Московский государственный технический университет им. Н. Э. Баумана

Курс «Технологии машинного обучения»

Отчёт по лабораторной работе №5

Выполнил:
Флоринский В. А.
группа ИУ5-64Б

Проверил:
Гапанюк Ю.Е.

Дата: 07.04.25

Подпись:

Дата:

Подпись:

Москва, 2025 г.

Ансамбли моделей машинного обучения. Часть 1.

Цель лабораторной работы: изучение ансамблей моделей машинного обучения.

Требования к отчету:

Отчет по лабораторной работе должен содержать:

1. **титульный лист;**
2. **описание задания;**
3. **текст программы;**
4. **экранные формы с примерами выполнения программы.**

Задание:

1. **Выберите набор данных (датасет) для решения задачи классификации или регресии.**
2. **В случае необходимости проведите удаление или заполнение пропусков и кодирование категориальных признаков.**
3. **С использованием метода train_test_split разделите выборку на обучающую и тестовую.**
4. **Обучите следующие ансамблевые модели:**
    - **две модели группы бэггинга (бэггинг или случайный лес или сверхслучайные деревья);**
    - **AdaBoost;**
    - **градиентный бустинг.**
5. **Оцените качество моделей с помощью одной из подходящих для задачи метрик. Сравните качество полученных моделей.**

Ход выполнения:

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, f1_score
```

[1] Python

```python
df = pd.read_csv("personalized_learning_dataset.csv")

print(df.head())
```

[4] Python

```
  Student_ID  Age  Gender Education_Level       Course_Name  \
0     S00001   15  Female     High School  Machine Learning
1     S00002   49    Male   Undergraduate     Python Basics
2     S00003   20  Female   Undergraduate     Python Basics
3     S00004   37  Female   Undergraduate      Data Science
4     S00005   34  Female    Postgraduate     Python Basics

   Time_Spent_on_Videos  Quiz_Attempts  Quiz_Scores  Forum_Participation  \
0                   171              4           67                    2
1                   156              4           64                    0
2                   217              2           55                    2
3                   489              1           65                   43
4                   496              3           59                   34

   Assignment_Completion_Rate Engagement_Level  Final_Exam_Score  \
0                          89           Medium                51
1                          94           Medium                92
2                          67           Medium                45
3                          60             High                59
4                          88           Medium                93
```

```
    Learning_Style  Feedback_Score Dropout_Likelihood
0          Visual               1                 No
1  Reading/Writing              5                 No
2  Reading/Writing              1                 No
3          Visual               4                 No
4          Visual               3                 No
```

```
[5]
print(df.info())
```

Python

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 15 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   Student_ID                 10000 non-null  object
 1   Age                        10000 non-null  int64
 2   Gender                     10000 non-null  object
 3   Education_Level            10000 non-null  object
 4   Course_Name                10000 non-null  object
 5   Time_Spent_on_Videos       10000 non-null  int64
 6   Quiz_Attempts              10000 non-null  int64
 7   Quiz_Scores                10000 non-null  int64
 8   Forum_Participation        10000 non-null  int64
 9   Assignment_Completion_Rate 10000 non-null  int64
 10  Engagement_Level           10000 non-null  object
 11  Final_Exam_Score           10000 non-null  int64
 12  Learning_Style             10000 non-null  object
 13  Feedback_Score             10000 non-null  int64
 14  Dropout_Likelihood         10000 non-null  object
dtypes: int64(8), object(7)
memory usage: 1.1+ MB
None
```

```python
# Удалим идентификатор и название курса (неинформативны)
df.drop(columns=["Student_ID", "Course_Name"], inplace=True)

# Кодируем целевую переменную: Yes -> 1, No -> 0
df["Dropout_Likelihood"] = df["Dropout_Likelihood"].map({"Yes": 1, "No": 0})

# Кодирование категориальных признаков
df = pd.get_dummies(df, drop_first=True)

# Делим на признаки и целевой столбец
X = df.drop("Dropout_Likelihood", axis=1)
y = df["Dropout_Likelihood"]

# Делим выборку
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)
```

