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

Факультет «Информатика и управление»

Кафедра «Системы обработки информации и управления»

Курс «Технологии машинного обучения»

Отчет по рубежному контролю №2

Вариант 15

Выполнила:
студентка группы ИУ5-64Б
Носова Элина
Подпись и дата:

Проверил:
преподаватель каф.ИУ5
Гапанюк Ю. Е.
Подпись и дата:

Москва, 2022 г.

**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.

| **Группа** | **Метод №1** | **Метод №2** |
|---|---|---|
| ИУ5-64Б | Линейная/логистическая регрессия | Градиентный бустинг |

Выполнение:

Загрузка данных.

```
data=pd.read_csv(r'C:\Users\Элина\Desktop\Учеба\ТМО\states_all_extended.csv') # импорт датасета
data
```

| | PRIMARY_KEY | STATE | YEAR | ENROLL | TOTAL_REVENUE | FEDERAL_REVENUE | STATE_REVENUE | LOCAL_REVENUE | TOTAL_EXPENDITU |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1992_ALABAMA | ALABAMA | 1992 | NaN | 2678885.0 | 304177.0 | 1659028.0 | 715680.0 | 265379 |
| 1 | 1992_ALASKA | ALASKA | 1992 | NaN | 1049591.0 | 106780.0 | 720711.0 | 222100.0 | 97248 |
| 2 | 1992_ARIZONA | ARIZONA | 1992 | NaN | 3258079.0 | 297888.0 | 1369815.0 | 1590376.0 | 340158 |
| 3 | 1992_ARKANSAS | ARKANSAS | 1992 | NaN | 1711959.0 | 178571.0 | 958785.0 | 574603.0 | 174302 |
| 4 | 1992_CALIFORNIA | CALIFORNIA | 1992 | NaN | 26260025.0 | 2072470.0 | 16546514.0 | 7641041.0 | 2713883 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| 1710 | 2019_VIRGINIA | VIRGINIA | 2019 | NaN | NaN | NaN | NaN | NaN | N |
| 1711 | 2019_WASHINGTON | WASHINGTON | 2019 | NaN | NaN | NaN | NaN | NaN | N |
| 1712 | 2019_WEST_VIRGINIA | WEST_VIRGINIA | 2019 | NaN | NaN | NaN | NaN | NaN | N |
| 1713 | 2019_WISCONSIN | WISCONSIN | 2019 | NaN | NaN | NaN | NaN | NaN | N |
| 1714 | 2019_WYOMING | WYOMING | 2019 | NaN | NaN | NaN | NaN | NaN | N |

1715 rows × 266 columns

Проверка пропусков

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

```
PRIMARY_KEY               0
STATE                     0
YEAR                      0
ENROLL                  491
TOTAL_REVENUE           440
                       ...
G08_AM_A_MATHEMATICS   1655
G08_HP_A_READING       1701
G08_HP_A_MATHEMATICS   1702
G08_TR_A_READING       1574
G08_TR_A_MATHEMATICS   1570
Length: 266, dtype: int64
```

```
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))
```

```
Колонка G03_A_A. Тип данных float64. Количество пустых значений 83, 4.84%.
Колонка G04_A_A. Тип данных float64. Количество пустых значений 83, 4.84%.
Колонка G05_A_A. Тип данных float64. Количество пустых значений 83, 4.84%.
Колонка G06_A_A. Тип данных float64. Количество пустых значений 83, 4.84%.
Колонка G07_A_A. Тип данных float64. Количество пустых значений 83, 4.84%.
Колонка G08_A_A. Тип данных float64. Количество пустых значений 83, 4.84%.
Колонка G09_A_A. Тип данных float64. Количество пустых значений 83, 4.84%.
Колонка G10_A_A. Тип данных float64. Количество пустых значений 83, 4.84%.
Колонка G11_A_A. Тип данных float64. Количество пустых значений 83, 4.84%.
Колонка G12_A_A. Тип данных float64. Количество пустых значений 83, 4.84%.
Колонка KG_A_A. Тип данных float64. Количество пустых значений 83, 4.84%.
Колонка PK_A_A. Тип данных float64. Количество пустых значений 173, 10.09%.
Колонка G01-G08_A_A. Тип данных float64. Количество пустых значений 695, 40.52%.
Колонка G09-G12_A_A. Тип данных float64. Количество пустых значений 644, 37.55%.
Колонка G01_AM_F. Тип данных float64. Количество пустых значений 1308, 76.27%.
Колонка G01_AM_M. Тип данных float64. Количество пустых значений 1307, 76.21%.
Колонка G01_AS_F. Тип данных float64. Количество пустых значений 1307, 76.21%.
Колонка G01_AS_M. Тип данных float64. Количество пустых значений 1307, 76.21%.
Колонка G01_BL_F. Тип данных float64. Количество пустых значений 1307, 76.21%.
```

