

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-24
Лозейко М. А.
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 from
sklearn.ensemble import RandomForestClassifier,
AdaBoostClassifier from xgboost import XGBClassifier from
catboost import CatBoostClassifier from sklearn.metrics import
classification_report,
confusion_matrix, ConfusionMatrixDisplay import matplotlib.pyplot as plt
import seaborn as sns

df = pd.read_csv("glass.csv")

X = df.drop("Type", axis=1)

from sklearn.preprocessing import LabelEncoder le
= LabelEncoder() y =
le.fit_transform(df["Type"])

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=np.unique(y))
    disp.plot(cmap="Blues", values_format='d')    plt.title(f"Confusion
Matrix: {name}")
    plt.show()

plt.figure(figsize=(35, 15))
plot_tree(results["Decision Tree"]["model"],
feature_names=X.columns, class_names=[str(c)
for c in np.unique(y)], 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():

```

```

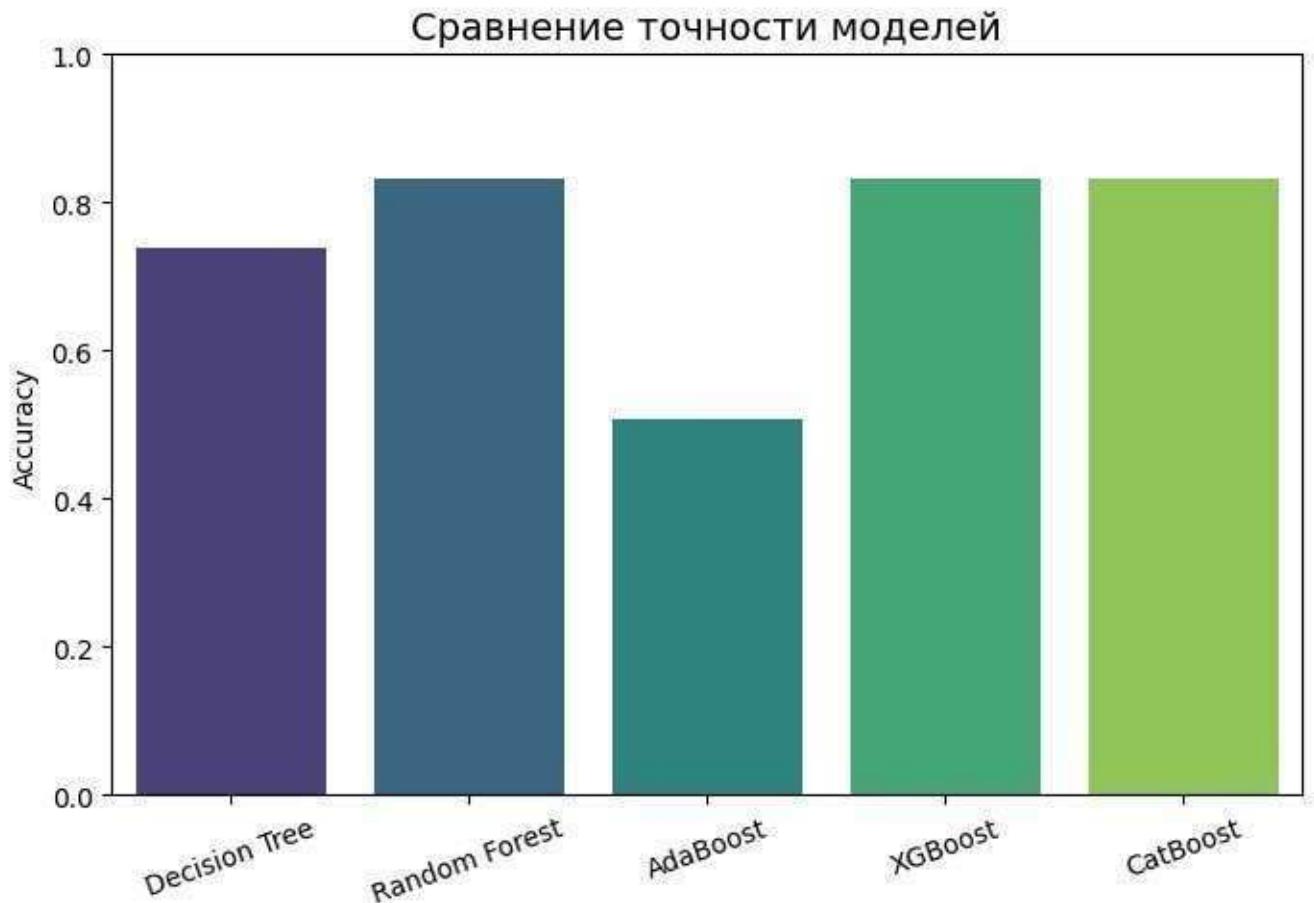
print(f" {name:15s} — Accuracy: {res['acc']:.4f} ")

