

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

**Отчет по лабораторной работе 5**

Специальность ИИ-23

**Выполнил:**

Макаревич Н.Р.

Студент группы ИИ-23

**Проверил:**

Андренко К. В.

Преподаватель-стажёр  
Кафедры ИИТ,

«\_\_\_» \_\_\_\_\_ 2025 г.

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

**Код программы:**

```
# -*- coding: utf-8 -*-
```

```
"""iad_5.ipynb
```

Automatically generated by Colab.

Original file is located at

<https://colab.research.google.com/drive/1snqO9zK8xkpLluhrLwlBCfCnpyWlrAjm>

```
"""
```

```
import pandas as pd
```

```
import numpy as np
```

```
from sklearn.model_selection import train_test_split
```

```
from sklearn.preprocessing import StandardScaler
```

```
from sklearn.metrics import f1_score, classification_report
```

```
from sklearn.tree import DecisionTreeClassifier
```

```
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier
```

```
!pip install catboost > /dev/null
```

```
from catboost import CatBoostClassifier
```

```
!pip install xgboost > /dev/null
```

```
from xgboost import XGBClassifier
```

```
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality-red.csv"
```

```
data = pd.read_csv(url, sep=";")
```

```
data["target"] = (data["quality"] >= 7).astype(int)
```

```
data["target"].value_counts()
```

```
X = data.drop(["quality", "target"], axis=1)
```

```
y = data["target"]
```

```
scaler = StandardScaler()
```

```
X_scaled = scaler.fit_transform(X)
```

```
X_train, X_test, y_train, y_test = train_test_split(  
    X_scaled, y, test_size=0.2, random_state=42, stratify=y  
)
```

```
tree = DecisionTreeClassifier(random_state=42)
```

```
tree.fit(X_train, y_train)
```

```
y_pred_tree = tree.predict(X_test)
```

```
f1_tree = f1_score(y_test, y_pred_tree)
```

```
print("F1-score Decision Tree:", f1_tree)
```

```
,  
rf = RandomForestClassifier(n_estimators=200, random_state=42)
```

```
rf.fit(X_train, y_train)
```

```
y_pred_rf = rf.predict(X_test)
```

```
f1_rf = f1_score(y_test, y_pred_rf)
```

```
print("F1-score Random Forest:", f1_rf)
```

```
ada = AdaBoostClassifier(n_estimators=200, random_state=42)
```

```
ada.fit(X_train, y_train)
```

```
y_pred_ada = ada.predict(X_test)
```

```
f1_ada = f1_score(y_test, y_pred_ada)
```

```
print("F1-score AdaBoost:", f1_ada)
```

```
cat = CatBoostClassifier(  
    iterations=300,  
    learning_rate=0.05,  
    depth=6,  
    verbose=0,  
    random_seed=42  
)
```

```
cat.fit(X_train, y_train)
```

```
,  
  
y_pred_cat = cat.predict(X_test)  
f1_cat = f1_score(y_test, y_pred_cat)
```

```
  
print("F1-score CatBoost:", f1_cat)
```

```
  
  
xgb = XGBClassifier(  
    n_estimators=300,  
    learning_rate=0.05,  
    max_depth=5,  
    eval_metric="logloss",  
    random_state=42  
)
```

```
  
xgb.fit(X_train, y_train)
```

```
  
y_pred_xgb = xgb.predict(X_test)  
f1_xgb = f1_score(y_test, y_pred_xgb)
```

```
  
print("F1-score XGBoost:", f1_xgb)
```

```
  
results = pd.DataFrame({  
    "Model": ["Decision Tree", "Random Forest", "AdaBoost", "CatBoost", "XGBoost"],  
    "F1-score": [f1_tree, f1_rf, f1_ada, f1_cat, f1_xgb]  
})
```

```
  
results.sort_values("F1-score", ascending=False)
```

## Результат:

	Model	F1-score
4	XGBoost	0.789474
1	Random Forest	0.760563
3	CatBoost	0.729730
0	Decision Tree	0.659341
2	AdaBoost	0.491803

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