## Заполнение пропусков

```
cols=["STATE","YEAR","ENROLL", "TOTAL_REVENUE","FEDERAL_REVENUE","STATE_REVENUE","TOTAL_EXPENDITURE","INSTRUCTION_EXPENDITURE",
      "CAPITAL_OUTLAY_EXPENDITURE", "G01_A_A","G06_A_A","G12_A_A","KG_A_A","PK_A_A"]
```

```
imp_mean = SimpleImputer(missing_values=np.nan, strategy='mean')
for column in ["ENROLL", "TOTAL_REVENUE","FEDERAL_REVENUE","STATE_REVENUE","TOTAL_EXPENDITURE","INSTRUCTION_EXPENDITURE",
      "CAPITAL_OUTLAY_EXPENDITURE", "G01_A_A","G06_A_A","G12_A_A","KG_A_A","PK_A_A"]:
    imp_mean.fit(data[[column]])
    data[column] = imp_mean.transform(data[[column]])
```

## Кодирование категориальных признаков

```
LE = LabelEncoder()
for column in ["STATE"]:
    data[column] = LE.fit_transform(data[column])
data
```

| | PRIMARY_KEY | STATE | YEAR | ENROLL | TOTAL_REVENUE | FEDERAL_REVENUE | STATE_REVENUE | LOCAL_REVENUE | TOTAL_EXPENDITURE |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1992_ALABAMA | 0 | 1992 | 917541.566176 | 2.678885e+06 | 3.041770e+05 | 1.659028e+06 | 715680.0 | 2.653798e+06 |
| 1 | 1992_ALASKA | 1 | 1992 | 917541.566176 | 1.049591e+06 | 1.067800e+05 | 7.207110e+05 | 222100.0 | 9.724880e+05 |
| 2 | 1992_ARIZONA | 2 | 1992 | 917541.566176 | 3.258079e+06 | 2.978880e+05 | 1.369815e+06 | 1590376.0 | 3.401580e+06 |
| 3 | 1992_ARKANSAS | 3 | 1992 | 917541.566176 | 1.711959e+06 | 1.785710e+05 | 9.587850e+05 | 574603.0 | 1.743022e+06 |
| 4 | 1992_CALIFORNIA | 4 | 1992 | 917541.566176 | 2.626002e+07 | 2.072470e+06 | 1.654651e+07 | 7641041.0 | 2.713883e+07 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 1710 | 2019_VIRGINIA | 48 | 2019 | 917541.566176 | 9.102045e+06 | 7.677799e+05 | 4.223743e+06 | NaN | 9.206242e+06 |
| 1711 | 2019_WASHINGTON | 49 | 2019 | 917541.566176 | 9.102045e+06 | 7.677799e+05 | 4.223743e+06 | NaN | 9.206242e+06 |
| 1712 | 2019_WEST_VIRGINIA | 50 | 2019 | 917541.566176 | 9.102045e+06 | 7.677799e+05 | 4.223743e+06 | NaN | 9.206242e+06 |
| 1713 | 2019_WISCONSIN | 51 | 2019 | 917541.566176 | 9.102045e+06 | 7.677799e+05 | 4.223743e+06 | NaN | 9.206242e+06 |
| 1714 | 2019_WYOMING | 52 | 2019 | 917541.566176 | 9.102045e+06 | 7.677799e+05 | 4.223743e+06 | NaN | 9.206242e+06 |

