Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования «Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)» (МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ Информатика и системы управления

КАФЕДРА Системы обработки информации и управления

Рубежный контроль №2

По курсу «Технологии машинного обучения»

Вариант 20

Подготовила:

Студентка группы ИУ5-64Б

Тахтамышева А.А.

Проверил:

Преподаватель кафедры ИУ5

Гапанюк Ю.Е.

Москва, 2022 г.

Тема: Методы построения моделей машинного обучения.

ИУ5-64Б, ИУ5Ц-84Б	Линейная/логистическая регрессия	Градиентный бустинг

Задание. Для заданного набора данных (по Вашему варианту) постройте модели классификации или регрессии (в зависимости от конкретной задачи, рассматриваемой в наборе данных). Для построения моделей используйте методы 1 и 2 (по варианту для Вашей группы). Оцените качество моделей на основе подходящих метрик качества (не менее двух метрик). Какие метрики качества Вы использовали и почему? Какие выводы Вы можете сделать о качестве построенных моделей? Для построения моделей необходимо выполнить требуемую предобработку данных: заполнение пропусков, кодирование категориальных признаков, и т.д.

Листинг программы:

```
import numpy as np
import pandas as pd
from typing import Dict, Tuple
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error, mean_squared_error,
mean_squared_log_error, median_absolute_error, r2_score
from sklearn.linear_model import LinearRegression
from pandas import DataFrame
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import LabelEncoder
from xgboost import XGBRegressor
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
sns.set(style="ticks")
#%%
data = pd.read_csv('impeachment-polls.csv')
data.head()

Оцени количество пропущенных значений в каждом поле.

data.isnull().sum()
data.shape

Посмотрим типы данных в исходном датасете.

data.dtypes

Выполним предобработку данных.

Посмотрим в каких полях (числовой тип данных) есть пропущенные
значения, а также посмотрим на их количество.

total_count = data.shape[0]
```

```
num_cols = []
for col in data.columns:
    # Количество пустых значений
    temp_null_count = data[data[col].isnull()].shape[0]
    dt = str(data[col].dtype)
    if temp_null_count>0 and (dt=='float64' or dt=='int64'):
        num_cols.append(col)
        temp_perc = round((temp_null_count / total_count) * 100.0, 2)
        print('Колонка {}. Тип данных {}. Количество пустых значений
{}, {}%.'.format(col, dt, temp_null_count, temp_perc))

data_num = data[num_cols]
data_num
```

Построим гистограмму по признакам.

```
for col in data_num:
    plt.hist(data[col], 50)
    plt.xlabel(col)
    plt.show()
```

Заполним пустые значения разными стратегиями.

```
imp_mean = SimpleImputer(missing_values=np.nan, strategy='mean')
for column in ["Rep Yes", "Rep No", "Ind Yes", "Ind No"]:
    imp_mean.fit(data[[column]])
    data[column] = imp_mean.transform(data[[column]])

imp_median = SimpleImputer(missing_values=np.nan, strategy='median')
for column in ["Unsure", "Dem Yes", "Dem No", "Ind Sample"]:
    imp_median.fit(data[[column]])
    data[column] = imp_median.transform(data[[column]])

imp_most_frequent = SimpleImputer(missing_values=np.nan,
strategy='most_frequent')
for column in ["Rep Sample", "Dem Sample"]:
    imp_most_frequent.fit(data[[column]])
    data[column] = imp_most_frequent.transform(data[[column]])
```

Посмотрим в каких категориальных признаках есть пропущенные значения, а также посмотрим на их количество.

```
cat_cols = []
for col in data.columns:
    # Количество пустых значений
    temp_null_count = data[data[col].isnull()].shape[0]
    dt = str(data[col].dtype)
    if temp_null_count>0 and (dt=='object'):
        cat_cols.append(col)
        temp_perc = round((temp_null_count / total_count) * 100.0, 2)
        print('Колонка {}. Тип данных {}. Количество пустых значений
{}, {}%.'.format(col, dt, temp_null_count, temp_perc))
```

