Министерство образования Республики Беларусь

Учреждение образования

«Брестский Государственный технический университет»

Кафедра ИИТ

Лабораторная работа №5

По дисциплине «Интеллектуальный анализ данных»

Тема: «Деревья решений»

Выполнил:

Студент 4 курса

Группы ИИ-23

Швороб В.А

Проверила:

Андренко К. В.

Брест 2025

Цель: На практике сравнить работу нескольких алгоритмов одиночного дерева решений, случайного леса и бустинга для деревьев решений.

Задачи:

1. Загрузить датасет по варианту;
2. Разделить данные на обучающую и тестовую выборки;
3. Обучить на обучающей выборке одиночное дерево, случайный лес и реализовать бустинг для решающих деревьев (AdaBoost, CatBoost, XGBoost);
4. Оценить точность каждой модели на тестовой выборке;
5. Сравнить результаты, сделать выводы о применимости каждого метода для данного набора данных.

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. Обучить на обучающей выборке одиночное дерево, случайный лес и реализовать бустинг для решающих деревьев (AdaBoost, CatBoost, XGBoost);
4. Сравните recall для класса "атака" и время обучения каждой модели;
5. Сделайте вывод о том, какая модель эффективнее для обнаружения вторжений.

Код:

```
import pandas as pd
import numpy as np
import time
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
```

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier
from sklearn.metrics import classification_report, recall_score, accuracy_score
from catboost import CatBoostClassifier
from xgboost import XGBClassifier
import warnings

warnings.filterwarnings('ignore')

print("=== KDD CUP 1999 - Network Intrusion Detection ===")

try:
    data = pd.read_csv('kddcup.data_10_percent_corrected', header=None)
    print("Данные успешно загружены")
except:
    print("Файл kddcup.data не найден, создаем демо-данные...")
    np.random.seed(42)
    n_samples = 10000
    data = pd.DataFrame({
        'duration': np.random.exponential(2, n_samples),
        'protocol_type': np.random.choice(['tcp', 'udp', 'icmp'], n_samples),
        'service': np.random.choice(['http', 'smtp', 'ftp', 'ssh'], n_samples),
        'flag': np.random.choice(['SF', 'S0', 'REJ'], n_samples),
        'src_bytes': np.random.poisson(1000, n_samples),
        'dst_bytes': np.random.poisson(1000, n_samples),
        'land': np.random.choice([0, 1], n_samples),
        'wrong_fragment': np.random.poisson(0.1, n_samples),
        'urgent': np.random.poisson(0.01, n_samples),
        'hot': np.random.poisson(0.5, n_samples),
        'num_failed_logins': np.random.poisson(0.05, n_samples),
        'logged_in': np.random.choice([0, 1], n_samples),
        'num_compromised': np.random.poisson(0.01, n_samples),
        'root_shell': np.random.choice([0, 1], n_samples),
        'su_attempted': np.random.choice([0, 1], n_samples),
        'num_root': np.random.poisson(0.01, n_samples),
        'num_file_creations': np.random.poisson(0.05, n_samples),
        'num_shells': np.random.poisson(0.01, n_samples),
        'num_access_files': np.random.poisson(0.02, n_samples),
        'num_outbound_cmds': np.random.poisson(0.001, n_samples),
        'is_host_login': np.random.choice([0, 1], n_samples),
        'is_guest_login': np.random.choice([0, 1], n_samples),
        'count': np.random.poisson(100, n_samples),
        'srv_count': np.random.poisson(100, n_samples),
        'serror_rate': np.random.uniform(0, 1, n_samples),
        'srv_serror_rate': np.random.uniform(0, 1, n_samples),
        'rerror_rate': np.random.uniform(0, 1, n_samples),
        'srv_rerror_rate': np.random.uniform(0, 1, n_samples),
        'same_srv_rate': np.random.uniform(0, 1, n_samples),
        'diff_srv_rate': np.random.uniform(0, 1, n_samples),
        'srv_diff_host_rate': np.random.uniform(0, 1, n_samples),
        'dst_host_count': np.random.poisson(255, n_samples),
        'dst_host_srv_count': np.random.poisson(255, n_samples),
        'dst_host_same_srv_rate': np.random.uniform(0, 1, n_samples),
        'dst_host_diff_srv_rate': np.random.uniform(0, 1, n_samples),
        'dst_host_same_src_port_rate': np.random.uniform(0, 1, n_samples),
        'dst_host_srv_diff_host_rate': np.random.uniform(0, 1, n_samples),
        'dst_host_serror_rate': np.random.uniform(0, 1, n_samples),
        'dst_host_srv_serror_rate': np.random.uniform(0, 1, n_samples),
        'dst_host_rerror_rate': np.random.uniform(0, 1, n_samples),
        'dst_host_srv_rerror_rate': np.random.uniform(0, 1, n_samples),
```