1715 rows × 266 columns

## Отбрасывание колонок с большим количеством пропусков

```
data1 = data[cols]
data1
```

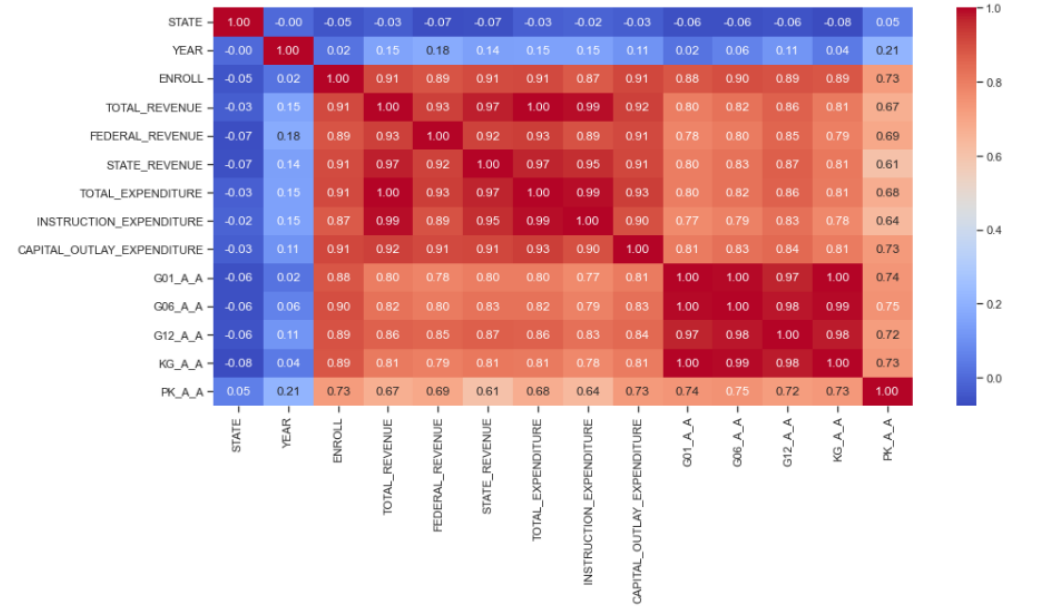
| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 1992 | 917541.566176 | 2.678885e+06 | 3.041770e+05 | 1.659028e+06 | 2.653798e+06 | 1.481703e+06 |
| 1 | 1 | 1992 | 917541.566176 | 1.049591e+06 | 1.067800e+05 | 7.207110e+05 | 9.724880e+05 | 4.983620e+05 |
| 2 | 2 | 1992 | 917541.566176 | 3.258079e+06 | 2.978880e+05 | 1.369815e+06 | 3.401580e+06 | 1.435908e+06 |
| 3 | 3 | 1992 | 917541.566176 | 1.711959e+06 | 1.785710e+05 | 9.587850e+05 | 1.743022e+06 | 9.643230e+05 |
| 4 | 4 | 1992 | 917541.566176 | 2.626002e+07 | 2.072470e+06 | 1.654651e+07 | 2.713883e+07 | 1.435892e+07 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 1710 | 48 | 2019 | 917541.566176 | 9.102045e+06 | 7.677799e+05 | 4.223743e+06 | 9.206242e+06 | 4.768010e+06 |
| 1711 | 49 | 2019 | 917541.566176 | 9.102045e+06 | 7.677799e+05 | 4.223743e+06 | 9.206242e+06 | 4.768010e+06 |
| 1712 | 50 | 2019 | 917541.566176 | 9.102045e+06 | 7.677799e+05 | 4.223743e+06 | 9.206242e+06 | 4.768010e+06 |
| 1713 | 51 | 2019 | 917541.566176 | 9.102045e+06 | 7.677799e+05 | 4.223743e+06 | 9.206242e+06 | 4.768010e+06 |
| 1714 | 52 | 2019 | 917541.566176 | 9.102045e+06 | 7.677799e+05 | 4.223743e+06 | 9.206242e+06 | 4.768010e+06 |

1715 rows × 14 columns

## Корреляционная матрица

```
fig, ax = plt.subplots(figsize=(15,7))
sns.heatmap(data1.corr(method='pearson'), ax=ax, annot=True, cmap='coolwarm', fmt='.2f')
```

```
<AxesSubplot:>
```




## Разделение выборки

```
xArray = data1.drop(["ENROLL","STATE","YEAR","PK_A_A"], axis=1)
yArray = data1["ENROLL"]
xArray
```

