**Московский государственный технический университет им. Н. Э. Баумана**

Курс «Технологии машинного обучения»

Отчёт по лабораторной работе №5

Выполнил:
Дувакин А.В.
группа ИУ5-63Б

Проверил:
Гапанюк Ю.Е.

Дата: 01.03.25

Подпись:

Дата:

Подпись:

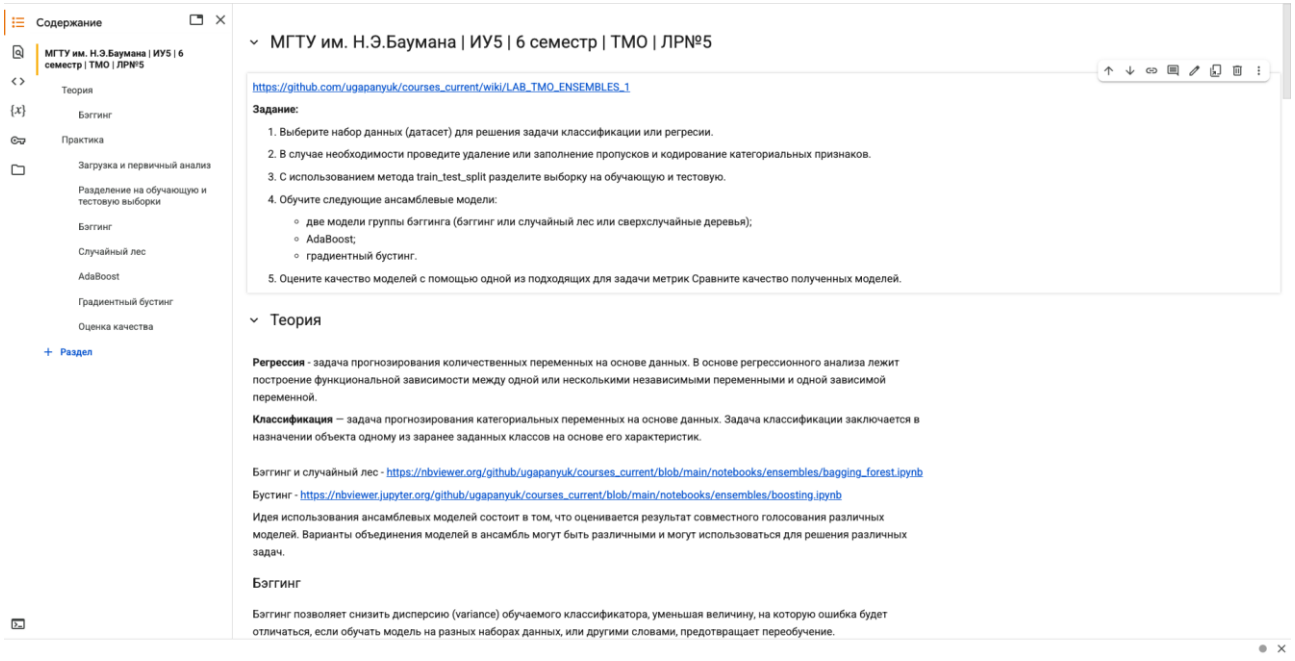
Москва, 2025 г.

**Цель лабораторной работы:** изучение ансамблей моделей машинного обучения.

**Задание:**

1. Выберите набор данных (датасет) для решения задачи классификации или регрессии.
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:
    a. две модели группы бэггинга (бэггинг или случайный лес или сверхслучайные деревья);
    b. AdaBoost;
    c. градиентный бустинг.
5. Оцените качество моделей

**Ход выполнения:**



## Практика

Датасет: https://github.com/ongaunjie1/credit-score-prediction

### Загрузка и первичный анализ

```
[ ] import numpy as np
    import pandas as pd
    import seaborn as sns
    import matplotlib.pyplot as plt
    %matplotlib inline
    sns.set(style="ticks")
```

```
[ ] df0 = pd.read_csv("/dataset.csv")
    df0.info()
    df0.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14790 entries, 0 to 14789
Data columns (total 12 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   age                       14790 non-null  int64
 1   annual_income             14790 non-null  float64
 2   num_bank_acc              14790 non-null  int64
 3   num_credit_card           14790 non-null  int64
 4   interest_rate             14790 non-null  int64
 5   delay_from_due_date       14790 non-null  int64
 6   outstanding_debt          14790 non-null  float64
 7   credit_history_age        14790 non-null  float64
 8   installment_per_month     14790 non-null  float64
 9   monthly_balance           14790 non-null  float64
 10  payment_of_min_amount_Yes 14790 non-null  bool
 11  Predicted Credit Score    14790 non-null  int64
dtypes: bool(1), float64(5), int64(6)
memory usage: 1.3 MB
```