Удалим столбцы, содержание большое количество пропущенных значений.

```
data = data.drop(columns=["Notes", "tracking"], axis=1)
```

Заполним категориальные признаки.

```
imp = SimpleImputer(missing_values=np.nan, strategy='most_frequent')
for column in ["Text", "Category", "URL", "Sponsor"]:
    data[column] = imp.fit_transform(data[[column]])
```

Проверим, что все значения заполнены.

```
data.isnull().sum()
```

Закодируем категориальные признаки.

```
LE = LabelEncoder()
for column in ["Start", "End", "Pollster", "Sponsor", "Pop", "Text",
"Category", "Include?", "URL"]:
    data[column] = LE.fit_transform(data[column])
```

```
data.head()
```

Построим корреляционную матрицу.

```
fig, ax = plt.subplots(figsize=(20,10))
sns.heatmap(data.corr(method="pearson"), ax=ax,annot=True, fmt=".2f",
center=0)
```

Будем решать задачу регрессии.

```
data['Yes'].unique()
```

```
data_new=data[['Yes','Pop', 'Rep Yes', 'Dem Yes', 'Ind Yes']]
```

```
data_new.head()
```

Метод №1: Линейная регрессия

```
xArray = data_new.drop("Yes", axis=1)
yArray = data_new["Yes"]
```

Разделим выборку на обучающую и тестовую.

```
trainX, testX, trainY, testY = train_test_split(xArray, yArray,
test_size=0.2, random_state=1)
```

Обучим модель.

```
model1 = LinearRegression(normalize=True)
model1.fit(trainX, trainY)
```

Предсказанные значение для тестовой выборки:

```
y_test_predict_LR = model1.predict(testX)
y_test_predict_LR
```

Посмотрим результат метрик.

```
print('MSE:', mean_squared_error(y_test_predict_LR, testY))
print('MAE:', (mean_absolute_error(testY, y_test_predict_LR)))

x_ax = range(len(testY))
plt.plot(x_ax, testY, label="истинные значения")
plt.plot(x_ax, y_test_predict_LR, label="предсказанные значения")
plt.title("Модель линейной регрессии")
plt.legend()
plt.show()
```

Метод №2: Градиентный бустинг

```
x_Array = data_new.drop("Yes", axis=1)
y_Array = data_new["Yes"]
```

Разделим выборку на обучающую и тестовую.

```
train_X, test_X, train_Y, test_Y = train_test_split(x_Array, y_Array,
test_size=0.2, random_state=1)
```

Обучим модель.

```
model2 = XGBRegressor( booster='gbtree', max_depth=4)
model2.fit(train_X, train_Y)


score = model2.score(train_X, train_Y)
print("Training score: ", score)
```

Предсказанные значения.

```
y_test_predict_XGBR = model2.predict(test_X)
y_test_predict_XGBR
```

Посмотрим результат метрик.