| | TOTAL_REVENUE | FEDERAL_REVENUE | STATE_REVENUE | TOTAL_EXPENDITURE | INSTRUCTION_EXPENDITURE | CAPITAL_OUTLAY_EXPENDITURE | G |
|---|---|---|---|---|---|---|---|
| 0 | 2.678885e+06 | 3.041770e+05 | 1.659028e+06 | 2.653798e+06 | 1.481703e+06 | 1.740530e+05 | 58168. |
| 1 | 1.049591e+06 | 1.067800e+05 | 7.207110e+05 | 9.724880e+05 | 4.983620e+05 | 3.745100e+04 | 11030. |
| 2 | 3.258079e+06 | 2.978880e+05 | 1.369815e+06 | 3.401580e+06 | 1.435908e+06 | 6.091140e+05 | 58914. |
| 3 | 1.711959e+06 | 1.785710e+05 | 9.587850e+05 | 1.743022e+06 | 9.643230e+05 | 1.452120e+05 | 34929. |
| 4 | 2.626002e+07 | 2.072470e+06 | 1.654651e+07 | 2.713883e+07 | 1.435892e+07 | 2.044688e+06 | 443955. |
| ... | ... | ... | ... | ... | ... | ... | |
| 1710 | 9.102045e+06 | 7.677799e+05 | 4.223743e+06 | 9.206242e+06 | 4.768010e+06 | 9.034675e+05 | 71699. |
| 1711 | 9.102045e+06 | 7.677799e+05 | 4.223743e+06 | 9.206242e+06 | 4.768010e+06 | 9.034675e+05 | 71699. |
| 1712 | 9.102045e+06 | 7.677799e+05 | 4.223743e+06 | 9.206242e+06 | 4.768010e+06 | 9.034675e+05 | 71699. |
| 1713 | 9.102045e+06 | 7.677799e+05 | 4.223743e+06 | 9.206242e+06 | 4.768010e+06 | 9.034675e+05 | 71699. |
| 1714 | 9.102045e+06 | 7.677799e+05 | 4.223743e+06 | 9.206242e+06 | 4.768010e+06 | 9.034675e+05 | 71699. |

1715 rows × 10 columns

```
trainX, testX, trainY, testY = train_test_split(xArray, yArray, test_size=0.1, random_state=1)
```

## Линейная регрессия

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

```
LinearRegression(normalize=True)
```

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

```
array([1983588.82648057,  443339.57153127,  235967.45529629,
        679636.26057594, 2004474.47547945, 2965255.04049319,
        735084.61450535,  491735.64073718, 2680161.64539013,
        773713.59342328,  401741.65231194,  798052.00208166,
        215101.01095928,  688453.20741952, 3807453.73298959,
        474723.13906104,  199508.79199588,  782935.04307022,
        779136.83171705,  583008.55546649,  916651.80541114,
        847924.44246123,  310879.53903866, 1058300.14406639,
        295034.98120554,  759476.21184398,  916651.80541114,
        241129.08830254,  839736.69191394,  742676.12720593,
        668255.11439305,  806111.84617609, 3219813.86246465,
        944536.4359178 , 2054692.62395223,  958063.83138969,
        683254.36279629,  807651.17042693,  177542.23421354,
        900461.08752499,  794673.59685552, 1682617.07379861,
       1068818.63894029,  755734.05490344,  736538.76923008,
        773252.4601638 ,  962817.37413001,  580092.09724947,
       1764366.27862559, 2395959.72765816,  785608.64716697,
        995969.90400733,  724073.74269521,  916651.80541114,
        752145.28891166,  714331.68689906,  292570.74681139,
        754421.04425677,  225889.27677444,  295815.79875044,
        966187.47414094,  448126.12245493,  550219.68053539,
       1967377.31506269,  438498.99551984,  916651.80541114,
        274626.73664659,  462165.67288484, 5273732.10531777,
        649059.27920511, 1210055.5002643 ,  778755.97289248,
       1777077.66232074, 2131076.01101511,  848832.59105946,
        862769.90723544,  789712.02695688,  574672.39927402,
        192398.86983755,  933442.04434443, 1095076.53335939,
       3237943.3083681 ,  742422.47764501,  413695.44910936,
        748622.61587579,  827519.56187971,  228774.33915843,
       1220396.60223679,  217569.00160342,  857747.35124172,
        752073.25914925,  519829.83399222,  593021.05243273,
        591739.67912358,  780156.31910399,  849534.14088221,
        455328.09357921,  790147.1711178 , 6055438.94813678,
       2029338.88871921, 1359604.13051285, 1618486.42450357,
        353713.74695461,  792733.77424464,  671362.12229638,
        773134.46130091,  277661.48080149,  791618.17816663,
        882945.37503562,  916651.80541114,  748115.12005081,
        811742.15668586, 2878402.28364643,  608951.64584793,
```

Метрики и график предсказаний

```
print('RMSE:', mean_squared_error(y_test_predict_LR, testY, squared=False))
print('MAE:', (mean_absolute_error(testY, y_test_predict_LR)))
```

```
RMSE: 179296.29262567617
MAE: 129564.10776981147
```

```
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()
```


