборке данных (например, 10 000 строк);
4. Сравните recall для класса "атака" и время обучения каждой модели;
5. Сделайте вывод о том, какая модель эффективнее для обнаружения вторжений.charges:

```
import time
import warnings
warnings.filterwarnings("ignore")

import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import LinearSVC
from sklearn.metrics import recall_score, accuracy_score

# Путь к файлу (тот же, что у тебя)
data_path = "kddcup.data_10_percent_corrected"

# Названия колонок — без них датасет выглядит как зловещая таблица без души
cols = [
    "duration","protocol_type","service","flag","src_bytes","dst_bytes","land","wrong_fragment","urgent",
    "hot","num_failed_logins","logged_in","num_compromised","root_shell","su_attempted","num_root",
    "num_file_creations","num_shells","num_access_files","num_outbound_cmds","is_host_login","is_guest_login",
    "count","srv_count","serror_rate","srv_serror_rate","rerror_rate","srv_rerror_rate","same_srv_rate",

"diff_srv_rate","srv_diff_host_rate","dst_host_count","dst_host_srv_count","dst_host_same_srv_rate",
```

```
"dst_host_diff_srv_rate","dst_host_same_src_port_rate","dst_host_srv_diff_host_rate","dst_
host_serror_rate",
    "dst_host_srv_serror_rate","dst_host_rerror_rate","dst_host_srv_rerror_rate","label"
]

# Загружаем датасет
df_raw = pd.read_csv(data_path, names=cols)

# Цель: 0 = нормальный трафик, 1 = атака
df_raw["target"] = (df_raw["label"] != "normal.").astype(int)

# Отключаем колонку label (она текстовая) и превращаем категориальные в дамми
categorical = ["protocol_type", "service", "flag"]
df_processed = pd.get_dummies(df_raw.drop(columns=["label"]), columns=categorical)

# Возьмём подмножество — чтобы не ждать вечность (и чтобы сервер не захандрил)
requested_sample = 5000
sample_n = min(requested_sample, len(df_processed))
df_sample = df_processed.sample(n=sample_n, random_state=42)

# Признаки и метки
X = df_sample.drop(columns=["target"])
y = df_sample["target"]

# Делим в том же стиле: стратифицированно, чтобы соотношение классов не улетело в
ад
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, stratify=y, random_state=42
)

# Список k для k-NN — проверим несколько соседей
k_values = [1, 3, 5, 7]
summary = []

# Функция для округления времени и метрик (немного аккуратности в сумрачном мире)
def r(x, nd=4):
    return round(x, nd)

# k-NN: масштабирование обязательно (иначе соседи будут в панике)
for k in k_values:
    pipeline = Pipeline([
        ("scaler", StandardScaler()),
        ("knn", KNeighborsClassifier(n_neighbors=k, n_jobs=1))
    ])

    t_start = time.time()
```

```
    pipeline.fit(X_train, y_train)
    fit_dur = time.time() - t_start

    t_start = time.time()
    preds = pipeline.predict(X_test)
    pred_dur = time.time() - t_start

    rec = recall_score(y_test, preds)
    acc = accuracy_score(y_test, preds)

    summary.append({
        "model": "k-NN",
        "param": f"k={k}",
        "recall_attack": r(rec),
        "accuracy": r(acc),
        "fit_time_s": r(fit_dur),
        "predict_time_s": r(pred_dur)
    })

# Дерево решений — без масштабирования (оно само по себе грубая сила)
dt_clf = DecisionTreeClassifier(random_state=42)
t_start = time.time()
dt_clf.fit(X_train, y_train)
fit_dur = time.time() - t_start

t_start = time.time()
preds = dt_clf.predict(X_test)
pred_dur = time.time() - t_start

rec = recall_score(y_test, preds)
acc = accuracy_score(y_test, preds)

summary.append({
    "model": "Decision Tree",
    "param": "",
    "recall_attack": r(rec),
    "accuracy": r(acc),
    "fit_time_s": r(fit_dur),
    "predict_time_s": r(pred_dur)
})

# Линейный SVM со стандартизацией — терпеливо (или почти терпеливо) смотрим на данные
svm_pipeline = Pipeline([
    ("scaler", StandardScaler()),
    ("svm", LinearSVC(max_iter=5000, random_state=42))
])
```

```
t_start = time.time()
svm_pipeline.fit(X_train, y_train)
fit_dur = time.time() - t_start

t_start = time.time()
preds = svm_pipeline.predict(X_test)
pred_dur = time.time() - t_start

rec = recall_score(y_test, preds)
acc = accuracy_score(y_test, preds)

summary.append({
    "model": "Linear SVM",
    "param": "",
    "recall_attack": r(rec),
    "accuracy": r(acc),
    "fit_time_s": r(fit_dur),
    "predict_time_s": r(pred_dur)
})

# Собираем таблицу результатов — красиво и без надрыва
results_df = pd.DataFrame(summary)

print("Результаты:")
print(results_df.to_string(index=False))

# Небольшой аналитический отчёт (кто в кого лучше стреляет по recall и кто ниндзя по времени)
best_recall = results_df.loc[results_df["recall_attack"].idxmax()]
best_fit = results_df.loc[results_df["fit_time_s"].idxmin()]
best_pred = results_df.loc[results_df["predict_time_s"].idxmin()]

print("\nКороткий анализ:")
print(f"- Лучшая по recall: {best_recall['model']} {best_recall['param']} (recall={best_recall['recall_attack']})")
print(f"- Быстрее всего обучается: {best_fit['model']} {best_fit['param']} (fit_time={best_fit['fit_time_s']}s)")
print(f"- Быстрее всего предсказывает: {best_pred['model']} {best_pred['param']} (predict_time={best_pred['predict_time_s']}s)")
```

Вывод: на практике сравнили работу несколько алгоритмов классификации.