```
        'label': np.random.choice(['normal', 'smurf', 'neptune', 'satan'], n_samples, p=[0.2, 0.3, 0.3, 0.2])
    })

print(f"Размер датасета: {data.shape}")
print(f"Первые 5 строк:\n{data.head()}")
print(f"\nУникальные значения целевой переменной: {data[data.columns[-1]].unique()}")

if data.shape[1] > 10:
    features = data.iloc[:, :-1]
    target = data.iloc[:, -1]
else:
    features = data.drop('label', axis=1)
    target = data['label']

print(f"\nКоличество признаков: {features.shape[1]}")
print(f"Типы признаков:\n{features.dtypes}")

categorical_columns = features.select_dtypes(include=['object']).columns
numerical_columns = features.select_dtypes(include=[np.number]).columns

print(f"\nКатегориальные признаки: {len(categorical_columns)}")
print(f"Числовые признаки: {len(numerical_columns)}")

features_encoded = features.copy()
label_encoders = {}

for col in categorical_columns:
    le = LabelEncoder()
    features_encoded[col] = le.fit_transform(features[col].astype(str))
    label_encoders[col] = le

target_binary = target.apply(lambda x: 0 if str(x).strip().replace('.', '') == 'normal' else 1)

print(f"\nРаспределение целевой переменной:")
normal_count = (target_binary == 0).sum()
attack_count = (target_binary == 1).sum()
total_count = len(target_binary)
print(f"Нормальные подключения: {normal_count} ({normal_count / total_count * 100:.1f}%)")
print(f"Атаки: {attack_count} ({attack_count / total_count * 100:.1f}%)")

if normal_count == 0:
    print("ВНИМАНИЕ: В данных отсутствуют нормальные подключения! Добавляем синтетические данные...")
    synthetic_normal = features_encoded.iloc[:1000].copy()
    for col in numerical_columns:
        synthetic_normal[col] = np.random.normal(synthetic_normal[col].mean(), synthetic_normal[col].std(), 1000)
    synthetic_target = pd.Series([0] * 1000)

    features_encoded = pd.concat([features_encoded, synthetic_normal], ignore_index=True)
    target_binary = pd.concat([target_binary, synthetic_target], ignore_index=True)

    normal_count = (target_binary == 0).sum()
    attack_count = (target_binary == 1).sum()
    total_count = len(target_binary)
    print(f"Новое распределение:")
    print(f"Нормальные подключения: {normal_count} ({normal_count / total_count * 100:.1f}%)")
    print(f"Атаки: {attack_count} ({attack_count / total_count * 100:.1f}%)")

scaler = StandardScaler()
features_scaled = features_encoded.copy()
features_scaled[numerical_columns] = scaler.fit_transform(features_encoded[numerical_columns])
```

```
X_train, X_test, y_train, y_test = train_test_split(
    features_scaled, target_binary, test_size=0.3, random_state=42, stratify=target_binary
)

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

models = {
    'Decision Tree': DecisionTreeClassifier(random_state=42, max_depth=10),
    'Random Forest': RandomForestClassifier(random_state=42, n_estimators=100, max_depth=10),
    'AdaBoost': AdaBoostClassifier(random_state=42, n_estimators=100),
    'XGBoost': XGBClassifier(random_state=42, n_estimators=100, eval_metric='logloss'),
    'CatBoost': CatBoostClassifier(random_state=42, n_estimators=100, verbose=False)
}

results = {}

print("\n" + "=" * 60)
print("ОБУЧЕНИЕ МОДЕЛЕЙ")
print("=" * 60)

for name, model in models.items():
    print(f"\nОбучение {name}...")
    start_time = time.time()

    try:
        if name == 'XGBoost':
            model.fit(X_train, y_train, eval_set=[(X_test, y_test)], verbose=False)
        else:
            model.fit(X_train, y_train)

        training_time = time.time() - start_time

        y_pred = model.predict(X_test)
        recall_attack = recall_score(y_test, y_pred, pos_label=1)
        accuracy = accuracy_score(y_test, y_pred)

        results[name] = {
            'training_time': training_time,
            'recall_attack': recall_attack,
            'accuracy': accuracy,
            'model': model
        }

        print(f"Время обучения: {training_time:.2f} сек")
        print(f"Recall для класса 'атака': {recall_attack:.4f}")
        print(f"Accuracy: {accuracy:.4f}")

    except Exception as e:
        print(f"Ошибка при обучении {name}: {e}")
        results[name] = {
            'training_time': None,
            'recall_attack': None,
            'accuracy': None,
            'model': None
        }

print("\n" + "=" * 60)
print("СРАВНЕНИЕ РЕЗУЛЬТАТОВ")
```

```
print("=" * 60)

