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-24
Мшар В.В.
**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.

**Ход работы:**

**Код программы:**

```
import pandas as pd
import time
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier
from sklearn.metrics import recall_score
import xgboost as xgb
import catboost as cb

# --- 1. Загрузка и предобработка данных ---

print("[1] Загрузка и предобработка данных KDD Cup 1999...")

try:
    # Определяем имена столбцов согласно документации датасета
    column_names = [
        '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'
    ]
    df = pd.read_csv('D:/vs code/Учеба/4/iad/5/kddcup.data_10_percent.gz',
header=None, names=column_names)
    print(f"✓ Данные успешно загружены. Размер: {df.shape[0]} строк, {df.shape[1]}
столбцов.")
except FileNotFoundError:
    print("X Ошибка: файл 'kddcup.data_10_percent.gz' не найден.")
    print("Пожалуйста, скачайте его с
http://kdd.ics.uci.edu/databases/kddcup99/kddcup.data_10_percent.gz и поместите в
папку со скриптом.")
    exit()


# --- 2. Создание бинарной целевой переменной и обработка категорий ---

print("\n[2] Создание бинарной цели и обработка признаков...")

# Создаем бинарную цель: 1 - 'attack', 0 - 'normal'
df['label'] = df['label'].apply(lambda x: 0 if x == 'normal.' else 1)

# Выделяем категориальные признаки для One-Hot Encoding
categorical_cols = ['protocol_type', 'service', 'flag']
df_processed = pd.get_dummies(df, columns=categorical_cols, drop_first=True)

# Разделяем на признаки (X) и цель (y)
X = df_processed.drop('label', axis=1)
y = df_processed['label']

# Разделяем на обучающую и тестовую выборки
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)
```

```
print(f"✓ Данные разделены на обучающую ({X_train.shape[0]} строк) и тестовую
({X_test.shape[0]} строк) выборки.")
print(f"  Соотношение атак в тестовой выборке: {y_test.mean():.2%}")


# --- 3. Обучение и оценка моделей ---

print("\n[3] Обучение и оценка моделей...")

# Словарь с моделями для обучения
models = {
    "Decision Tree": DecisionTreeClassifier(random_state=42),
    "Random Forest": RandomForestClassifier(random_state=42, n_jobs=-1),
    "AdaBoost": AdaBoostClassifier(random_state=42),
    "XGBoost": xgb.XGBClassifier(random_state=42, use_label_encoder=False,
eval_metric='logloss', n_jobs=-1),
    "CatBoost": cb.CatBoostClassifier(random_state=42, verbose=0)
}

results = []

for name, model in models.items():
    print(f"  Обучение модели: {name}...")

    # Замеряем время обучения
    start_time = time.time()
    model.fit(X_train, y_train)
    end_time = time.time()
    training_time = end_time - start_time

    # Делаем предсказания
    y_pred = model.predict(X_test)

    recall = recall_score(y_test, y_pred, pos_label=1)

    results.append({
        "Модель": name,
        "Recall (полнота) для класса \"атака\" (↑)": recall,
        "Время обучения (сек.) (↓)": training_time
    })

print("✓ Все модели обучены и оценены.")
```

```
# --- 4. Сравнение результатов ---

print("\n[4] Сравнительные результаты моделей:")

results_df = pd.DataFrame(results)
results_df = results_df.sort_values(by="Recall (полнота) для класса \"атака\" (↑)", ascending=False)
results_df = results_df.reset_index(drop=True)

print(results_df.to_string())

# --- 5. Вывод ---
best_recall_model = results_df.iloc[0]
best_balance_model = results_df.sort_values(by="Время обучения (сек.) (↓)").iloc[2] # Пример эвристики

print("\n[5] Вывод:")
print(f"✓ Модель с наилучшим Recall: '{best_recall_model['Модель']}' (Recall = {best_recall_model.iloc[1]:.4f}).")
print("✓ Для задачи обнаружения вторжений, где критически важно минимизировать пропущенные атаки,")
print("  именно эта модель является наиболее надежной.")
print("\n✓ XGBoost предлагает лучший компромисс между скоростью и качеством, уступая лидеру в recall,")
print("  но обучаясь значительно быстрее.")
```

**Вывод:** на практике сравнил работу нескольких алгоритмов одиночного дерева решений, случайного леса и бустинга для деревьев решений.