| | age | annual_income | num_bank_acc | num_credit_card | interest_rate | delay_from_due_date | outstanding_debt | credit_history_age | installment_per_month | monthly_balance | payment_of_m |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 23 | 19114.12 | 3 | 4 | 3 | 3 | 809.98 | 22.90 | 49.57 | 186.27 | |
| 1 | 24 | 19114.12 | 3 | 4 | 3 | 3 | 809.98 | 22.10 | 49.57 | 361.44 | |
| 2 | 28 | 34847.84 | 2 | 4 | 6 | 3 | 605.03 | 27.40 | 18.82 | 303.36 | |
| 3 | 28 | 34847.84 | 2 | 4 | 6 | 3 | 605.03 | 27.50 | 18.82 | 452.30 | |

Содержание
МГТУ им. Н.Э.Баумана | ИУ5 | 6 семестр | ТМО | ЛРК№5
Теория
Бэггинг
Практика
Загрузка и первичный анализ
Разделение на обучающую и тестовую выборки
Бэггинг
Случайный лес
AdaBoost
Градиентный бустинг
Оценка качества
+ Раздел

```
(11832, 11)
(2958, 11)
(11832,)
(2958,)
```

### Бэггинг

```
[ ] from sklearn.ensemble import BaggingClassifier


    bagging_model = BaggingClassifier(random_state=1)
    bagging_model.fit(xTrain, yTrain)
    y_pred_bagging = bagging_model.predict(xTest)
```

### Случайный лес

```
[ ] from sklearn.ensemble import RandomForestClassifier


    rf_model = RandomForestClassifier(random_state=1)
    rf_model.fit(xTrain, yTrain)
    y_pred_rf = rf_model.predict(xTest)
```

### AdaBoost

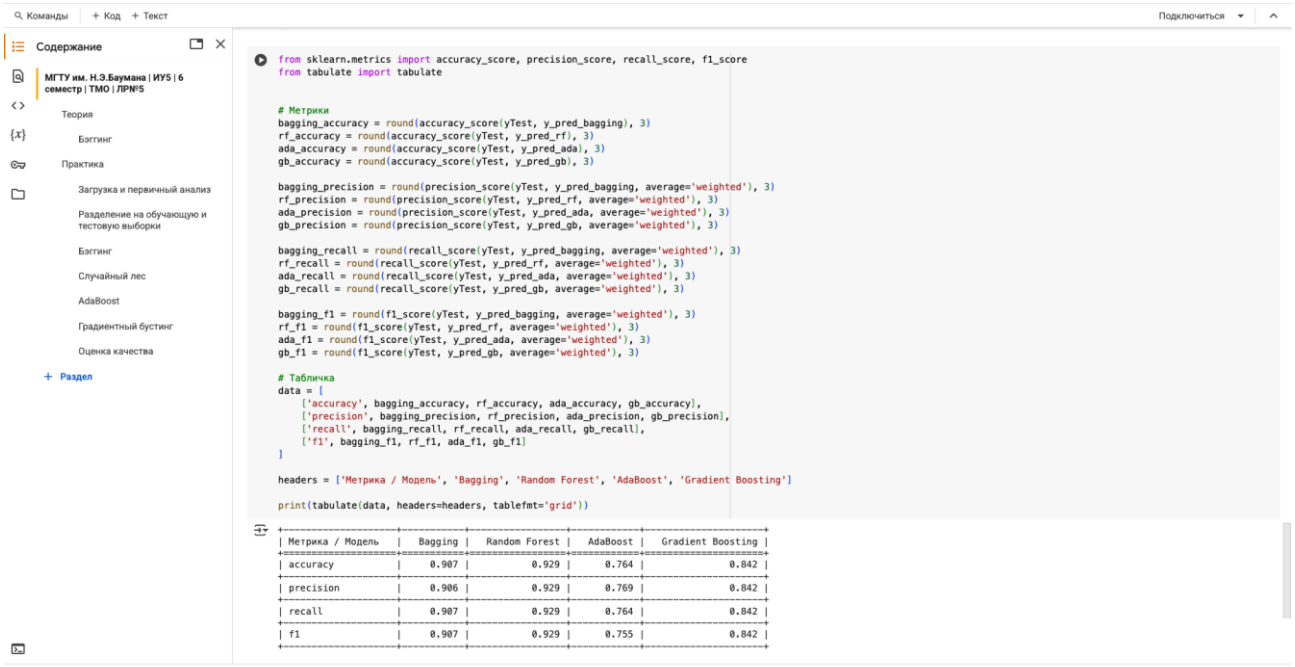
```
[ ] from sklearn.ensemble import AdaBoostClassifier


    ada_model = AdaBoostClassifier(random_state=1)
    ada_model.fit(xTrain, yTrain)
    y_pred_ada = ada_model.predict(xTest)
```