[6] Python

```python
from sklearn.ensemble import (
    BaggingClassifier,
    RandomForestClassifier,
    ExtraTreesClassifier,
    AdaBoostClassifier,
    GradientBoostingClassifier
)
```

[7] Python

```python
# Бэггинг
bagging = BaggingClassifier(random_state=42)
bagging.fit(X_train, y_train)

# Случайный лес
rf = RandomForestClassifier(random_state=42)
rf.fit(X_train, y_train)

# Сверхслучайные деревья (если хочешь — альтернатива)
et = ExtraTreesClassifier(random_state=42)
et.fit(X_train, y_train)

# AdaBoost
ada = AdaBoostClassifier(random_state=42)
ada.fit(X_train, y_train)

# Градиентный бустинг
gb = GradientBoostingClassifier(random_state=42)
gb.fit(X_train, y_train)
```

Python

```
GradientBoostingClassifier(random_state=42)
```

```python
models = {
    "Bagging": bagging,
    "Random Forest": rf,
    "Extra rees": et,
    "AdaBoost": ada,
    "Gradient Boosting": gb
}
```

```python
print("📊 Сравнение моделей:\n")
for name, model in models.items():
    y_pred = model.predict(X_test)
    acc = accuracy_score(y_test, y_pred)
    f1 = f1_score(y_test, y_pred)
    print(f"{name}: Accuracy = {acc:.4f}, F1 Score = {f1:.4f}")
```

9] Python

```
📊 Сравнение моделей:

Bagging: Accuracy = 0.7995, F1 Score = 0.0475
Random Forest: Accuracy = 0.8115, F1 Score = 0.0000
Extra rees: Accuracy = 0.8105, F1 Score = 0.0052
AdaBoost: Accuracy = 0.8115, F1 Score = 0.0000
Gradient Boosting: Accuracy = 0.8115, F1 Score = 0.0053
```

