

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
import matplotlib.pyplot as plt
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
```

```
data_path = "kddcup.data_10_percent_corrected"
```

```
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", "error_rate", "srv_error_rate", "error_rate", "srv_error_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_error_rate",
    "dst_host_srv_error_rate","dst_host_error_rate","dst_host_srv_error_rate","label"
]
```

```
df_raw = pd.read_csv(data_path, names=cols)
df_raw["target"] = (df_raw["label"] != "normal.").astype(int)
```

```
categorical = ["protocol_type", "service", "flag"]
df_processed = pd.get_dummies(df_raw.drop(columns=["label"]), columns=categorical)
```

```
sample_n = min(5000, len(df_processed))
df_sample = df_processed.sample(n=sample_n, random_state=42)
```

```
X = df_sample.drop(columns=["target"])
y = df_sample["target"]
```

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

```
k_values = [1, 3, 5, 7]
summary = []
```

```
def r(x, nd=4):
    return round(x, nd)
```

```
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)
})
```

```
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)
})
```

```
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)
})
```

```
results_df = pd.DataFrame(summary)
```

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

```
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"]})')
print(f'- Быстрее всего предсказывает: {best_pred["model"]} {best_pred["param"]} (predict_time={best_pred["predict_time_s"]})')
```

```
knn_results = results_df[results_df["model"] == "k-NN"]
```

```
k_list = [int(p.split("=")[1]) for p in knn_results["param"]]
recall_list = knn_results["recall_attack"].tolist()
acc_list = knn_results["accuracy"].tolist()
```

```
plt.figure(figsize=(7,5))
plt.plot(k_list, recall_list, marker="o")
plt.xlabel("k")
plt.ylabel("Recall (атака)")
plt.title("Recall vs k (k-NN)")
plt.grid(True)
plt.show()
```

