Министерство образования Республики Беларусь
Учреждение образования
«Брестский государственный технический университет»
Кафедра ИИТ

Лабораторная работа №5

По дисциплине: «ИАД»

Тема: «**Деревья решений**»

Выполнил:
Студент 4 курса
Группы ИИ-23
Ежевский Е.Р.
Проверила:
Андренко К.В.

Брест 2025

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

Общее задание

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.

Задание по вариантам

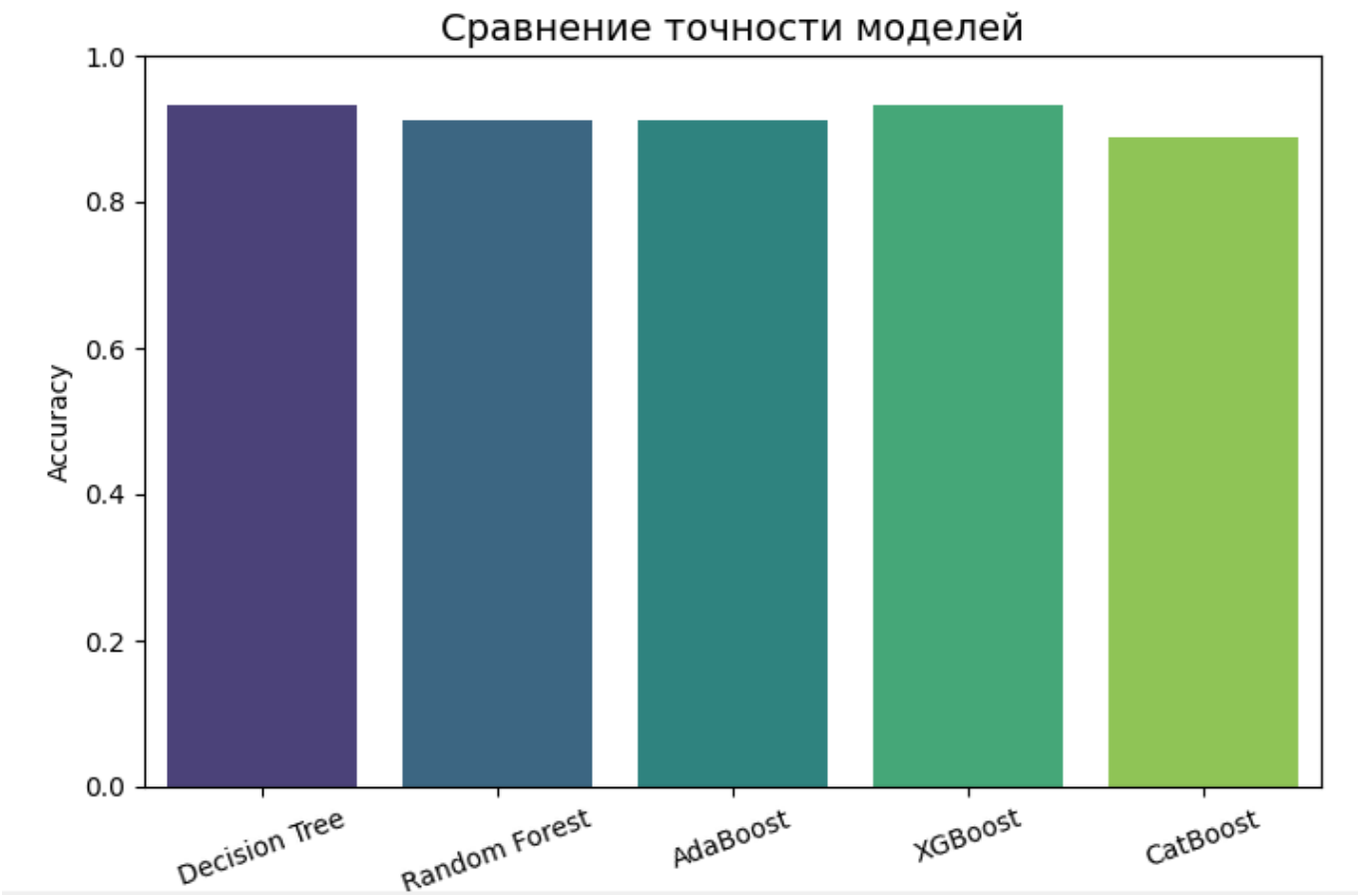
- **Задания:**
    1. Загрузите данные и стандартизируйте (Для деревьев стандартизация не нужна, на лекции же говорили) их;
    2. Разделите выборку;
    3. Обучить на обучающей выборке одиночное дерево, случайный лес и реализовать бустинг для решающих деревьев (AdaBoost, CatBoost, XGBoost);
    4. Сравните производительность моделей с помощью classification_report (sklearn.metrics);
    5. Укажите, какой класс модели определяют хуже всего, и предположите, почему.