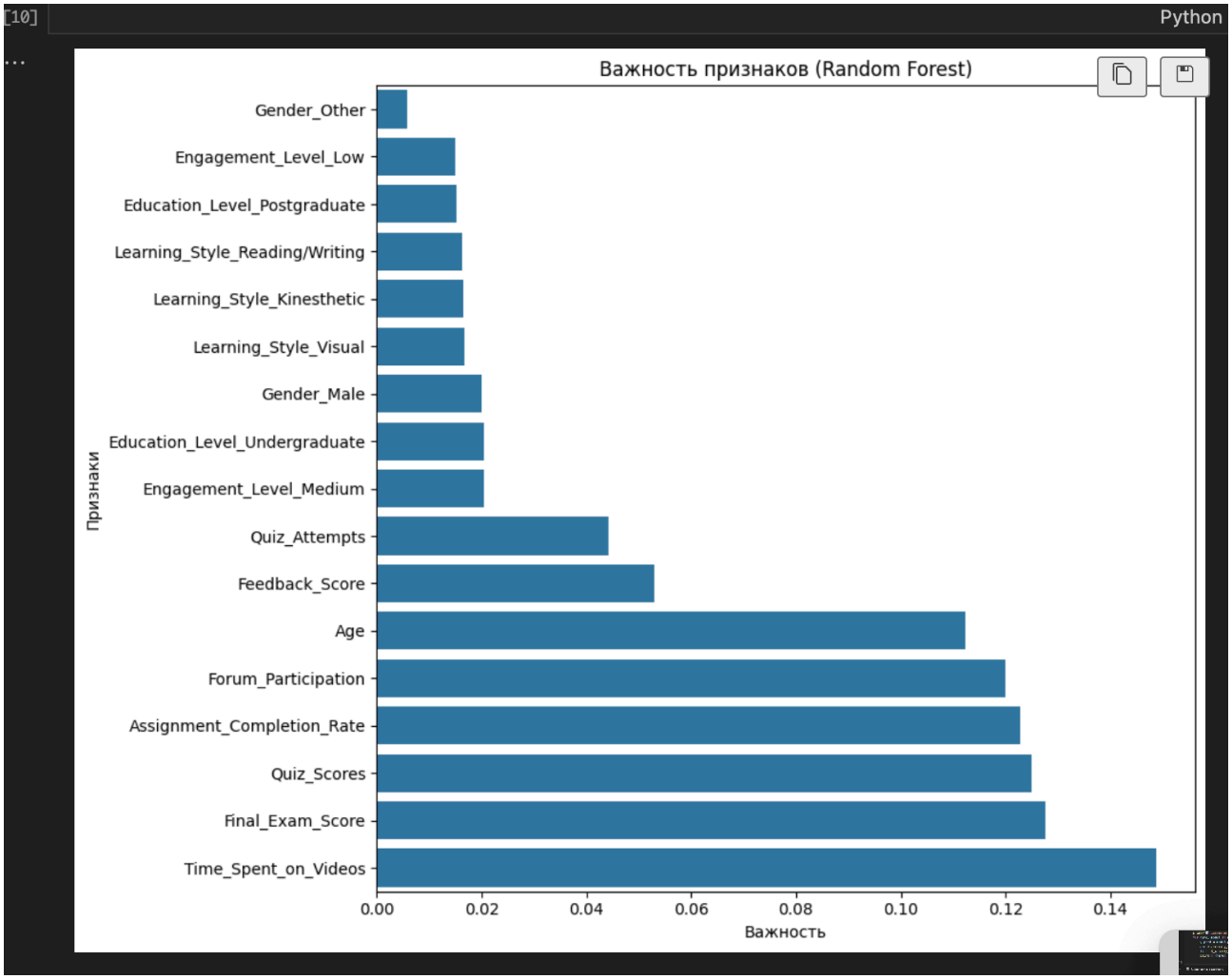
```python
import matplotlib.pyplot as plt
import seaborn as sns

# Получаем важность признаков из модели случайного леса
feature_importances = pd.Series(rf.feature_importances_, index=X.columns)
feature_importances = feature_importances.sort_values(ascending=True)

# Построим график
plt.figure(figsize=(10, 8))
sns.barplot(x=feature_importances.values, y=feature_importances.index)
plt.title("Важность признаков (Random Forest)")
plt.xlabel("Важность")
plt.ylabel("Признаки")
plt.tight_layout()
plt.show()
```

