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

Вышинский А. С.

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.

Задание по вариантам

Вариант 3

- `Wine Quality`
- Классифицировать вино на "хорошее" (оценка >= 7) и "обычное" (оценка < 7)
- **Задания:**
  1)Загрузите данные и создайте бинарную целевую переменную;
  2)Стандартизируйте признаки и разделите выборку;
  3)Обучить на обучающей выборке одиночное дерево, случайный лес и реализовать бустинг для решающих деревьев (AdaBoost, CatBoost, XGBoost);
  4)Сравните `F1-score` для каждой модели, так как классы могут быть несбалансированы;
  5)Определите, какой алгоритм показал наилучший баланс между точностью и полнотой.

Код:

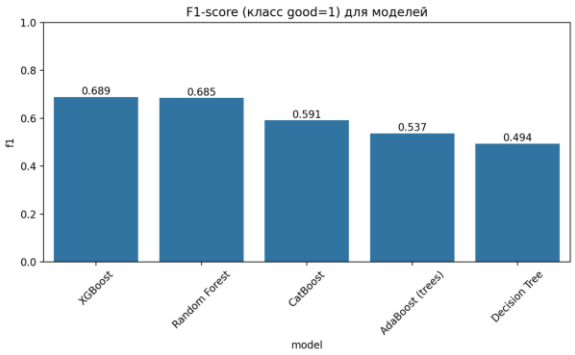
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier
from sklearn.metrics import f1_score, precision_score, recall_score,
classification_report, confusion_matrix
from xgboost import XGBClassifier
from catboost import CatBoostClassifier

import matplotlib.pyplot as plt
import seaborn as sns

import warnings
warnings.filterwarnings('ignore')


df = pd.read_csv('winequality-white.csv', sep=';')
```

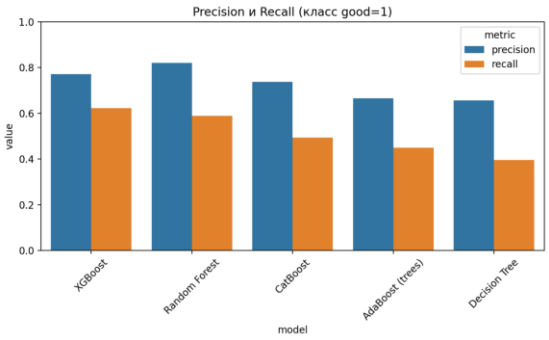
```
# Создаём бинарную целевую переменную: "good" = 1 если quality >= 7, иначе
0
df['good'] = (df['quality'] >= 7).astype(int)

X = df.drop(columns=['quality', 'good'])
y = df['good'].values

print("Размер датасета:", df.shape)
print("Распределение классов (0 = ordinary, 1 = good):")
print(pd.Series(y).value_counts())

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

X_train, X_test, y_train, y_test = train_test_split(
    X_scaled, y, test_size=0.30, random_state=42, stratify=y
)

print(f"Train: {X_train.shape[0]} samples, Test: {X_test.shape[0]}
samples")

models = {
    'Decision Tree': DecisionTreeClassifier(random_state=42, max_depth=6),
    'Random Forest': RandomForestClassifier(n_estimators=200,
random_state=42, n_jobs=-1),
    'AdaBoost (trees)': AdaBoostClassifier(
        estimator=DecisionTreeClassifier(max_depth=3, random_state=42),
        n_estimators=100, random_state=42
    ),
    'XGBoost': XGBClassifier(n_estimators=200, use_label_encoder=False,
eval_metric='logloss', random_state=42),
    'CatBoost': CatBoostClassifier(iterations=200, random_state=42,
verbose=False)
}


results = []
detailed_reports = {}

for name, model in models.items():
    print(f"\nОбучаем модель: {name} ...")
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)

    f1 = f1_score(y_test, y_pred, pos_label=1)
    precision = precision_score(y_test, y_pred, pos_label=1)
    recall = recall_score(y_test, y_pred, pos_label=1)
    report = classification_report(y_test, y_pred,
target_names=['ordinary','good'], output_dict=True)
    cm = confusion_matrix(y_test, y_pred)

    results.append({
        'model': name,
```

```
        'f1': f1,
        'precision': precision,
        'recall': recall
    })
    detailed_reports[name] = {
        'report': report,
        'confusion_matrix': cm,
        'y_pred': y_pred,
        'model_obj': model
    }

    print(f"  F1 (good=1): {f1:.4f}  Precision: {precision:.4f}  Recall:
{recall:.4f}")

results_df = pd.DataFrame(results).sort_values('f1',
ascending=False).reset_index(drop=True)
print("\nСравнение моделей по F1 (по убыванию):")
print(results_df)

best = results_df.iloc[0]
print(f"\nЛучшая модель по F1: {best['model']} (F1 = {best['f1']:.4f},
Precision = {best['precision']:.4f}, Recall = {best['recall']:.4f})")