```
plt.figure(figsize=(7,5))
plt.plot(k_list, acc_list, marker="o")
plt.xlabel("k")
plt.ylabel("Accuracy")
plt.title("Accuracy vs k (k-NN)")
plt.grid(True)
plt.show()
```

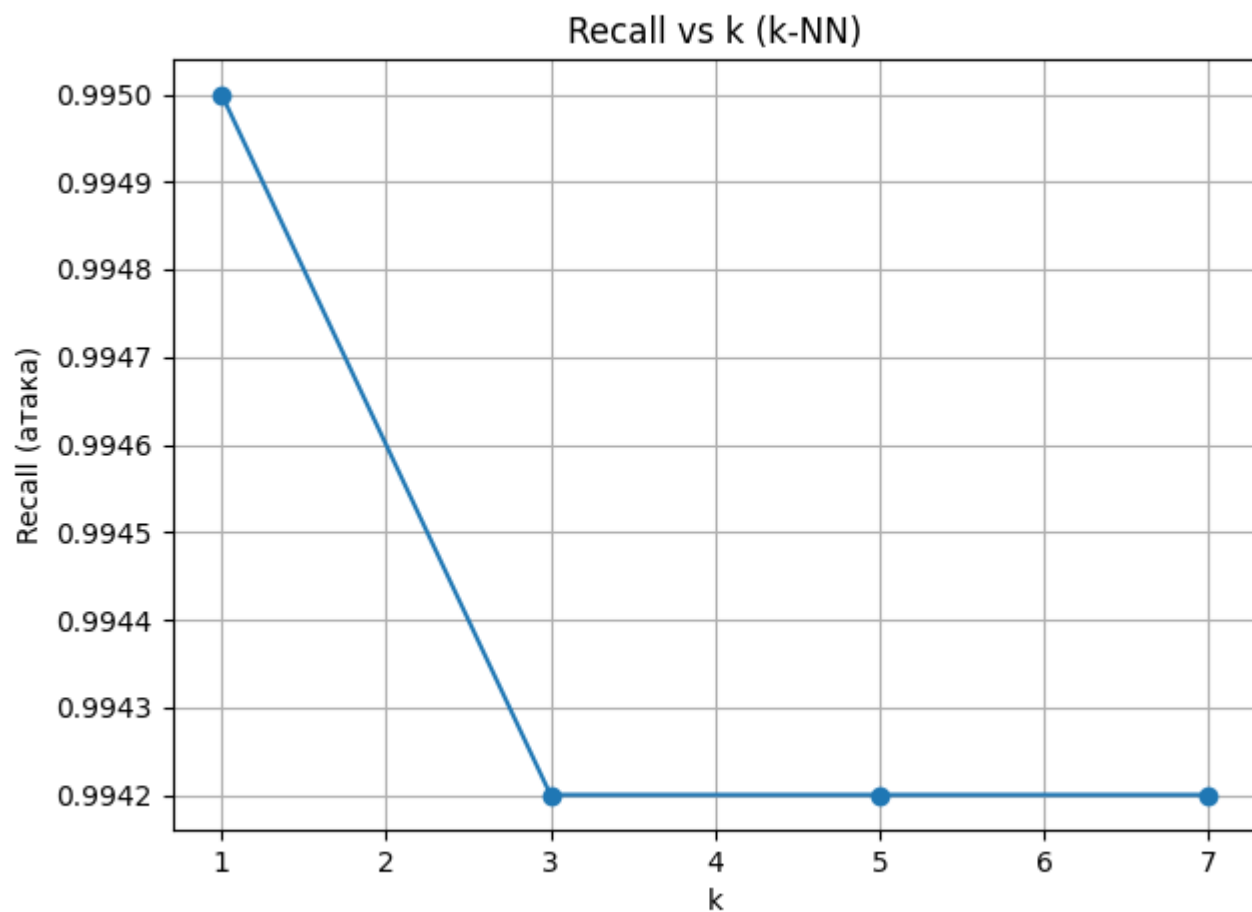
Результаты:

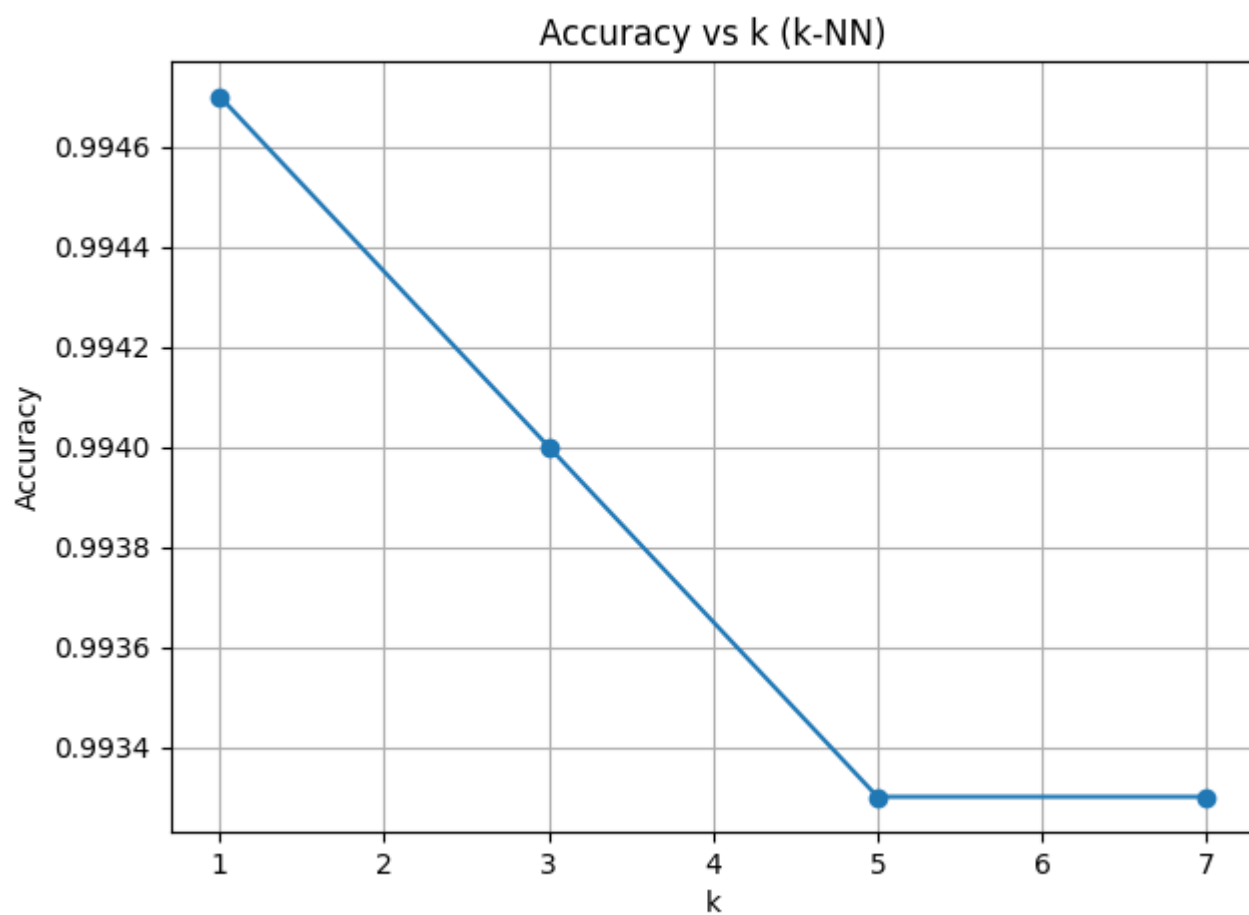
model	param	recall_attack	accuracy	fit_time_s	predict_time_s
k-NN	k=1	0.9950	0.9947	0.0353	2.4583
k-NN	k=3	0.9942	0.9940	0.0273	0.0272
k-NN	k=5	0.9942	0.9933	0.0269	0.0386
k-NN	k=7	0.9942	0.9933	0.0268	0.0301
Decision Tree		0.9967	0.9960	0.0155	0.0079
Linear SVM		0.9967	0.9967	0.3102	0.0067

Короткий анализ:

- Лучшая по recall: Decision Tree (recall=0.9967)
- Быстрее всего обучается: Decision Tree (fit_time=0.0155s)
- Быстрее всего предсказывает: Linear SVM (predict_time=0.0067s)

Figure 1





Вывод: на практике сравнили работу несколько алгоритмов классификации.