10] Python

[10] Python

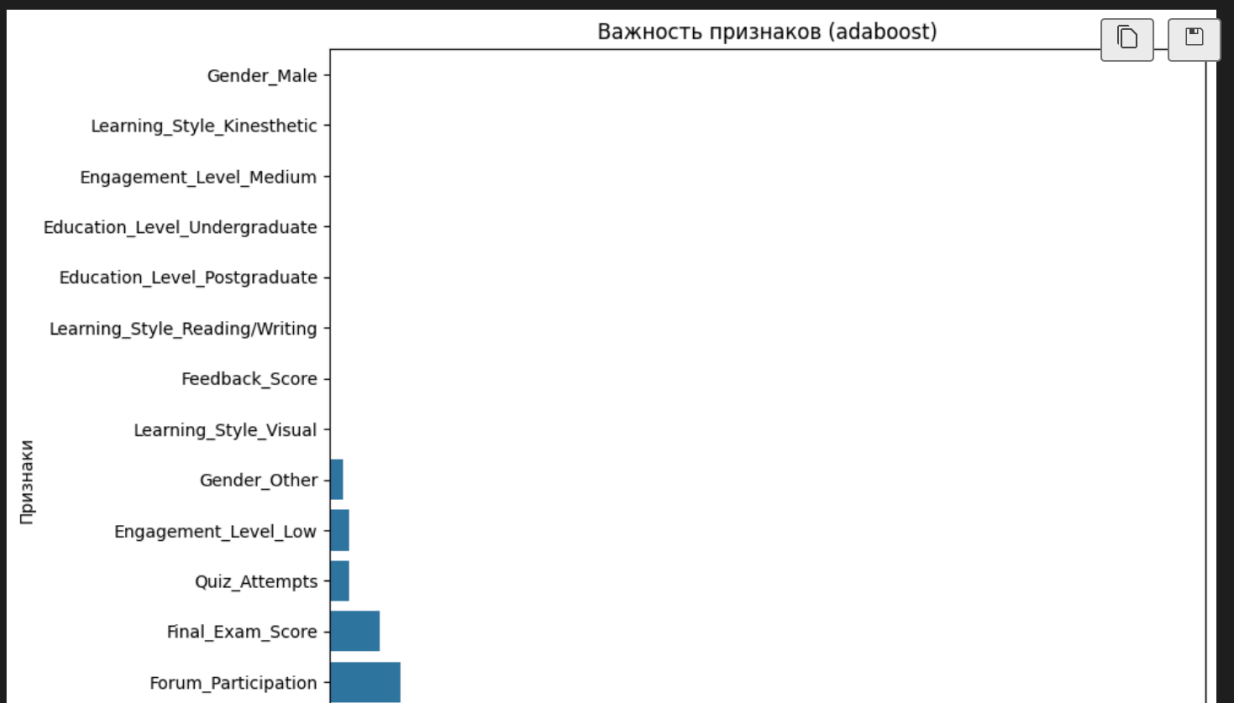

Важность признаков (Random Forest)
Gender_Other
Engagement_Level_Low
Education_Level_Postgraduate
Learning_Style_Reading/Writing
Learning_Style_Kinesthetic
Learning_Style_Visual
Gender_Male
Education_Level_Undergraduate
Engagement_Level_Medium
Quiz_Attempts
Feedback_Score
Age
Forum_Participation
Assignment_Completion_Rate
Quiz_Scores
Final_Exam_Score
Time_Spent_on_Videos
Признаки
0.00
0.02
0.04
0.06
0.08
0.10
0.12
0.14
Важность

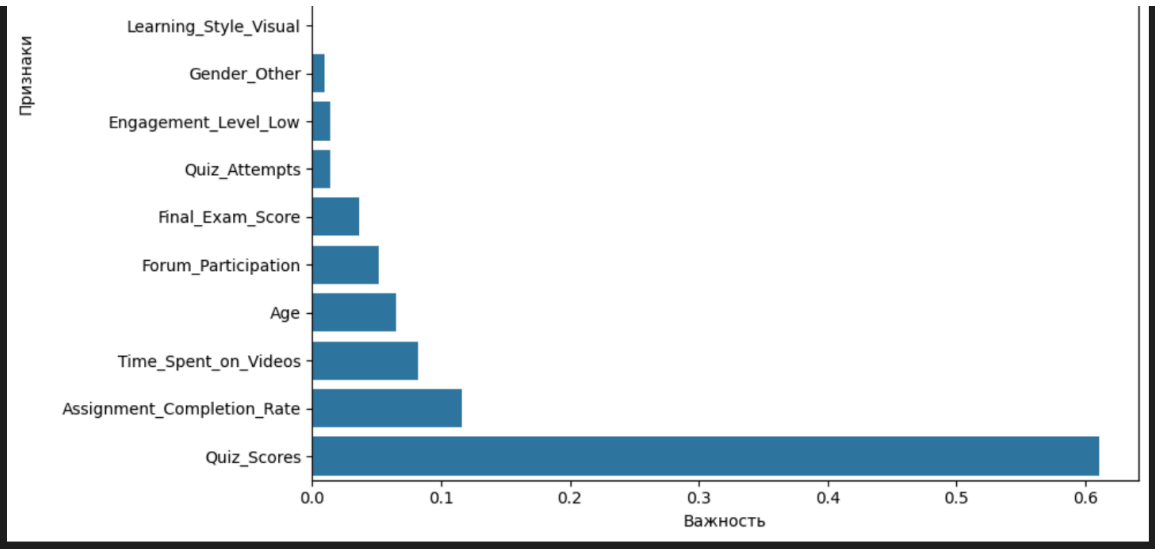
```python
# Получаем важность признаков из модели adaboost
feature_importances = pd.Series(ada.feature_importances_, index=X.columns)
feature_importances = feature_importances.sort_values(ascending=True)

# Построим график
plt.figure(figsize=(10, 8))
sns.barplot(x=feature_importances.values, y=feature_importances.index)
plt.title("Важность признаков (adaboost)")
plt.xlabel("Важность")
plt.ylabel("Признаки")
plt.tight_layout()
plt.show()
```