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}")

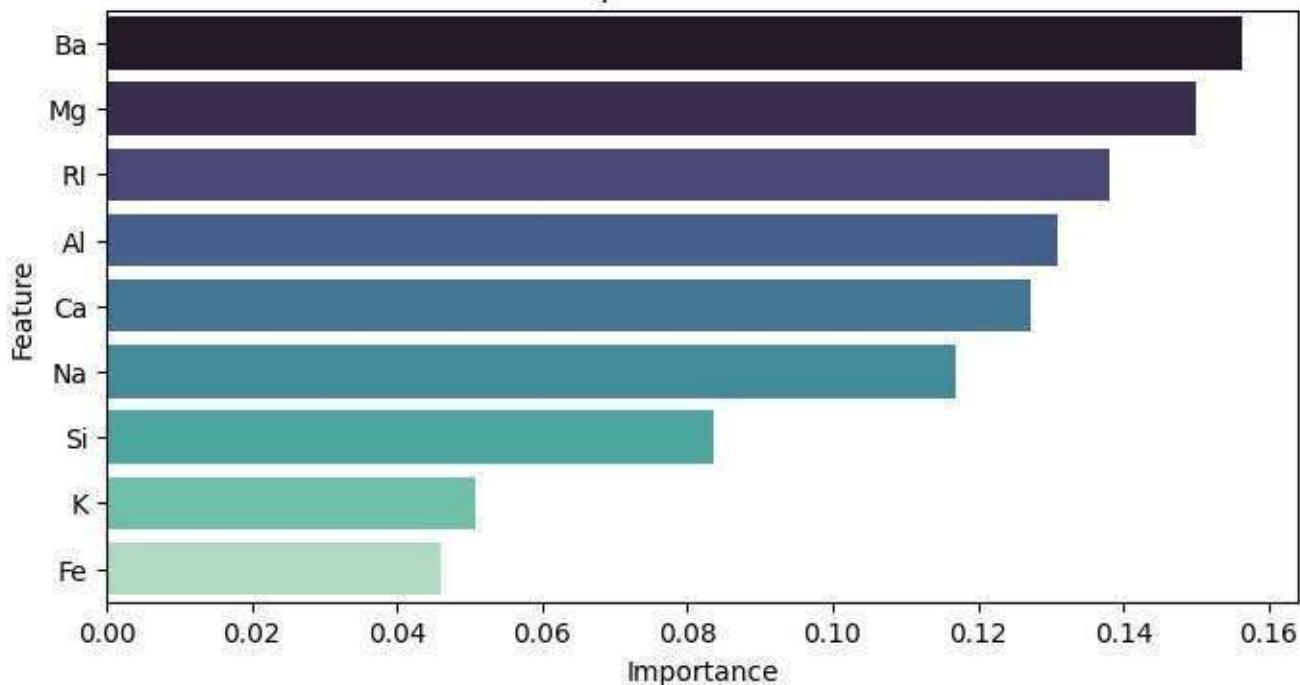
from sklearn.metrics import classification_report

print("\n--- Classification Reports ---") for
name, res in results.items():
    print(f"\n-- {name} --")    print(classification_report(y_test, res["pred"],
target_names=[str(c) for c in np.unique(y)]))