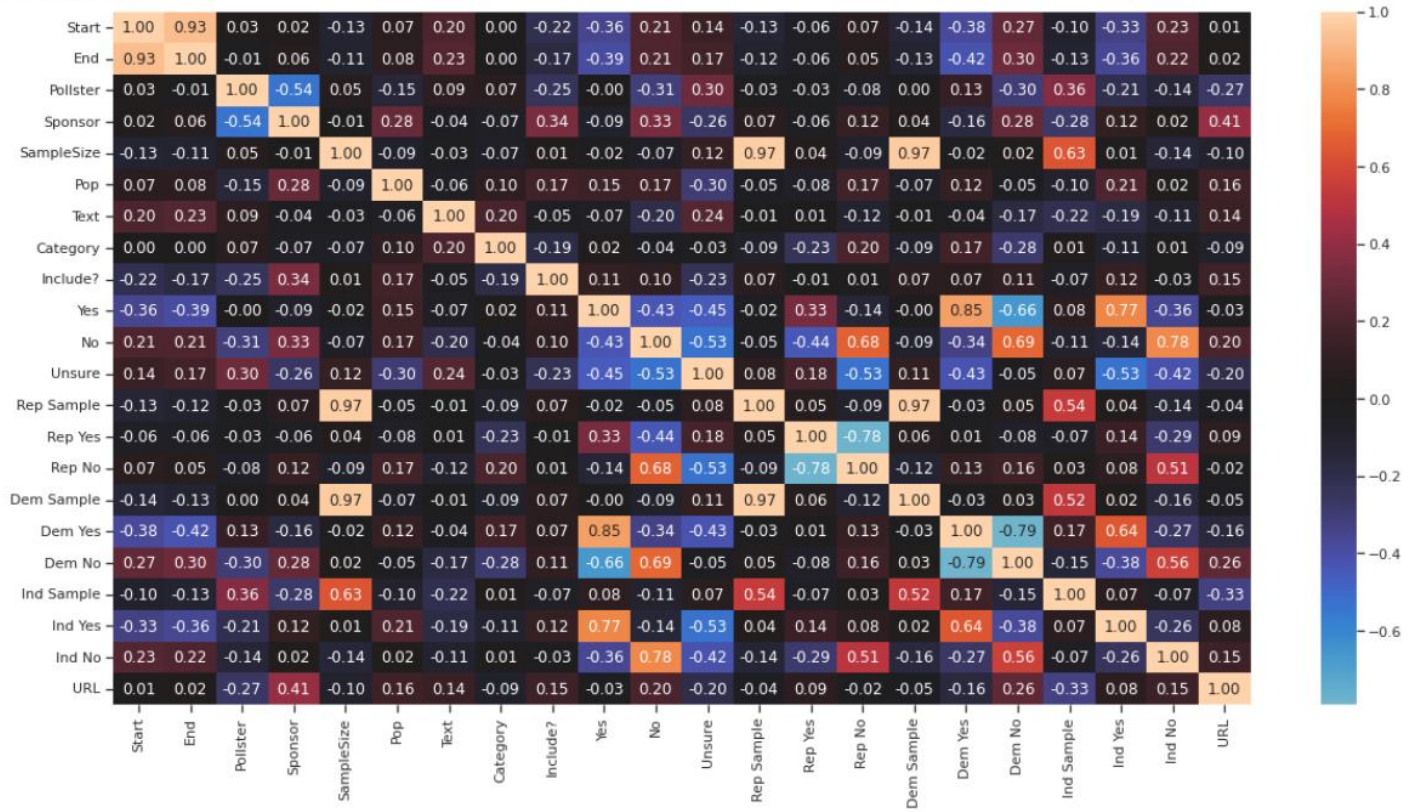
```
print('MSE:', mean_squared_error(y_test_predict_XGBR, test_Y))
print('MAE:', (mean_absolute_error(y_test_predict_XGBR, test_Y)))

x_ax = range(len(test_Y))
plt.plot(x_ax, test_Y, label="истинные значения")
plt.plot(x_ax, y_test_predict_XGBR, label="предсказанные значения")
plt.title("Модель градиентного бустинга")
plt.legend()
plt.show()
```

Экранные формы:

1. Корреляционная матрица:

```
fig, ax = plt.subplots(figsize=(20,10))
sns.heatmap(data.corr(method="pearson"), ax=ax,annot=True, fmt=".2f", center=0)
```

```
<AxesSubplot:>
```



2. График истинных и предсказанных значений (линейная регрессия):

```
▷ 0.4s
x_ax = range(len(testY))
plt.plot(x_ax, testY, label="истинные значения")
plt.plot(x_ax, y_test_predict_LR, label="предсказанные значения")
plt.title("Модель линейной регрессии")
plt.legend()
plt.show()
```