1]

Python



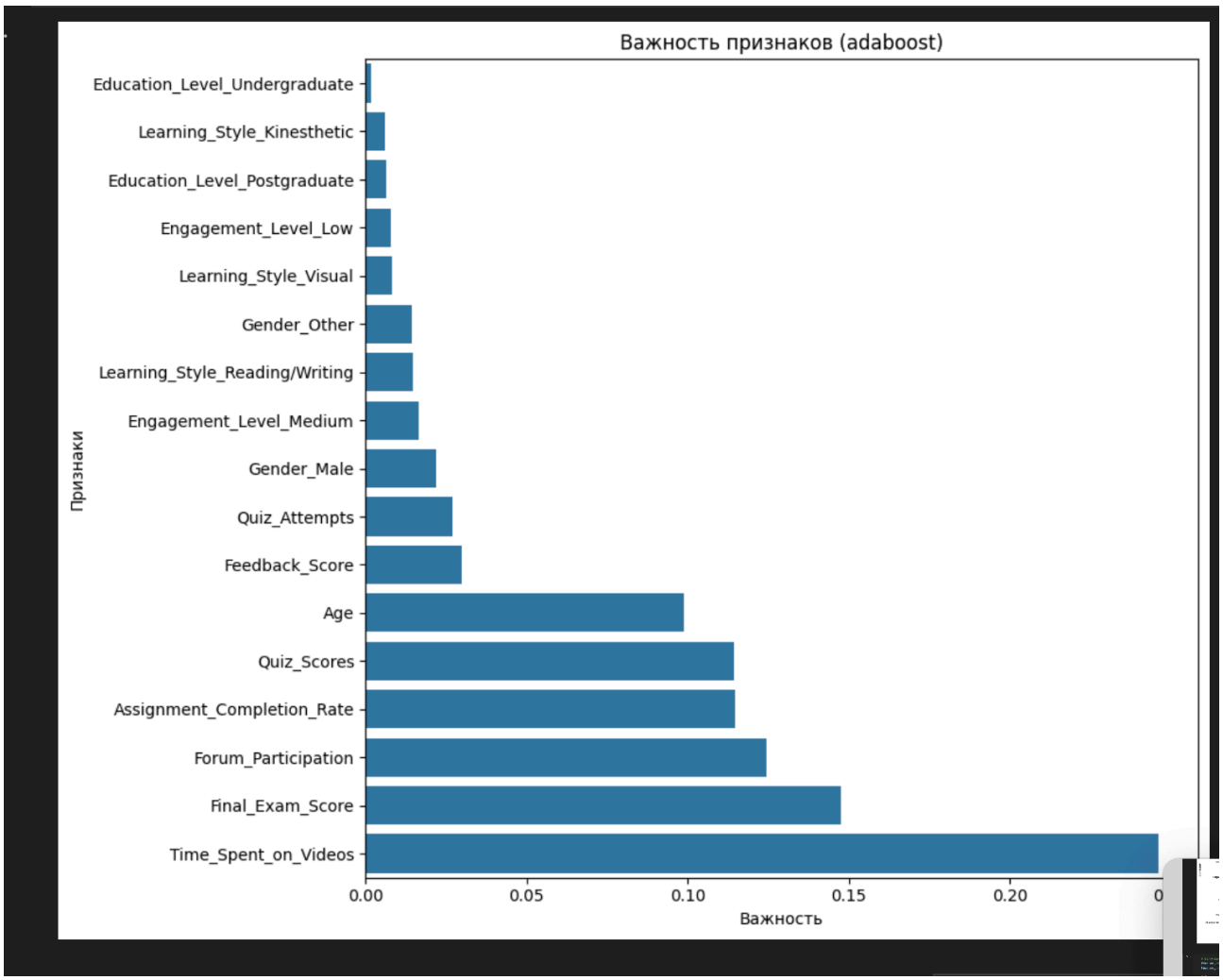


+ Code + Markdown

```python
# Получаем важность признаков из модели gb
feature_importances = pd.Series(gb.feature_importances_, index=X.columns)
feature_importances = feature_importances.sort_values(ascending=True)