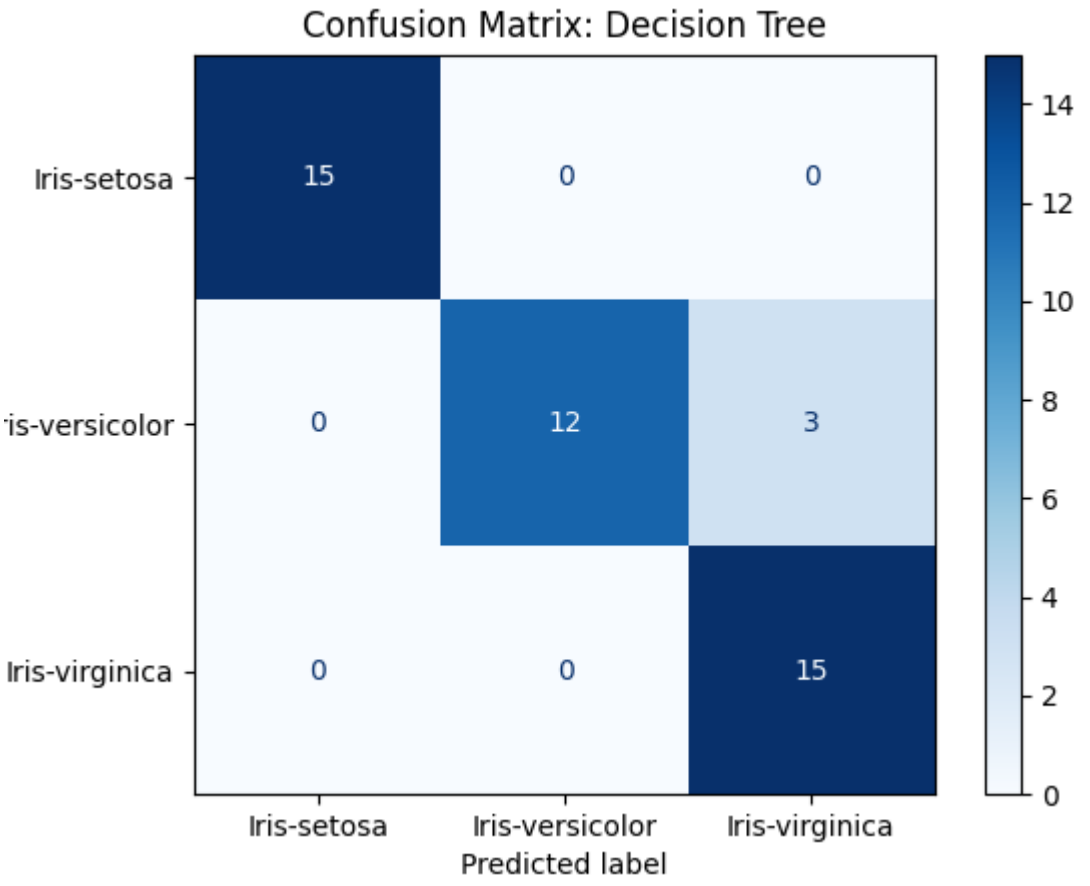
Код программы:

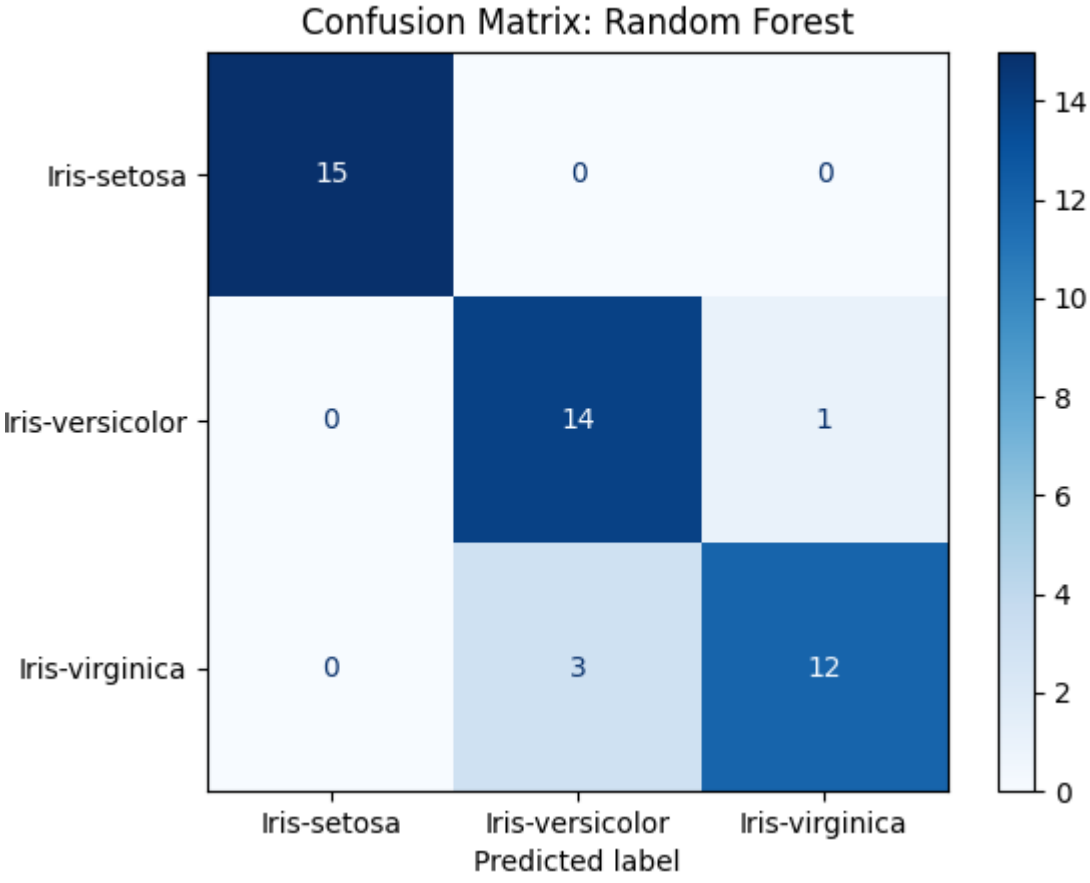
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier
from xgboost import XGBClassifier
from catboost import CatBoostClassifier
from sklearn.metrics import classification_report, confusion_matrix,
ConfusionMatrixDisplay, accuracy_score
import matplotlib.pyplot as plt
import seaborn as sns