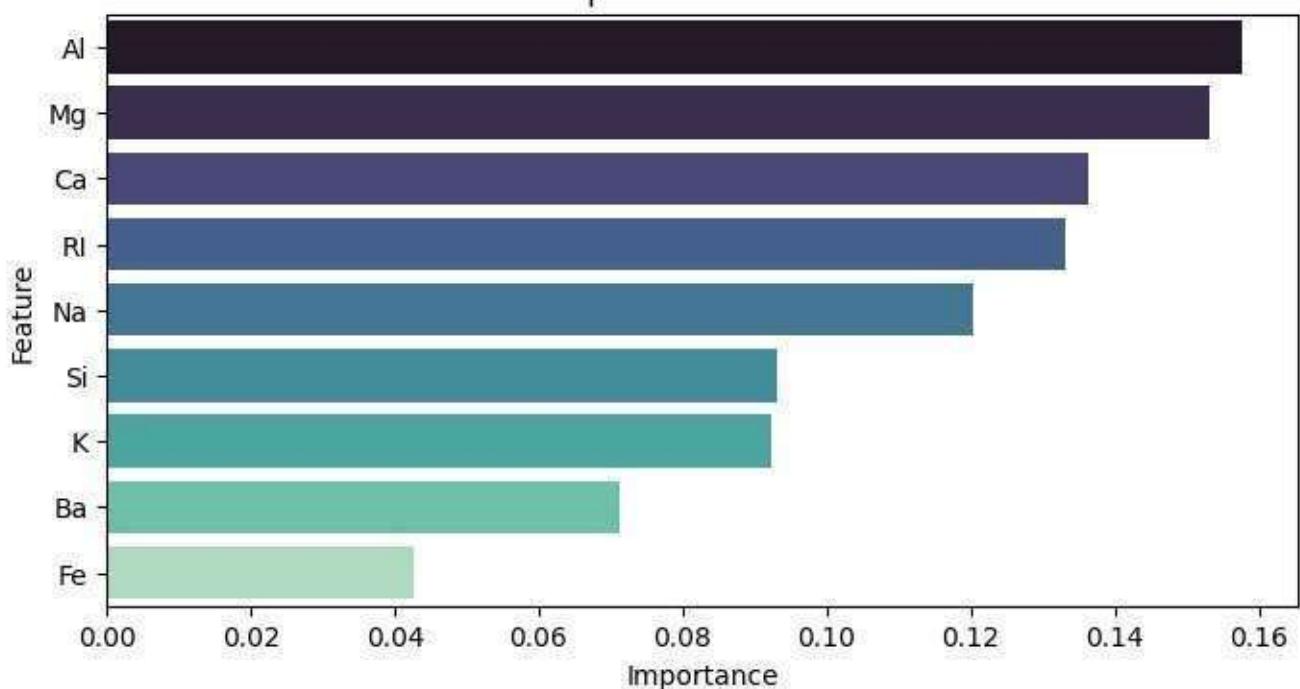
```



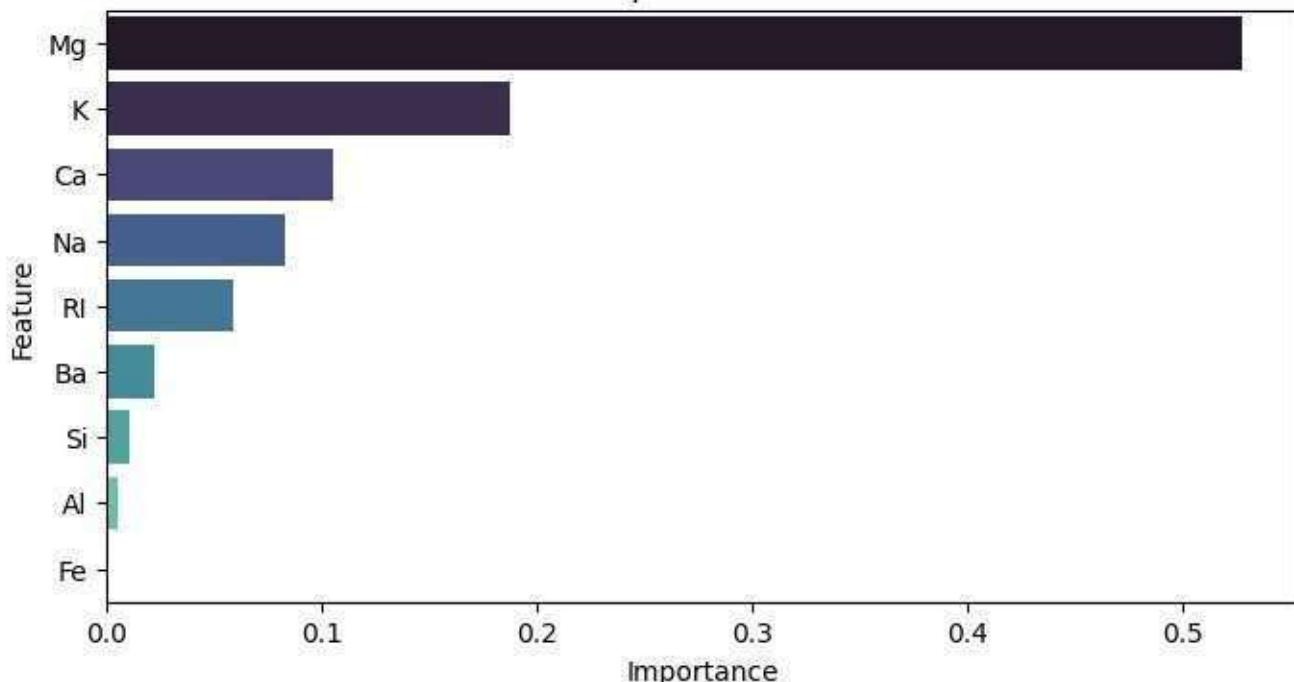
Feature Importance: Decision Tree



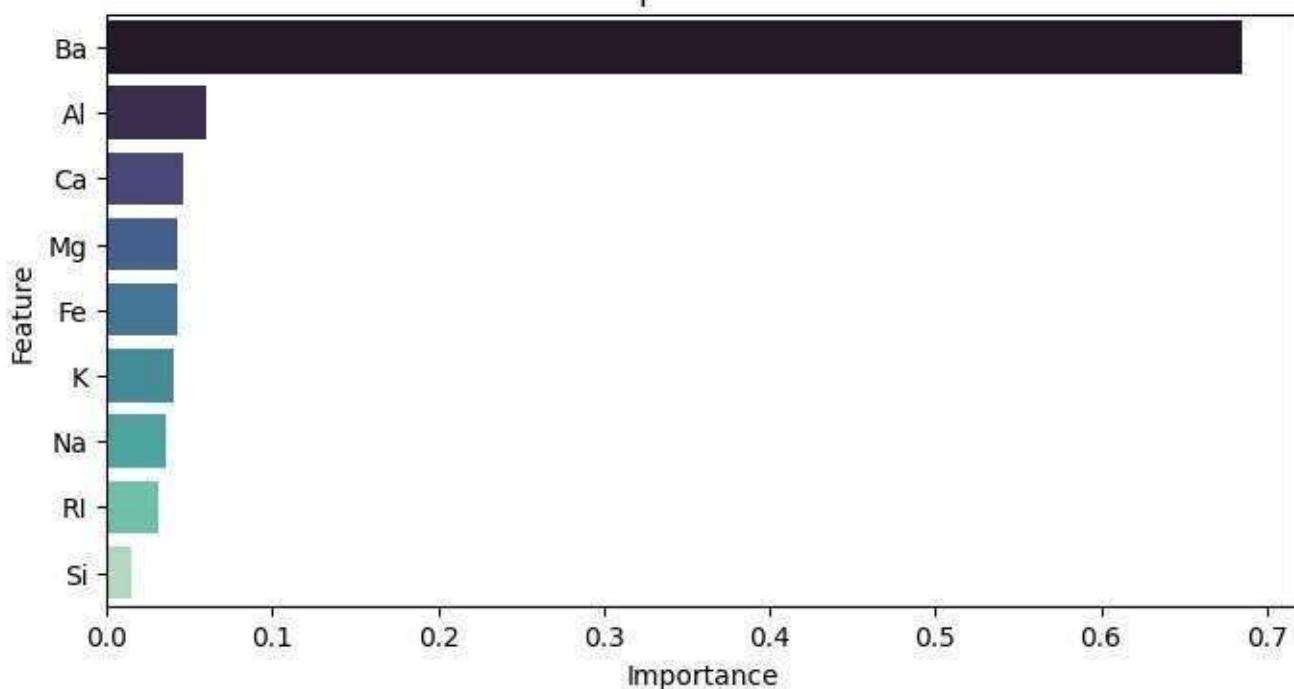
Feature Importance: Random Forest

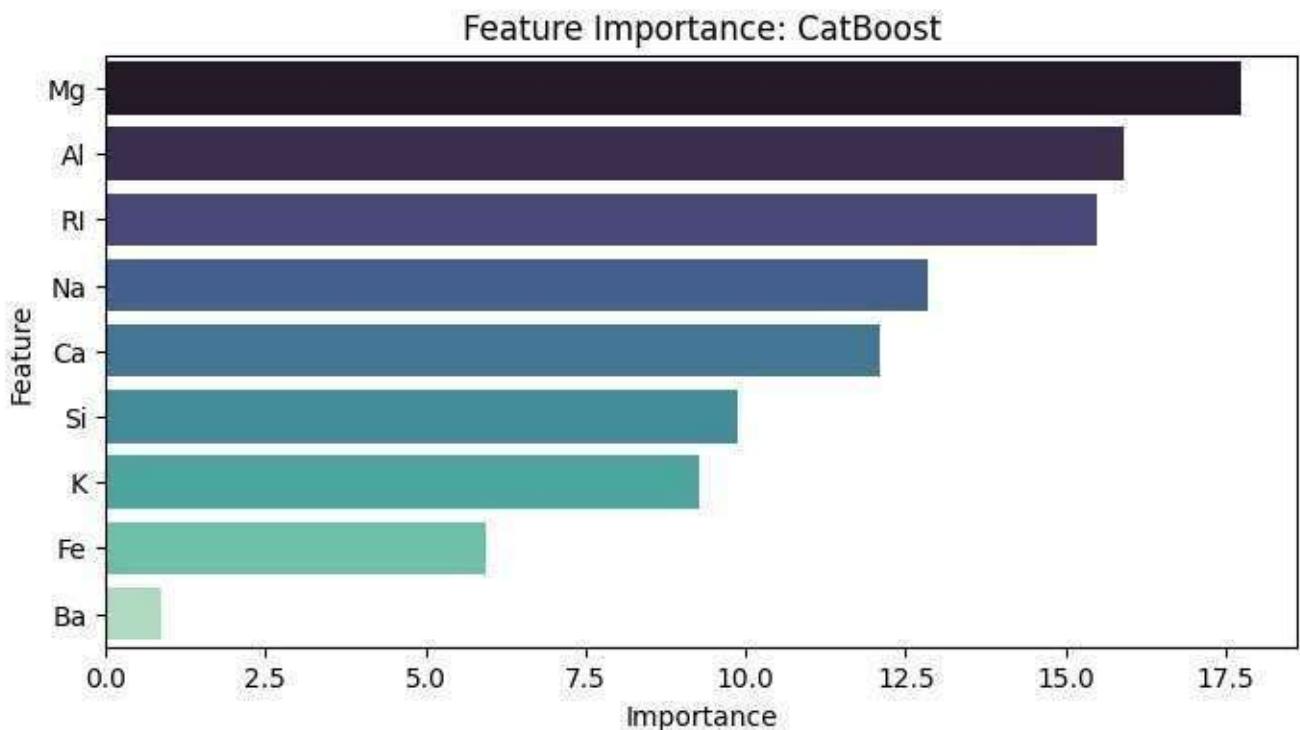


Feature Importance: AdaBoost



Feature Importance: XGBoost





Сравнение точности:

Decision Tree — Accuracy: 0.7385

Random Forest — Accuracy: 0.8308

AdaBoost — Accuracy: 0.5077

XGBoost — Accuracy: 0.8308

CatBoost — Accuracy: 0.8308

Лучшая модель: Random Forest с точностью 0.8308

==== Classification Reports ===

--- Decision Tree ---

	precision	recall	f1-score	support