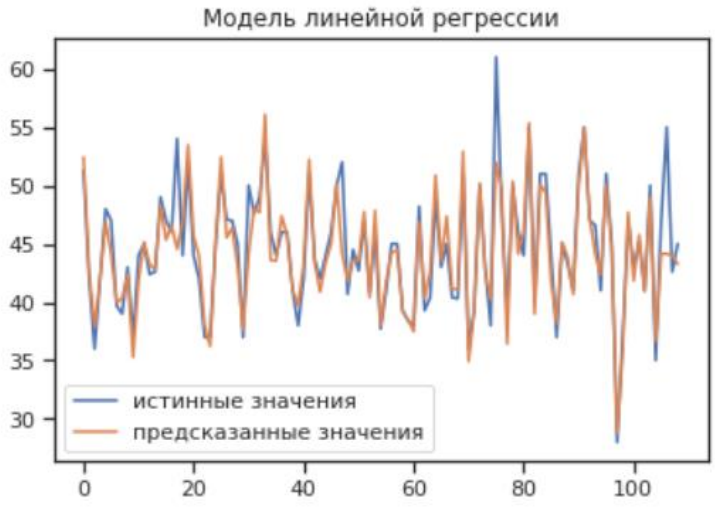
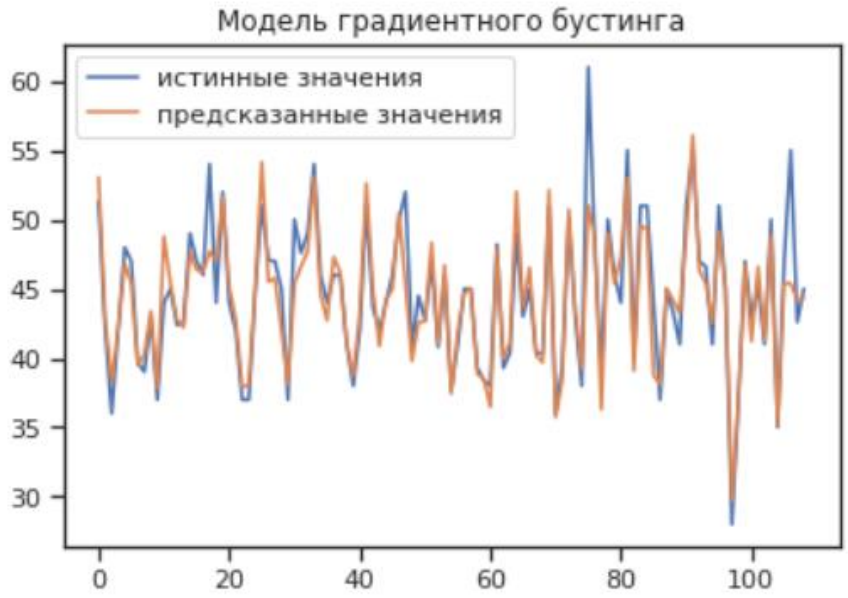


График истинных и предсказанных значений (градиентный бустинг):

```
x_ax = range(len(test_Y))
plt.plot(x_ax, test_Y, label="истинные значения")
plt.plot(x_ax, y_test_predict_XGBR, label="предсказанные значения")
plt.title("Модель градиентного бустинга")
plt.legend()
plt.show()
```



3. Метрики:

Посмотрим результат метрик. Модель линейная регрессия:

```
print('MSE:', mean_squared_error(y_test_predict_LR, testY))
print('MAE:', (mean_absolute_error(testY, y_test_predict_LR)))
```

```
MSE: 5.1305150647967235
MAE: 1.4154756725716189
```

Посмотрим результат метрик. Модель градиентный бустинг:

```
print('MSE:', mean_squared_error(y_test_predict_XGBR, test_Y))
print('MAE:', (mean_absolute_error(y_test_predict_XGBR, test_Y)))
```

```
MSE: 4.695915081992975
MAE: 1.3921362024709723
```

Вывод:

Для решения данной задачи (регрессии) были выбраны две метрики: среднеквадратичная ошибка и средняя абсолютная ошибка.

Значение метрик для каждой модели близки, но модель градиентного бустинга (библиотека XGBoost) обучилась и предсказала значения лучше, чем модель линейной регрессии.