df = pd.read_csv("iris.csv")
df = df.drop("Id", axis=1)
X = df.drop("Species", axis=1)
le = LabelEncoder()
y = le.fit_transform(df["Species"])
X_train, X_test, y_train, y_test = train_test_split(
   X, y, test_size=0.3, random_state=42, stratify=y
)
models = {
   "Decision Tree": DecisionTreeClassifier(random_state=42),
```

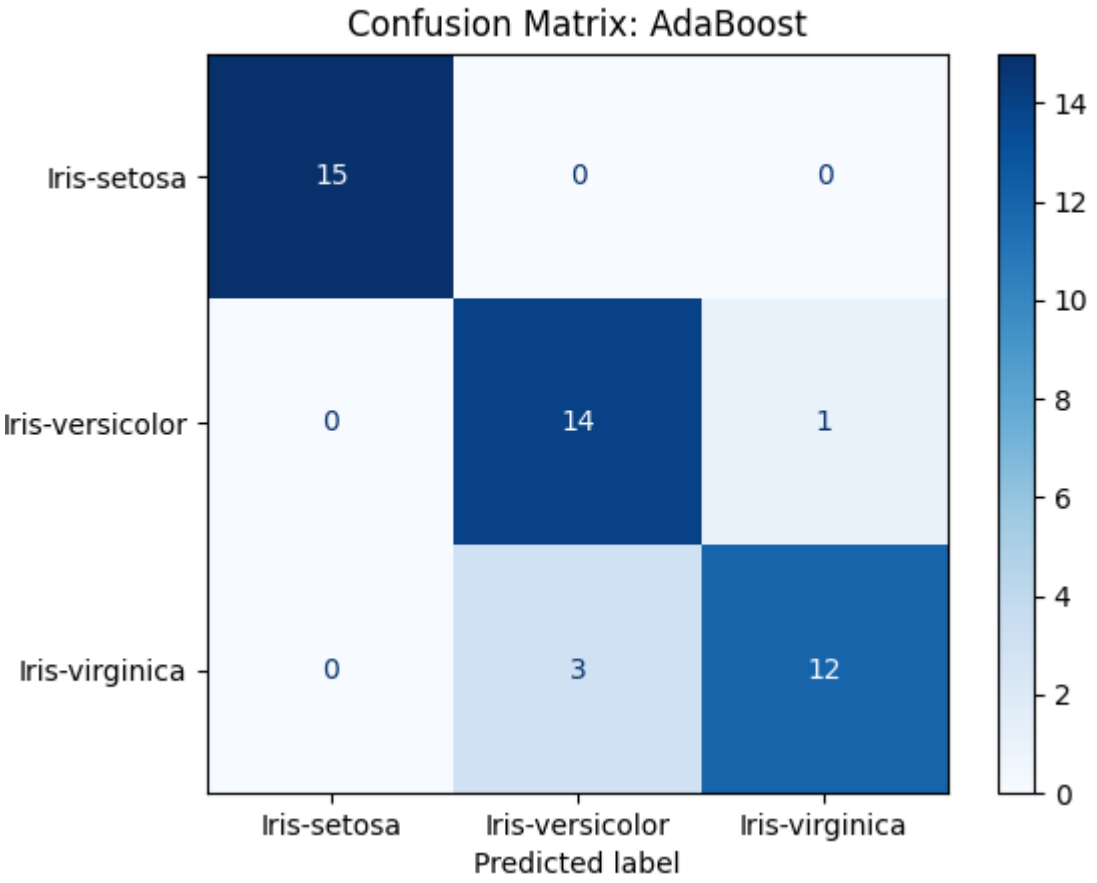
```
    "Random Forest": RandomForestClassifier(n_estimators=200,
random_state=42),
    "AdaBoost": AdaBoostClassifier(n_estimators=200, random_state=42),
    "XGBoost": XGBClassifier(n_estimators=200, random_state=42,
eval_metric='mlogloss', use_label_encoder=False),
    "CatBoost": CatBoostClassifier(iterations=200, verbose=0, random_seed=42)
}

results = {}

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

plt.figure(figsize=(8,5))
sns.barplot(x=list(results.keys()), y=[r["acc"] for r in results.values()],
palette="viridis")
plt.title("Сравнение точности моделей", fontsize=14)
plt.ylabel("Accuracy")
plt.ylim(0, 1)
plt.xticks(rotation=20)
plt.show()

for name, res in results.items():
    cm = confusion_matrix(y_test, res["pred"])
    disp = ConfusionMatrixDisplay(cm, display_labels=le.classes_)
    disp.plot(cmap="Blues", values_format='d')
    plt.title(f"Confusion Matrix: {name}")
    plt.show()

plt.figure(figsize=(14, 14))
plot_tree(
    results["Decision Tree"]["model"],
    feature_names=X.columns,
    class_names=le.classes_,
    filled=True, rounded=True, fontsize=8
)
plt.title("Decision Tree Structure")
plt.show()

def plot_feature_importance(model, model_name):
    if hasattr(model, "feature_importances_"):
        importances = model.feature_importances_
        indices = np.argsort(importances)[::-1]
        plt.figure(figsize=(8, 4))
        sns.barplot(x=importances[indices], y=np.array(X.columns)[indices],
palette="mako")
        plt.title(f"Feature Importance: {model_name}")
        plt.xlabel("Importance")
```

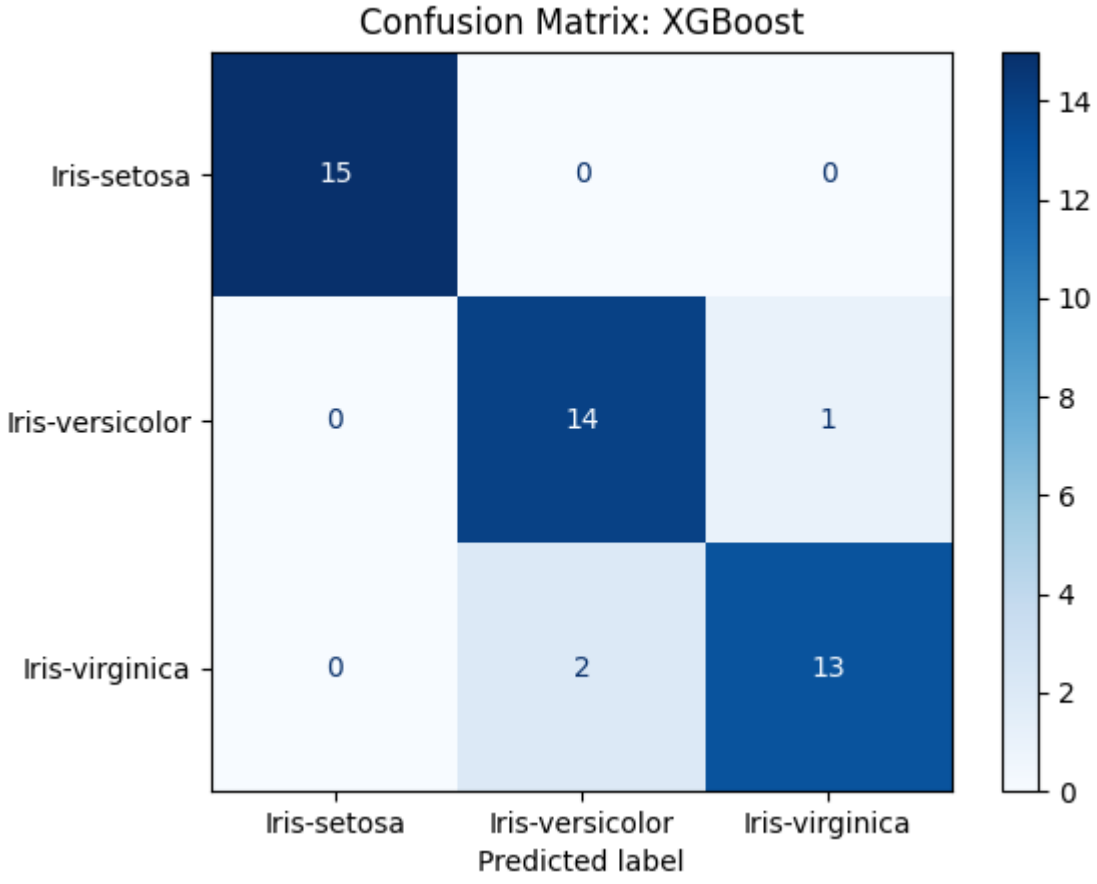
```
        plt.ylabel("Feature")
        plt.show()