# Построим график
plt.figure(figsize=(10, 8))
sns.barplot(x=feature_importances.values, y=feature_importances.index)
plt.title("Важность признаков (gb)")
plt.xlabel("Важность")
plt.ylabel("Признаки")
plt.tight_layout()
plt.show()
```

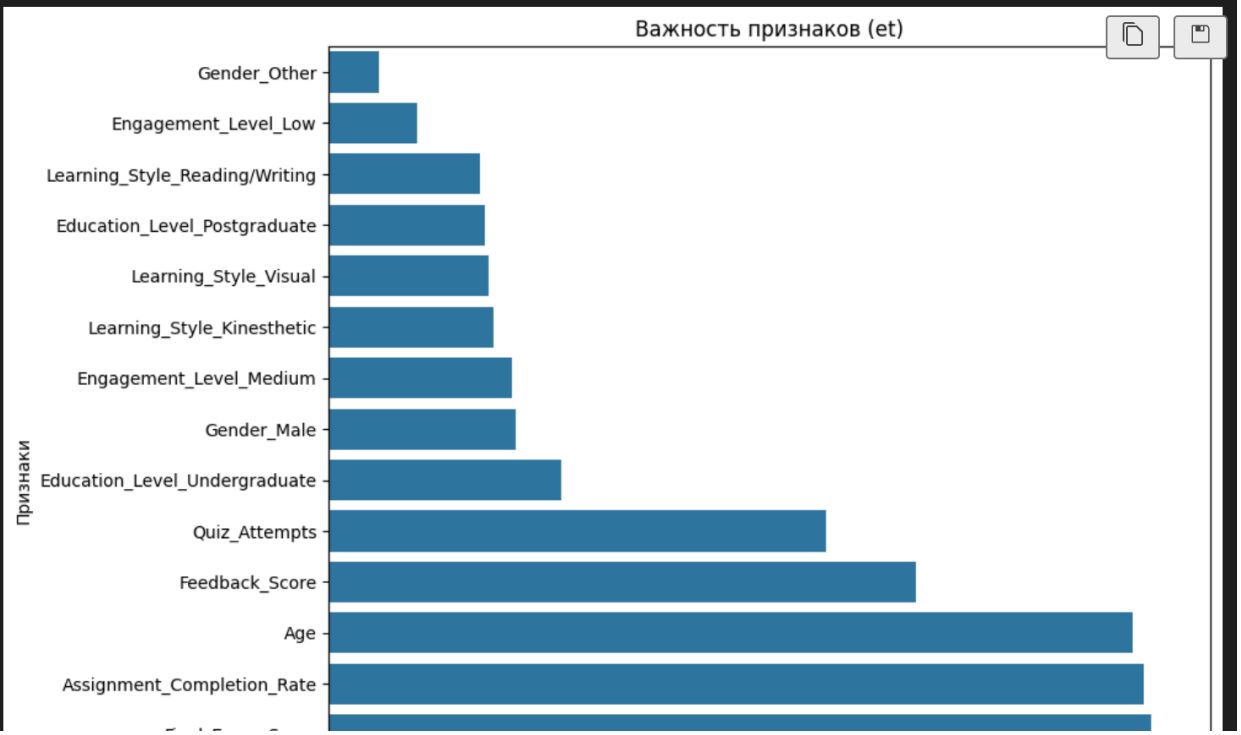
[ ]