# Покажем classification_report и матрицу ошибок для лучшей модели
best_name = best['model']
print(f"\nClassification report для лучшей модели ({best_name}):")
print(classification_report(y_test, detailed_reports[best_name]['y_pred'],
target_names=['ordinary','good']))

print("Confusion matrix (rows: true, cols: predicted):")
print(detailed_reports[best_name]['confusion_matrix'])

plt.figure(figsize=(8,5))
sns.barplot(x='model', y='f1', data=results_df)
plt.title('F1-score (класс good=1) для моделей')
plt.xticks(rotation=45)
plt.ylim(0,1)
for i, v in enumerate(results_df['f1']):
    plt.text(i, v + 0.01, f"{v:.3f}", ha='center')
plt.tight_layout()
plt.show()

plt.figure(figsize=(8,5))
df_pr = results_df.melt(id_vars=['model'],
value_vars=['precision','recall'], var_name='metric', value_name='value')
sns.barplot(x='model', y='value', hue='metric', data=df_pr)
plt.title('Precision и Recall (класс good=1)')
plt.xticks(rotation=45)
plt.ylim(0,1)
plt.tight_layout()
plt.show()
```

```
results_df.to_csv('wine_models_f1_comparison.csv', index=False)
print("\nТаблица результатов сохранена в wine_models_f1_comparison.csv")
```

Вывод:





Размер датасета: (4898, 13)
Распределение классов (0 = ordinary, 1 = good):
0 3838
1 1060
Name: count, dtype: int64
Train: 3428 samples, Test: 1470 samples

Обучаем модель: Decision Tree ...
F1 (good=1): 0.4941 Precision: 0.6562 Recall: 0.3962

Обучаем модель: Random Forest ...
F1 (good=1): 0.6850 Precision: 0.8202 Recall: 0.5881

Обучаем модель: AdaBoost (trees) ...
F1 (good=1): 0.5366 Precision: 0.6651 Recall: 0.4497

Обучаем модель: XGBoost ...
F1 (good=1): 0.6887 Precision: 0.7704 Recall: 0.6226

Обучаем модель: CatBoost ...
F1 (good=1): 0.5913 Precision: 0.7371 Recall: 0.4937

Сравнение моделей по F1 (по убыванию):

	model	f1	precision	recall
0	XGBoost	0.688696	0.770428	0.622642
1	Random Forest	0.684982	0.820175	0.588050
2	CatBoost	0.591337	0.737089	0.493711
3	AdaBoost (trees)	0.536585	0.665116	0.449686
4	Decision Tree	0.494118	0.656250	0.396226

Лучшая модель по F1: XGBoost (F1 = 0.6887, Precision = 0.7704, Recall = 0.6226)

Classification report для лучшей модели (XGBoost):

	precision	recall	f1-score	support
ordinary	0.90	0.95	0.92	1152
good	0.77	0.62	0.69	318

accuracy			0.88	1470
macro avg	0.84	0.79	0.81	1470
weighted avg	0.87	0.88	0.87	1470

Confusion matrix (rows: true, cols: predicted):
[[1093 59]
[120 198]]

Лучший результат по F1-score показал XGBoost (0.6887) — это говорит о том, что данная модель обеспечивает наилучший баланс между точностью (precision = 0.77) и полнотой (recall = 0.62) для класса *"good"*. Random Forest показал почти сопоставимый F1 (0.6850), но при этом имеет более высокую точность и немного меньшую полноту, что говорит о большей склонности к таким предсказаниям, которые говорят о меньше ложноположительных, но больше пропущенных хороших вин. CatBoost, AdaBoost и Decision Tree показали существенно более низкие F1, что свидетельствует о меньшем качестве классификации.

XGBoost продемонстрировал наилучший баланс между точностью и полнотой и, следовательно, является оптимальным алгоритмом для данной задачи классификации качества вина.

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