for name, res in results.items():
    plot_feature_importance(res["model"], name)

print("Сравнение точности:")
for name, res in results.items():
    print(f"{name:15s} — Accuracy: {res['acc']:.4f}")

best_model = max(results.items(), key=lambda x: x[1]['acc'])
print(f"\nЛучшая модель: {best_model[0]} с точностью
{best_model[1]['acc']:.4f}")

print("\n=== Classification Reports ===")
for name, res in results.items():
    print(f"\n--- {name} ---")
    print(classification_report(y_test, res["pred"], target_names=le.classes_))
```

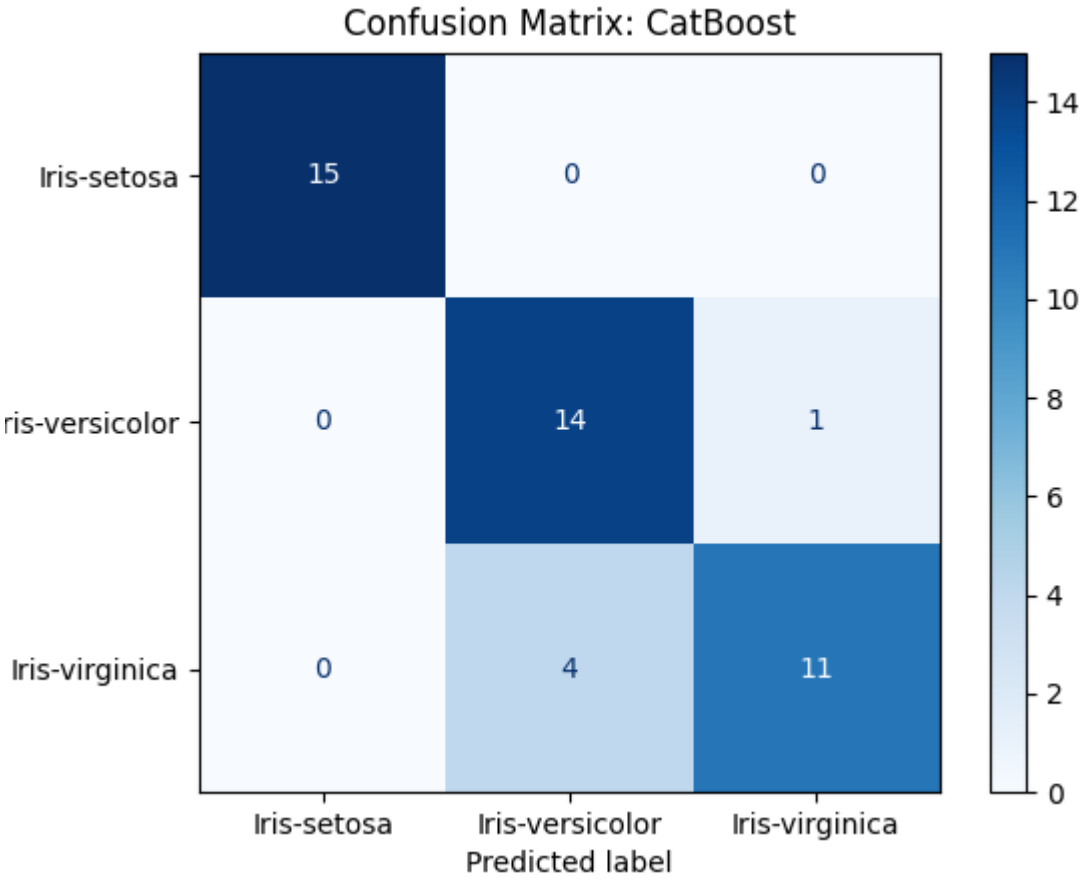


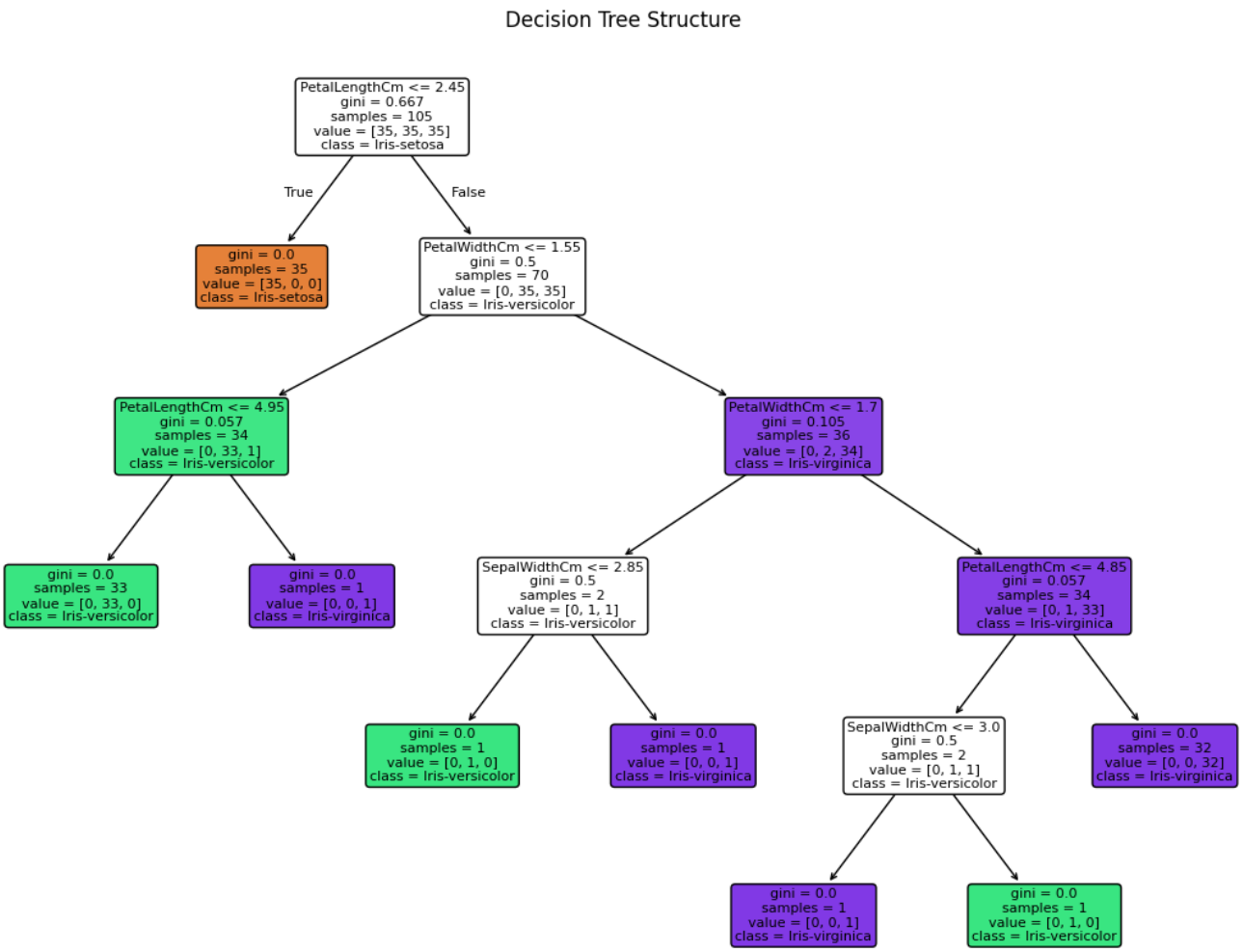

Confusion Matrix: Decision Tree
Iris-setosa
is-versicolor
Iris-virginica
15
0
0
0
12
3
0
0
15
Iris-setosa
Iris-versicolor
Iris-virginica
Predicted label
14
12
10
8
6
4
2
0

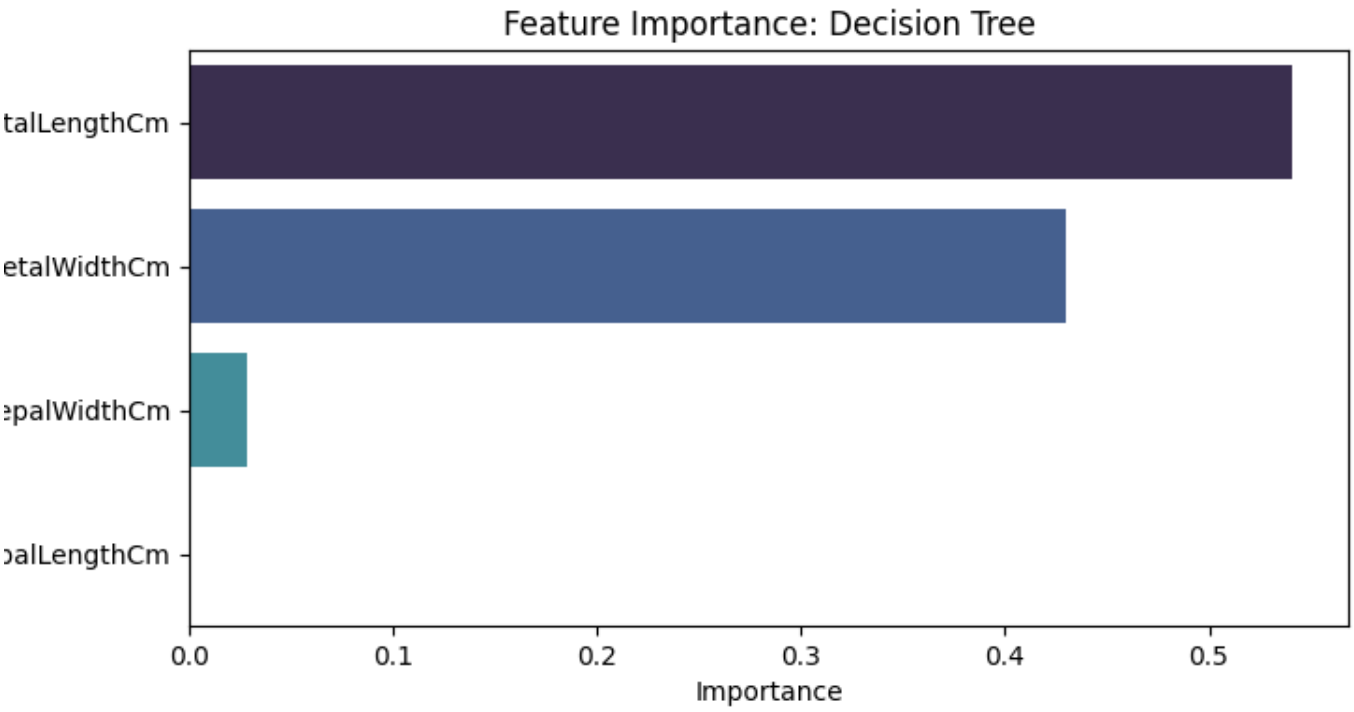

Confusion Matrix: Random Forest
Iris-setosa
Iris-versicolor
Iris-virginica
15
0
0
0
14
1
0
3
12
Iris-setosa
Iris-versicolor
Iris-virginica
Predicted label
14
12
10
8
6
4
2
0

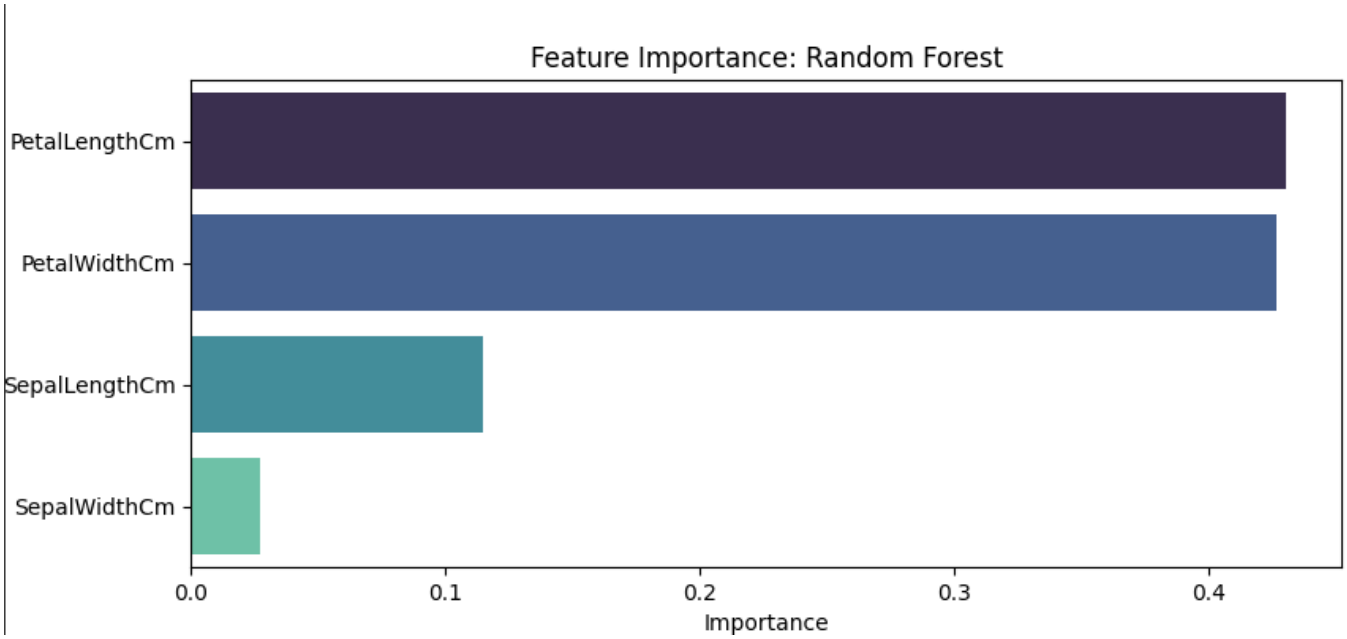

