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
Скварнюк Д. Н.
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.

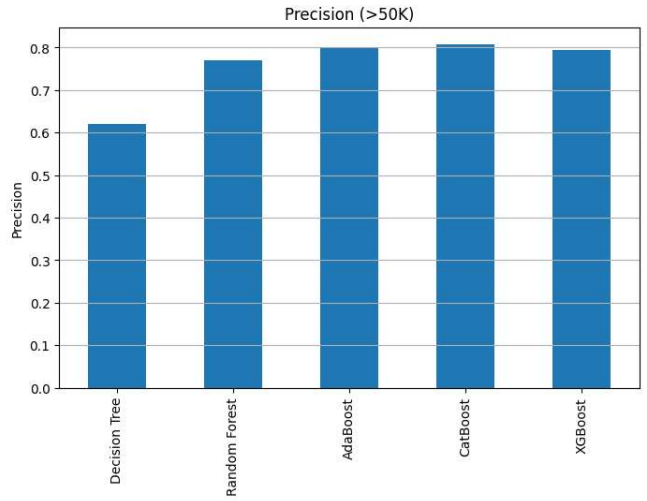
Вариант 10

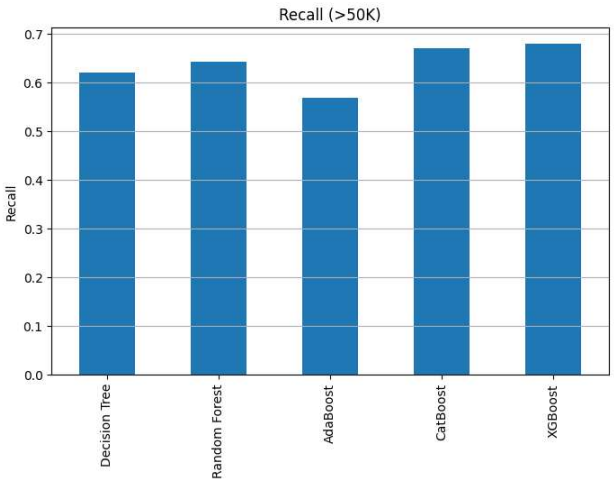
- Adult Census Income
- Предсказать, превышает ли доход человека $50 тыс. в год
- **Задания:**
    1. Загрузите данные, обработайте пропуски и категориальные признаки;
    2. Разделите данные на обучающую и тестовую выборки;
    3. Обучить на обучающей выборке одиночное дерево, случайный лес и реализовать бустинг для решающих деревьев (AdaBoost, CatBoost, XGBoost);
    4. Сравните модели по метрике precision для класса ">50K";
    5. Определите, какой алгоритм лучше всего идентифицирует людей с высоким доходом.

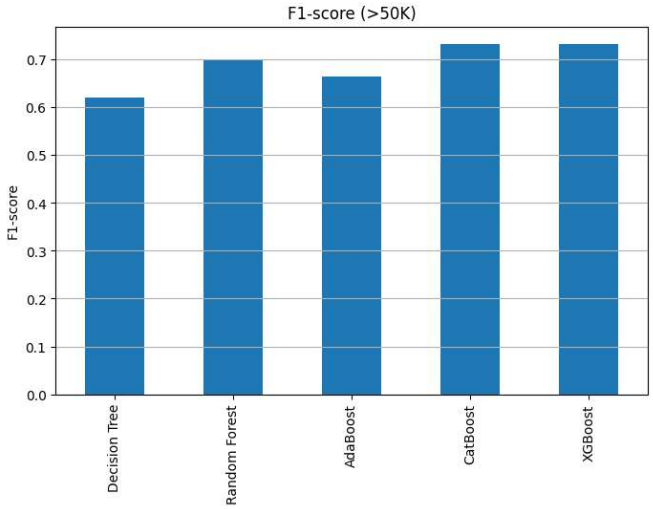
Код программы:

```
!pip install catboost xgboost
df = pd.read_csv("/content/drive/MyDrive/dataset.csv")
df.head()
df = df.replace("?", np.nan)
df = df.dropna()
target = "income"
X = df.drop(columns=[target])
y = df[target]
label_encoders = {}
for col in X.columns:
    if X[col].dtype == "object":
        le = LabelEncoder()
        X[col] = le.fit_transform(X[col])
        label_encoders[col] = le
y = LabelEncoder().fit_transform(y)

X.head()

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

model_tree = DecisionTreeClassifier(random_state=42)
model_tree.fit(X_train, y_train)
pred_tree = model_tree.predict(X_test)
```

```
precision_tree = precision_score(y_test, pred_tree, pos_label=1)
precision_tree

model_rf = RandomForestClassifier(
    n_estimators=200,
    max_depth=None,
    random_state=42
)
model_rf.fit(X_train, y_train)
pred_rf = model_rf.predict(X_test)
precision_rf = precision_score(y_test, pred_rf, pos_label=1)
precision_rf

model_ada = AdaBoostClassifier(
    n_estimators=200,
    learning_rate=0.5,
    random_state=42
)
model_ada.fit(X_train, y_train)
pred_ada = model_ada.predict(X_test)
precision_ada = precision_score(y_test, pred_ada, pos_label=1)
precision_ada

model_cat = CatBoostClassifier(
    iterations=300,
    depth=6,
    learning_rate=0.1,
    verbose=0,
    random_seed=42
)

model_cat.fit(X_train, y_train)
pred_cat = model_cat.predict(X_test)
precision_cat = precision_score(y_test, pred_cat, pos_label=1)
precision_cat

model_xgb = XGBClassifier(
    n_estimators=300,
    max_depth=6,
    learning_rate=0.1,
    subsample=0.8,
    colsample_bytree=0.8,
    random_state=42,
    objective="binary:logistic",
    eval_metric="logloss"
)

model_xgb.fit(X_train, y_train)
pred_xgb = model_xgb.predict(X_test)
precision_xgb = precision_score(y_test, pred_xgb, pos_label=1)
precision_xgb
```







Вывод:

По результатам сравнения видно, что одиночное дерево решений показывает самые низкие метрики и значительно уступает ансамблевым методам. Случайный лес заметно повышает качество за счёт использования множества деревьев, однако его recall остаётся на среднем уровне, что говорит о пропуске части объектов класса ">50K".

Лучшие результаты достигают бустинговые модели — **CatBoost и XGBoost**, обеспечивая одновременно высокий precision (~0.79–0.81) и лучший recall (~0.67–0.68). Их F1-мера также максимальна среди всех алгоритмов. Это делает бустинг наиболее эффективным подходом для задачи прогнозирования дохода, поскольку он лучше всего распознаёт пользователей с высоким доходом и делает это наиболее сбалансированно.