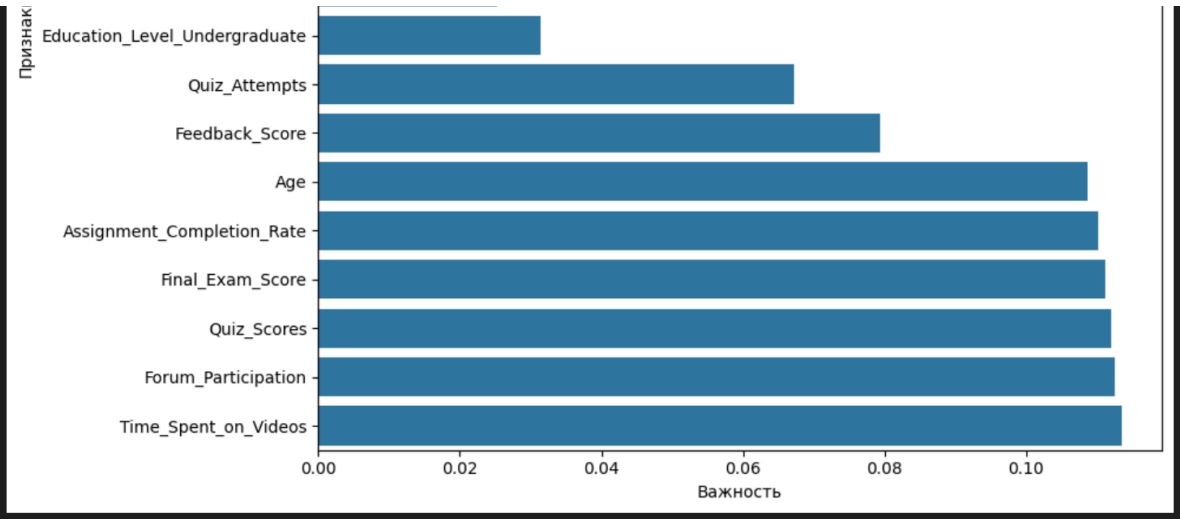

Важность признаков (adaboost)
Education_Level_Undergraduate
Learning_Style_Kinesthetic
Education_Level_Postgraduate
Engagement_Level_Low
Learning_Style_Visual
Gender_Other
Learning_Style_Reading/Writing
Engagement_Level_Medium
Gender_Male
Quiz_Attempts
Feedback_Score
Age
Quiz_Scores
Assignment_Completion_Rate
Forum_Participation
Final_Exam_Score
Time_Spent_on_Videos
Признаки
0.00
0.05
0.10
0.15
0.20
Важность

```python
# Получаем важность признаков из модели adaboost
feature_importances = pd.Series(et.feature_importances_, index=X.columns)
feature_importances = feature_importances.sort_values(ascending=True)

# Построим график
plt.figure(figsize=(10, 8))
sns.barplot(x=feature_importances.values, y=feature_importances.index)
plt.title("Важность признаков (et)")
plt.xlabel("Важность")
plt.ylabel("Признаки")
plt.tight_layout()
plt.show()
```

Python






```python
# Получаем важность признаков из модели adaboost
feature_importances = pd.Series(bagging.feature_importances_, index=X.columns)
feature_importances = feature_importances.sort_values(ascending=True)

# Построим график
plt.figure(figsize=(10, 8))
sns.barplot(x=feature_importances.values, y=feature_importances.index)
plt.title("Важность признаков (bagging)")
plt.xlabel("Важность")
plt.ylabel("Признаки")
plt.tight_layout()
plt.show()
```

[14]