Confusion Matrix: AdaBoost
Iris-setosa
Iris-versicolor
Iris-virginica
15
0
0
0
14
1
0
3
12
Iris-setosa
Iris-versicolor
Iris-virginica
Predicted label
14
12
10
8
6
4
2
0

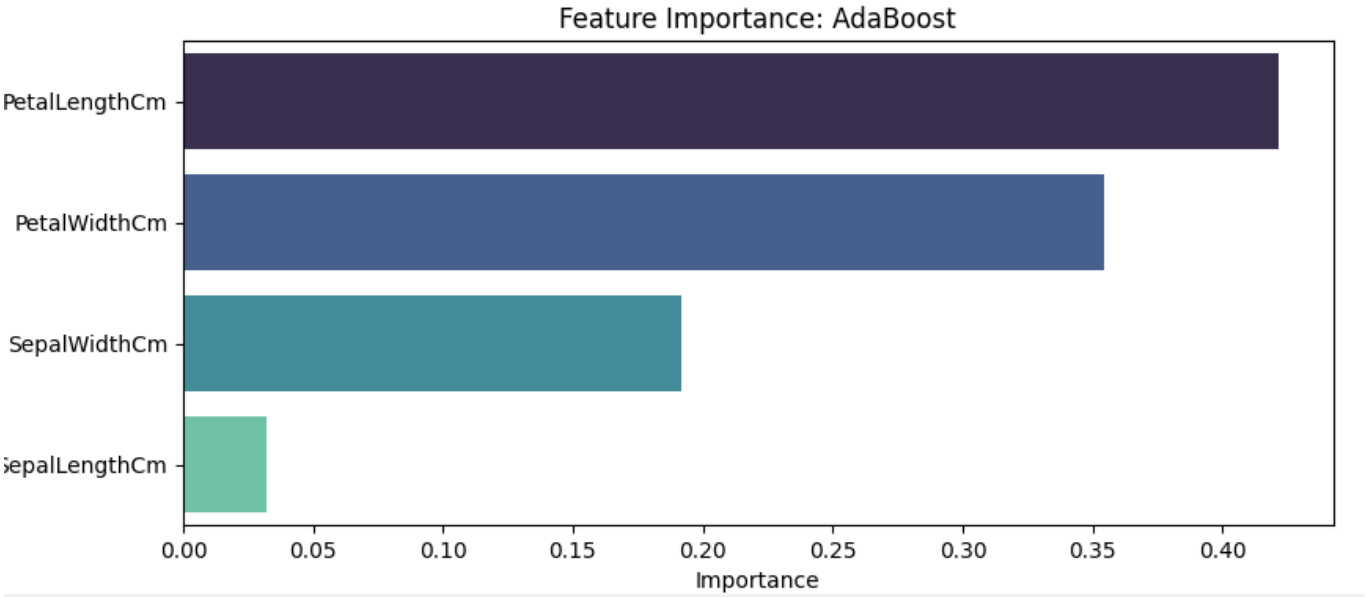

Confusion Matrix: XGBoost
Iris-setosa
Iris-versicolor
Iris-virginica
15
0
0
0
14
1
0
2
13
Iris-setosa
Iris-versicolor
Iris-virginica
Predicted label
14
12
10
8
6
4
2
0

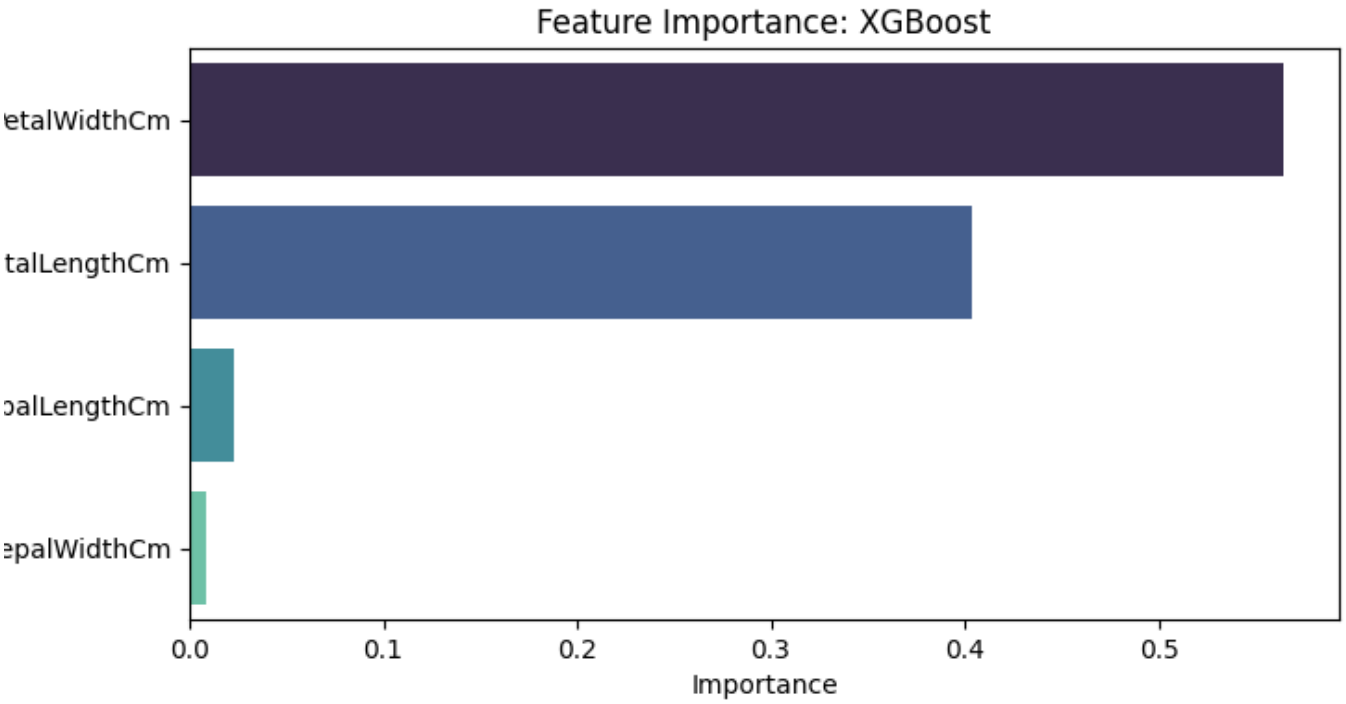

Confusion Matrix: CatBoost
Iris-setosa
ris-versicolor
Iris-virginica
15
0
0
0
14
1
0
4
11
Iris-setosa
Iris-versicolor
Iris-virginica
Predicted label
14
12
10
8
6
4
2
0

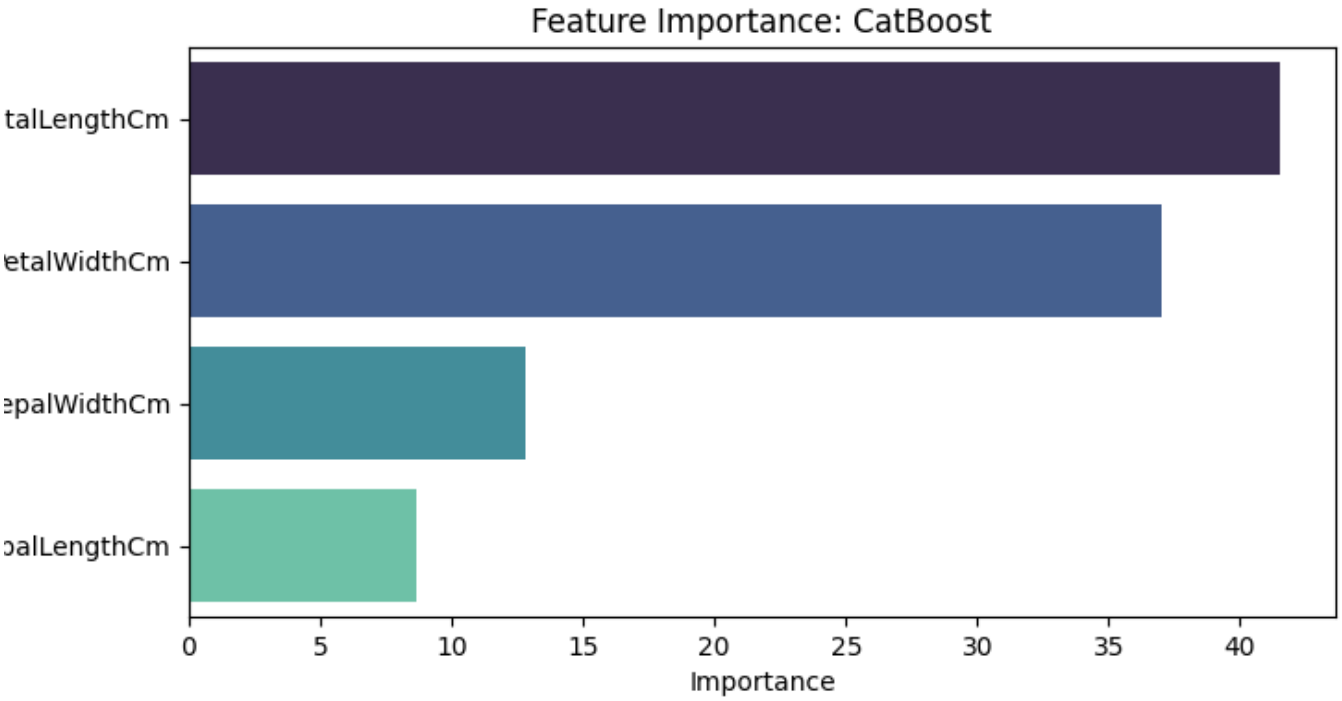

Decision Tree Structure
PetalLengthCm <= 2.45
gini = 0.667
samples = 105
value = [35, 35, 35]
class = Iris-setosa
True
False
gini = 0.0
samples = 35
value = [35, 0, 0]
class = Iris-setosa
PetalWidthCm <= 1.55
gini = 0.5
samples = 70
value = [0, 35, 35]
class = Iris-versicolor
PetalLengthCm <= 4.95
gini = 0.057
samples = 34
value = [0, 33, 1]
class = Iris-versicolor
PetalWidthCm <= 1.7
gini = 0.105
samples = 36