### Градиентный бустинг

```
[ ] from sklearn.ensemble import GradientBoostingClassifier


    gb_model = GradientBoostingClassifier(random_state=1)
    gb_model.fit(xTrain, yTrain)
    y_pred_gb = gb_model.predict(xTest)
```


Команды
+ Код
+ Текст
Подключиться
Содержание
МГТУ им. Н.Э.Баумана | ИУ5 | 6 семестр | ТМО | ЛР№5
Теория
Бэггинг
Практика
Загрузка и первичный анализ
Разделение на обучающую и тестовую выборки
Бэггинг
Случайный лес
AdaBoost
Градиентный бустинг
Оценка качества
+ Раздел
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
from tabulate import tabulate
# Метрики
bagging_accuracy = round(accuracy_score(yTest, y_pred_bagging), 3)
rf_accuracy = round(accuracy_score(yTest, y_pred_rf), 3)
ada_accuracy = round(accuracy_score(yTest, y_pred_ada), 3)
gb_accuracy = round(accuracy_score(yTest, y_pred_gb), 3)
bagging_precision = round(precision_score(yTest, y_pred_bagging, average='weighted'), 3)
rf_precision = round(precision_score(yTest, y_pred_rf, average='weighted'), 3)
ada_precision = round(precision_score(yTest, y_pred_ada, average='weighted'), 3)
gb_precision = round(precision_score(yTest, y_pred_gb, average='weighted'), 3)
bagging_recall = round(recall_score(yTest, y_pred_bagging, average='weighted'), 3)
rf_recall = round(recall_score(yTest, y_pred_rf, average='weighted'), 3)
ada_recall = round(recall_score(yTest, y_pred_ada, average='weighted'), 3)
gb_recall = round(recall_score(yTest, y_pred_gb, average='weighted'), 3)
bagging_f1 = round(f1_score(yTest, y_pred_bagging, average='weighted'), 3)
rf_f1 = round(f1_score(yTest, y_pred_rf, average='weighted'), 3)
ada_f1 = round(f1_score(yTest, y_pred_ada, average='weighted'), 3)
gb_f1 = round(f1_score(yTest, y_pred_gb, average='weighted'), 3)
# Табличка
data = [
['accuracy', bagging_accuracy, rf_accuracy, ada_accuracy, gb_accuracy],
['precision', bagging_precision, rf_precision, ada_precision, gb_precision],
['recall', bagging_recall, rf_recall, ada_recall, gb_recall],
['f1', bagging_f1, rf_f1, ada_f1, gb_f1]
]
headers = ['Метрика / Модель', 'Bagging', 'Random Forest', 'AdaBoost', 'Gradient Boosting']
print(tabulate(data, headers=headers, tablefmt='grid'))
+--------------------+-----------+---------------+------------+-------------------+
| Метрика / Модель | Bagging | Random Forest | AdaBoost | Gradient Boosting |
+====================+===========+===============+============+===================+
| accuracy | 0.907 | 0.929 | 0.764 | 0.842 |
+--------------------+-----------+---------------+------------+-------------------+
| precision | 0.906 | 0.929 | 0.769 | 0.842 |
+--------------------+-----------+---------------+------------+-------------------+
| recall | 0.907 | 0.929 | 0.764 | 0.842 |
+--------------------+-----------+---------------+------------+-------------------+
| f1 | 0.907 | 0.929 | 0.755 | 0.842 |
+--------------------+-----------+---------------+------------+-------------------+