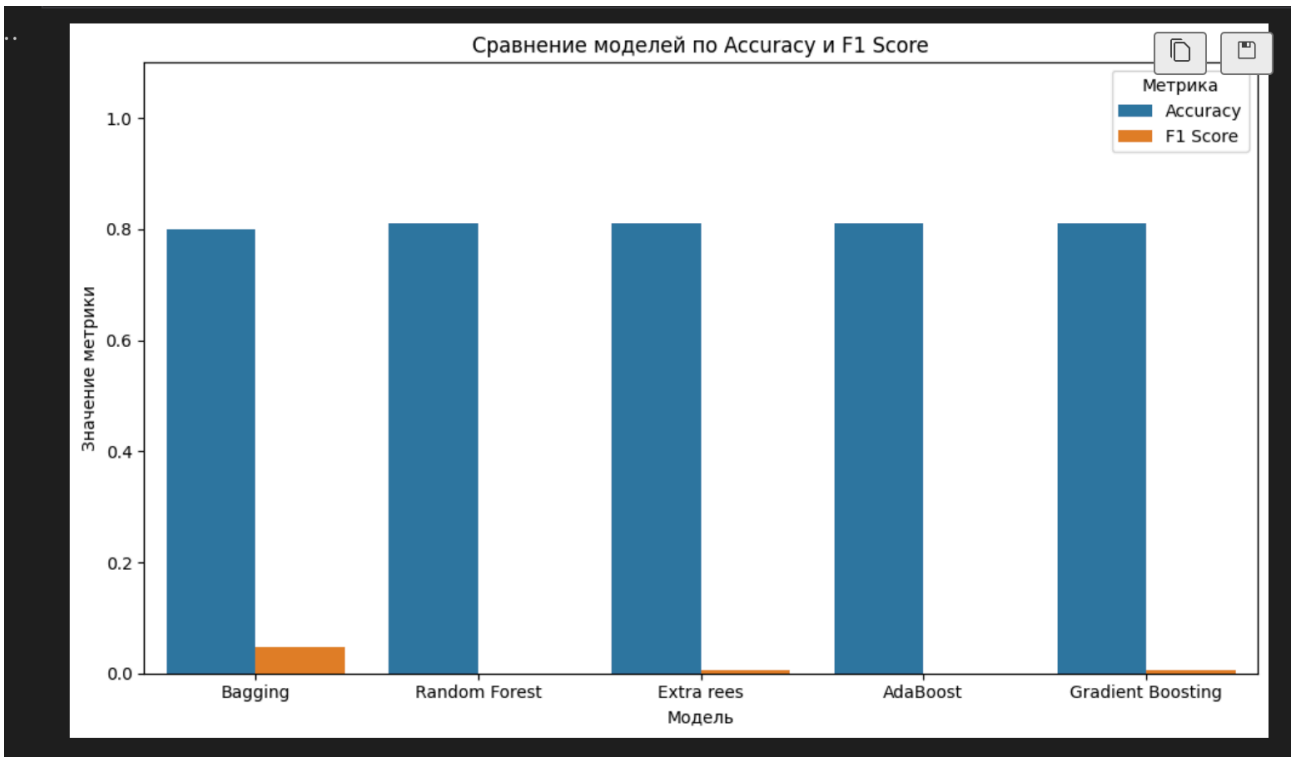
```python
# Сохраним метрики в датафрейм
results = []

for name, model in models.items():
    y_pred = model.predict(X_test)
    results.append({
        "Model": name,
        "Accuracy": accuracy_score(y_test, y_pred),
        "F1 Score": f1_score(y_test, y_pred)
    })

results_df = pd.DataFrame(results)

# Визуализация
plt.figure(figsize=(10, 6))
sns.barplot(data=results_df.melt(id_vars="Model"), x="Model", y="value", hue="variable")
plt.title("Сравнение моделей по Accuracy и F1 Score")
plt.ylabel("Значение метрики")
plt.xlabel("Модель")
plt.ylim(0, 1.1)
plt.legend(title="Метрика")
plt.tight_layout()
plt.show()
```

[15] Python


Сравнение моделей по Accuracy и F1 Score
Метрика
Accuracy
F1 Score
Значение метрики
1.0
0.8
0.6
0.4
0.2
0.0
Bagging
Random Forest
Extra rees
AdaBoost
Gradient Boosting
Модель