value = [0, 2, 34]
class = Iris-virginica
gini = 0.0
samples = 33
value = [0, 33, 0]
class = Iris-versicolor
gini = 0.0
samples = 1
value = [0, 0, 1]
class = Iris-virginica
SepalWidthCm <= 2.85
gini = 0.5
samples = 2
value = [0, 1, 1]
class = Iris-versicolor
PetalLengthCm <= 4.85
gini = 0.057
samples = 34
value = [0, 1, 33]
class = Iris-virginica
gini = 0.0
samples = 1
value = [0, 1, 0]
class = Iris-versicolor
gini = 0.0
samples = 1
value = [0, 0, 1]
class = Iris-virginica
SepalWidthCm <= 3.0
gini = 0.5
samples = 2
value = [0, 1, 1]
class = Iris-versicolor
gini = 0.0
samples = 32
value = [0, 0, 32]
class = Iris-virginica
gini = 0.0
samples = 1
value = [0, 0, 1]
class = Iris-virginica
gini = 0.0
samples = 1
value = [0, 1, 0]
class = Iris-versicolor


Feature Importance: Decision Tree
talLengthCm
etalWidthCm
epalWidthCm
palLengthCm
0.0
0.1
0.2
0.3
0.4
0.5
Importance


Feature Importance: Random Forest
PetalLengthCm
PetalWidthCm
SepalLengthCm
SepalWidthCm
0.0
0.1
0.2
0.3
0.4
Importance


Feature Importance: AdaBoost
PetalLengthCm