--	-----------	--------	----------	---------

0	0.78	0.86	0.82	21						accuracy	0.74
1	0.82	0.61	0.70	23							
2	0.33	0.40	0.36	5							
3	0.75	0.75	0.75	4							
4	0.50	0.67	0.57	3 5	0.82	1.00	0.90	9			
	65	macro avg	0.67	0.71	0.68	65 weighted avg	0.75	0.74	0.74		

65

--- Random Forest ---

	precision	recall	f1-score	support
--	-----------	--------	----------	---------

0	0.80	0.95	0.87	21
1	0.86	0.78	0.82	23
2	1.00	0.40	0.57	5
3	0.60	0.75	0.67	4

4	0.75	1.00	0.86	3
5	1.00	0.89	0.94	9
	accuracy		0.83	65
	macro avg	0.83	0.80	0.79 65
	weighted avg	0.85	0.83	0.83 65

--- AdaBoost ---

		precision	recall
f1-score	support		

0	0.56	0.67	0.61	21
1	0.53	0.35	0.42	23
2	0.00	0.00	0.00	5
3	0.25	0.50	0.33	4
4	0.67	0.67	0.67	3
5	0.88	0.78	0.82	9
	accuracy		0.51	65
	macro avg	0.48	0.49	0.48 65
	weighted avg	0.54	0.51	0.51 65

--- XGBoost ---

		precision	recall
f1-score	support		

0	0.83	0.90	0.86	21
1	0.94	0.74	0.83	23
2	0.50	0.40	0.44	5
3	0.67	1.00	0.80	4
4	0.75	1.00	0.86	3
5	0.90	1.00	0.95	9
	accuracy		0.83	65
	macro avg	0.76	0.84	0.79 65
	weighted avg	0.84	0.83	0.83 65

--- CatBoost ---

	precision	recall	f1-score	support
--	-----------	--------	----------	---------

0	0.83	0.95	0.89	21
1	0.86	0.78	0.82	23
2	0.75	0.60	0.67	5
3	0.75	0.75	0.75	4
4	0.67	0.67	0.67	3
5	0.89	0.89	0.89	9
	accuracy		0.83	65
	macro avg	0.79	0.77	0.78 65
	weighted avg	0.83	0.83	0.83 65

На датасете Glass Identification AdaBoost показал худшую точность (0.51).

Основная причина - многоклассовая структура и малая представленность некоторых типов стекла.

Метод усиливает ошибки редких и пересекающихся объектов, что приводит к переобучению на этих сложных примерах.

В отличие от него, Random Forest и продвинутые бустинги (XGBoost, CatBoost) усредняют ошибки или используют регуляризацию, что делает их более устойчивыми.

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