Градиентный бустинг

```
x_Array = data1.drop(["ENROLL","STATE","YEAR","PK_A_A"], axis=1)
y_Array = data1["ENROLL"]
train_X, test_X, train_Y, test_Y = train_test_split(x_Array, y_Array, test_size=0.2, random_state=1)
```

```
#градиентный бустинг
from xgboost import XGBRegressor
model2 = XGBRegressor( booster='gbtree', max_depth=4)
model2.fit(train_X, train_Y)
```

```
XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,
             colsample_bynode=1, colsample_bytree=1, enable_categorical=False,
             gamma=0, gpu_id=-1, importance_type=None,
             interaction_constraints='', learning_rate=0.300000012,
             max_delta_step=0, max_depth=4, min_child_weight=1, missing=nan,
             monotone_constraints='()', n_estimators=100, n_jobs=8,
             num_parallel_tree=1, predictor='auto', random_state=0, reg_alpha=0,
             reg_lambda=1, scale_pos_weight=1, subsample=1, tree_method='exact',
             validate_parameters=1, verbosity=None)
```

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

```
Training score:  0.9996040600407091
```

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

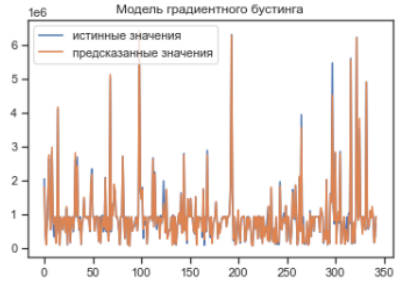
```
array([1799708.5  ,  520878.06 ,   98067.46 ,  727532.94 , 1671415.   ,
       2743484.8  ,  917339.7  ,  544695.56 , 2973854.2  ,  929720.1  ,
        701361.25 ,  914554.75 ,  138678.67 ,  552195.7  , 4159593.2  ,
        532971.6  ,   95161.07 ,  904491.5  ,  827691.2  ,  535084.5  ,
        917765.56 ,  888073.25 ,  448956.16 , 1223758.2  ,  110665.11 ,
        914626.06 ,  917765.56 ,  158237.84 ,  896542.75 ,  914626.06 ,
        530104.25 ,  922221.44 , 2804736.   ,  929290.1  , 2418474.8  ,
        924772.6  ,  548007.25 ,  845432.06 ,  107459.305,  703655.56 ,
        819255.1  , 1493305.8  ,  936337.1  ,  910465.3  ,  856046.94 ,
        920179.6  ,  919470.1  ,  507115.97 , 1882098.2  , 2158425.5  ,
        922119.6  ,  902418.75 ,  496877.34 ,  917765.56 ,  939377.4  ,
        899222.2  ,  204492.69 ,  913413.94 ,  133449.61 ,  542400.44 ,
        928194.06 ,  660562.7  ,  596398.9  , 2061615.1  ,  521845.8  ,
        917765.56 ,  207987.61 ,  508658.28 , 5112961.5  ,  527453.8  ,
       1261657.6  ,  966908.5  , 1879437.   , 1673995.6  ,  916834.3  ,
        930108.1  ,  836423.   ,  535293.94 ,   94661.5  ,  937942.6  ,
        923146.7  , 2704623.   ,  920179.6  ,  276602.5  ,  915422.3  ,
        887121.06 ,  101280.21 ,  923136.8  ,   43063.715,  857296.4  ,
        913733.56 ,  495237.38 ,  657343.3  ,  685445.94 ,  792203.2  ,
        921092.4  ,  497754.88 ,  881083.94 , 6414897.5  , 1829550.6  ,
       1439443.4  , 1737457.8  ,  553917.25 ,  920179.6  ,  813401.7  ,
        914626.06 ,  127099.69 ,  687357.44 ,  912492.25 ,  917765.56 ,
        913608.   ,  813024.06 , 2617080.5  ,  618778.5  , 2185160.
```

## Метрики и график предсказаний

```
print('RMSE:', mean_squared_error(y_test_predict_XGBR, test_Y, squared=False))
print('MAE:', (mean_absolute_error(y_test_predict_XGBR, test_Y)))
```

```
RMSE: 119535.3872361927
MAE: 50205.08998376136
```

```
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()
```




## Сравнение метрик

```
print('Метрики линейной регрессии')
print('RMSE:', mean_squared_error(y_test_predict_LR, testY, squared=False))
print('MAE:', (mean_absolute_error(testY, y_test_predict_LR)))
print('\nМетрики градиентного бустинга')
print('RMSE:', mean_squared_error(y_test_predict_XGBR, test_Y, squared=False))
print('MAE:', (mean_absolute_error(y_test_predict_XGBR, test_Y)))
```

```
Метрики линейной регрессии
RMSE: 179296.29262567617
MAE: 129564.10776981147

Метрики градиентного бустинга
RMSE: 119535.3872361927
MAE: 50205.08998376136
```

В данном случае лучше сработал метод градиентного спуска, т.к. значения его метрик меньше.