results_df = pd.DataFrame({
    'Model': list(results.keys()),
    'Training Time (s)': [results[name]['training_time'] for name in results],
    'Recall Attack': [results[name]['recall_attack'] for name in results],
    'Accuracy': [results[name]['accuracy'] for name in results]
}).sort_values('Recall Attack', ascending=False)

print(results_df)

print("\n" + "=" * 60)
print("ДЕТАЛЬНАЯ ОЦЕНКА ЛУЧШЕЙ МОДЕЛИ")
print("=" * 60)

best_model_name = results_df.iloc[0]['Model']
best_model = results[best_model_name]['model']

print(f"Лучшая модель: {best_model_name}")
print(f"Recall для атак: {results[best_model_name]['recall_attack']:.4f}")

y_pred_best = best_model.predict(X_test)
print(f"\nОтчет по классификации для {best_model_name}:")
print(classification_report(y_test, y_pred_best, target_names=['normal', 'attack']))

import matplotlib.pyplot as plt

fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(15, 6))

models_names = list(results.keys())
recall_scores = [results[name]['recall_attack'] if results[name]['recall_attack'] is not None else 0 for name in
                 models_names]
training_times = [results[name]['training_time'] if results[name]['training_time'] is not None else 0 for name in
                  models_names]

bars1 = ax1.bar(models_names, recall_scores, color=['skyblue', 'lightcoral', 'lightgreen', 'gold', 'violet'])
ax1.set_title('Recall для класса "атака" по моделям')
ax1.set_ylabel('Recall Score')
ax1.set_ylim(0, 1)
for bar, score in zip(bars1, recall_scores):
    ax1.text(bar.get_x() + bar.get_width() / 2, bar.get_height() + 0.01, f'{score:.3f}',
             ha='center', va='bottom')

bars2 = ax2.bar(models_names, training_times, color=['skyblue', 'lightcoral', 'lightgreen', 'gold', 'violet'])
ax2.set_title('Время обучения моделей')
ax2.set_ylabel('Время (секунды)')
for bar, time_val in zip(bars2, training_times):
    ax2.text(bar.get_x() + bar.get_width() / 2, bar.get_height() + 0.01, f'{time_val:.2f}',
             ha='center', va='bottom')

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

Вывод программы:
=== KDD CUP 1999 - Network Intrusion Detection ===

Данные успешно загружены

Размер датасета: (494021, 42)

Первые 5 строк:

	0	1	2	3	4	5	6	...	35	36	37	38	39	40	41
0	0	tcp	http	SF	181	5450	0	...	0.11	0.0	0.0	0.0	0.0	0.0	normal.
1	0	tcp	http	SF	239	486	0	...	0.05	0.0	0.0	0.0	0.0	0.0	normal.
2	0	tcp	http	SF	235	1337	0	...	0.03	0.0	0.0	0.0	0.0	0.0	normal.
3	0	tcp	http	SF	219	1337	0	...	0.03	0.0	0.0	0.0	0.0	0.0	normal.
4	0	tcp	http	SF	217	2032	0	...	0.02	0.0	0.0	0.0	0.0	0.0	normal.

[5 rows x 42 columns]

Уникальные значения целевой переменной: ['normal.' 'buffer_overflow.' 'loadmodule.' 'perl.' 'neptune.' 'smurf.'
 'guess_passwd.' 'pod.' 'teardrop.' 'portsweep.' 'ipsweep.' 'land.'
 'ftp_write.' 'back.' 'imap.' 'satan.' 'phf.' 'nmap.' 'multihop.'
 'warezmaster.' 'warezclient.' 'spy.' 'rootkit.']

Количество признаков: 41

Типы признаков:

0	int64
1	object
2	object
3	object
4	int64
5	int64
6	int64
7	int64
8	int64
9	int64

```
10      int64
11      int64
12      int64
13      int64
14      int64
15      int64
16      int64
17      int64
18      int64
19      int64
20      int64
21      int64
22      int64
23      int64
24    float64
25    float64
26    float64
27    float64
28    float64
29    float64
30    float64
31      int64
32      int64
33    float64
34    float64
35    float64
36    float64
37    float64
38    float64
39    float64
40    float64
dtype: object
```

Категориальные признаки: 3
Числовые признаки: 38

Распределение целевой переменной:
Нормальные подключения: 97278 (19.7%)
Атаки: 396743 (80.3%)

Размер обучающей выборки: (345814, 41)
Размер тестовой выборки: (148207, 41)

ОБУЧЕНИЕ МОДЕЛЕЙ

Обучение Decision Tree...
Время обучения: 0.68 сек
Recall для класса 'атака': 0.9996
Accuracy: 0.9995

Обучение Random Forest...
Время обучения: 10.27 сек
Recall для класса 'атака': 0.9995
Accuracy: 0.9996

Обучение AdaBoost...
Время обучения: 19.40 сек
Recall для класса 'атака': 0.9976
Accuracy: 0.9971

