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
Бусень А.Д.
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 1:
`Iris`
Определить вид ириса (setosa, versicolor, virginica) по измерениям его цветка
Задания:

1. Загрузите данные и ознакомьтесь с ними;
2. Разделите выборку на обучающую и тестовую;
3. Обучить на обучающей выборке одиночное дерево, случайный лес и реализовать бустинг для решающих деревьев (AdaBoost, CatBoost, XGBoost);
4. Оцените точность (accuracy) каждой модели на тестовой выборке;
5. Сравните результаты и сделайте вывод, какая модель лучше всего справилась с этой задачей.

Код программы:

```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import numpy as np
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier

try:
    from xgboost import XGBClassifier
    xgb_available = True
except:
    xgb_available = False

try:
    from catboost import CatBoostClassifier
    cat_available = True
except:
    cat_available = False

data = load_iris()
X = pd.DataFrame(data.data, columns=data.feature_names)
y = pd.Series(data.target)
```

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

models = {}
models["Decision Tree"] = DecisionTreeClassifier(random_state=42)
models["Random Forest"] = RandomForestClassifier(random_state=42)
models["AdaBoost"] = AdaBoostClassifier(random_state=42)

if xgb_available:
    models["XGBoost"] = XGBClassifier(
        objective="multi:softprob",
        eval_metric="mlogloss",
        random_state=42
    )

if cat_available:
    models["CatBoost"] = CatBoostClassifier(verbose=0, random_state=42)

results = {}

for name, model in models.items():
    model.fit(X_train, y_train)
    pred = model.predict(X_test)
    acc = accuracy_score(y_test, pred)
    results[name] = (model, pred, acc)

print("Accuracy results:")
for name, (_, _, acc) in results.items():
    print(f"{name}: {acc:.4f}")

plt.figure(figsize=(15, 10))

for i, (name, (_, pred, _)) in enumerate(results.items(), 1):
    plt.subplot(2, 3, i)
    cm = confusion_matrix(y_test, pred)
    sns.heatmap(cm, annot=True, fmt="d", cmap="Blues",
                xticklabels=data.target_names,
                yticklabels=data.target_names)
    plt.title(name)

plt.tight_layout()
plt.show()
plt.figure(figsize=(15, 10))

for i, (name, (model, _, _)) in enumerate(results.items(), 1):
    if name == "CatBoost":
        importances = model.get_feature_importance()
    elif name == "XGBoost":
        importances = model.feature_importances_
```

```
    else:
        if hasattr(model, "feature_importances_"):
            importances = model.feature_importances_
        else:
            continue

    plt.subplot(2, 3, i)
    plt.barh(X.columns, importances)
    plt.title(f"Feature importance: {name}")
    plt.xlabel("Importance")

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

Результат работы программы:

Accuracy results:

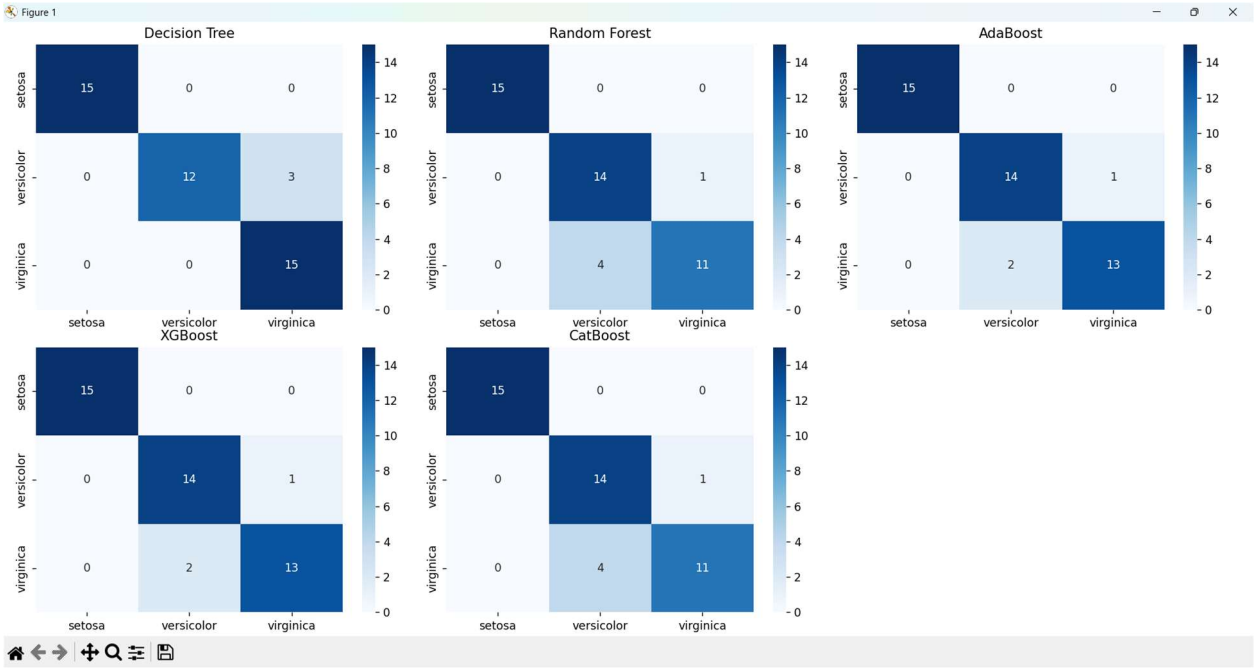
Decision Tree: 0.9333

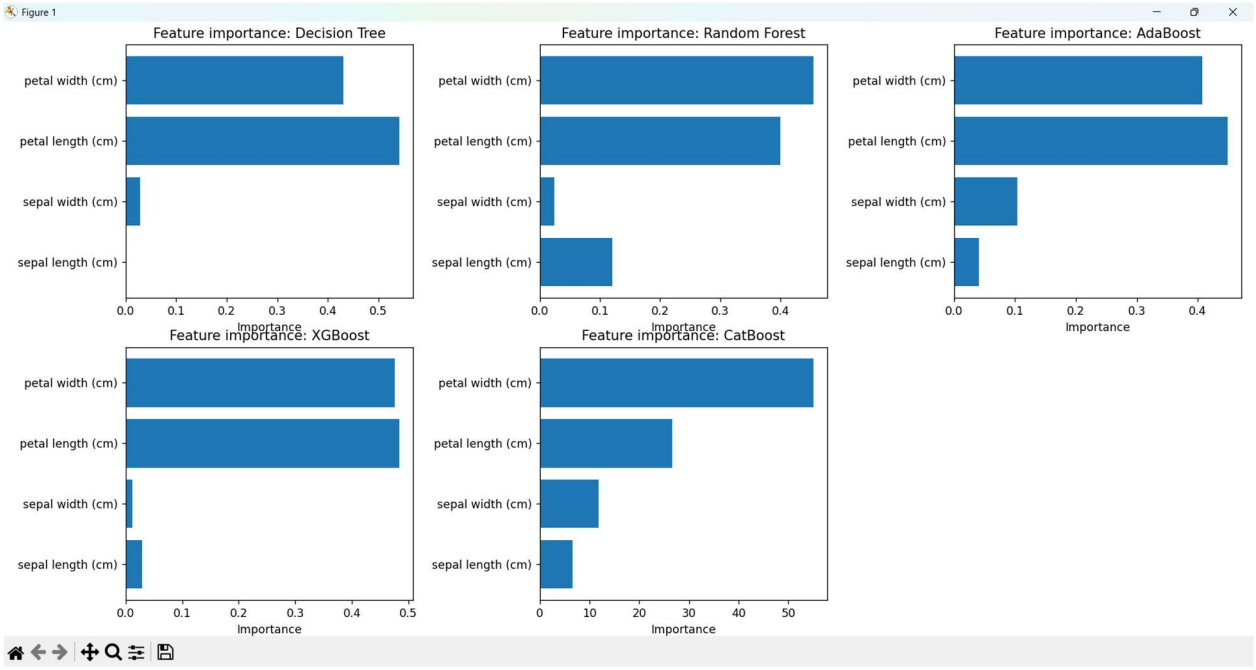
Random Forest: 0.8889

AdaBoost: 0.9333

XGBoost: 0.9333

CatBoost: 0.8889





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