PetalWidthCm
SepalWidthCm
epalLengthCm
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0.40
Importance





Сравнение точности:

Decision Tree — Accuracy: 0.9333

Random Forest — Accuracy: 0.9111

AdaBoost — Accuracy: 0.9111

XGBoost — Accuracy: 0.9333

CatBoost — Accuracy: 0.8889

Лучшая модель: Decision Tree с точностью 0.9333

=== Classification Reports ===

--- Decision Tree ---

	precision	recall	f1-score	support
Iris-setosa	1.00	1.00	1.00	15
Iris-versicolor	1.00	0.80	0.89	15
Iris-virginica	0.83	1.00	0.91	15
accuracy			0.93	45
macro avg	0.94	0.93	0.93	45
weighted avg	0.94	0.93	0.93	45

--- Random Forest ---

	precision	recall	f1-score	support
Iris-setosa	1.00	1.00	1.00	15
Iris-versicolor	0.82	0.93	0.88	15
Iris-virginica	0.92	0.80	0.86	15
accuracy			0.91	45
macro avg	0.92	0.91	0.91	45
weighted avg	0.92	0.91	0.91	45

--- AdaBoost ---

	precision	recall	f1-score	support
Iris-setosa	1.00	1.00	1.00	15
Iris-versicolor	0.82	0.93	0.88	15

Iris-virginica	0.92	0.80	0.86	15
accuracy			0.91	45
macro avg	0.92	0.91	0.91	45
weighted avg	0.92	0.91	0.91	45

--- XGBoost ---

	precision	recall	f1-score	support
Iris-setosa	1.00	1.00	1.00	15
Iris-versicolor	0.88	0.93	0.90	15
Iris-virginica	0.93	0.87	0.90	15
accuracy			0.93	45
macro avg	0.93	0.93	0.93	45
weighted avg	0.93	0.93	0.93	45

--- CatBoost ---

	precision	recall	f1-score	support
Iris-setosa	1.00	1.00	1.00	15
Iris-versicolor	0.78	0.93	0.85	15
Iris-virginica	0.92	0.73	0.81	15
accuracy			0.89	45
macro avg	0.90	0.89	0.89	45
weighted avg	0.90	0.89	0.89	45

На датасете Iris CatBoost показал худшую точность (0.8889).
CatBoost показал худшую точность на датасете Iris прежде всего из-за небольшой выборки и сравнительно простой структуры данных. Модель обладает высокой гибкостью и склонна переусложнять решение на малом объёме данных, что приводит к нестабильности и снижению точности. В задачах такого типа более простые модели, такие как логистическая

регрессия, SVM или Random Forest, работают устойчивее и приближаются к максимальной достижимой точности на этом датасете.

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