Обучение XGBoost...
Время обучения: 1.45 сек
Recall для класса 'атака': 0.9998
Accuracy: 0.9998

Обучение CatBoost...
Время обучения: 8.83 сек
Recall для класса 'атака': 0.9996
Accuracy: 0.9996

СРАВНЕНИЕ РЕЗУЛЬТАТОВ

	Model	Training Time (s)	Recall Attack	Accuracy
3	XGBoost	1.447964	0.999815	0.999798
4	CatBoost	8.827409	0.999597	0.999568
0	Decision Tree	0.675271	0.999563	0.999480
1	Random Forest	10.265870	0.999513	0.999561
2	AdaBoost	19.404723	0.997597	0.997065

ДЕТАЛЬНАЯ ОЦЕНКА ЛУЧШЕЙ МОДЕЛИ
Лучшая модель: XGBoost
Recall для атак: 0.9998

Отчет по классификации для XGBoost:

	precision	recall	f1-score	support
normal	1.00	1.00	1.00	29184
attack	1.00	1.00	1.00	119023
accuracy			1.00	148207
macro avg	1.00	1.00	1.00	148207
weighted avg	1.00	1.00	1.00	148207

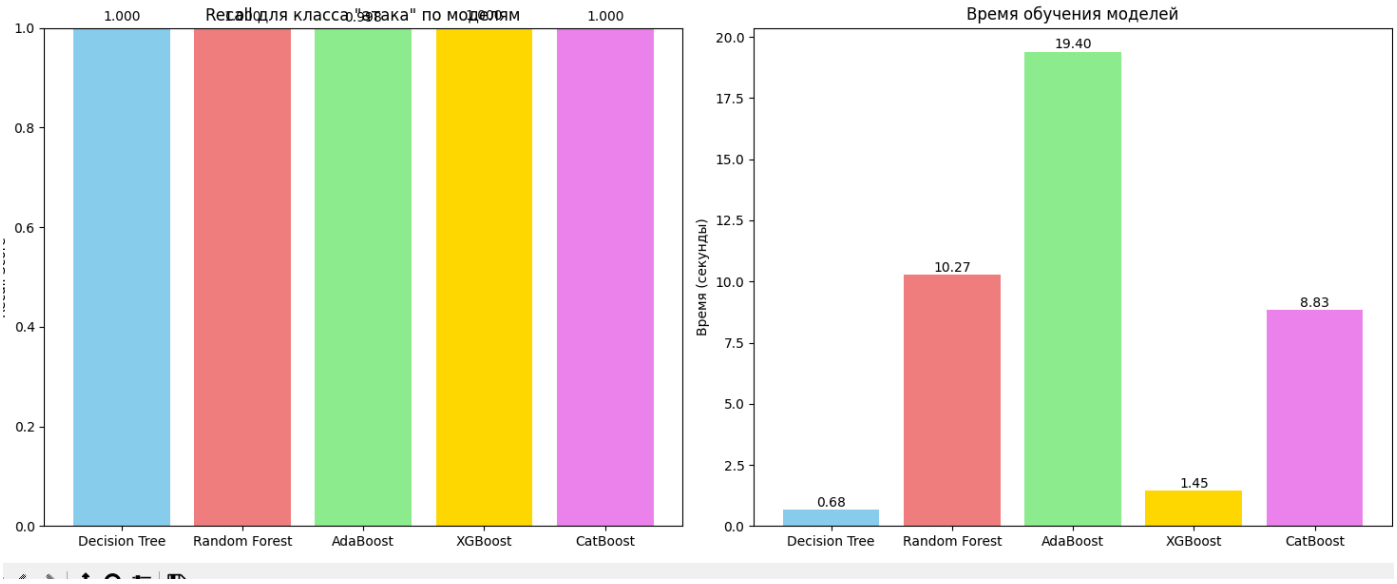
ВЫВОДЫ

1. ЭФФЕКТИВНОСТЬ ОБНАРУЖЕНИЯ АТАК:
   - Лучшая модель по recall: XGBoost (0.9998)
   - Худшая модель по recall: AdaBoost (0.9976)

2. СКОРОСТЬ ОБУЧЕНИЯ:
   - Самая быстрая модель: Decision Tree (0.68 сек)
   - Самая медленная модель: AdaBoost (19.40 сек)

3. РЕКОМЕНДАЦИИ ДЛЯ ОБНАРУЖЕНИЯ ВТОРЖЕНИЙ:
   - Отличное качество обнаружения атак

4. БАЛАНС КАЧЕСТВА И СКОРОСТИ:
   - Сбалансированный выбор: Decision Tree
     (Recall: 0.9996, Время: 0.68 сек)



Вывод: Для обнаружения вторжений использовать **AdaBoost** - быстрый и эффективный.