Ноздрова В. С.

Группа ИУ5-61Б

Вариант 15

Рубежный контроль №2

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.

Используемый набор данных: U.S. Education Datasets: Unification Project (states_all_extended.csv)

Методы по варианту группы:

- Линейная/логистическая регрессия
- Случайный лес

Загрузка данных

Набор данных содержит следующие колонки:

- PRIMARY_KEY - комбинация года и штата
- YEAR - год
- STATE - штат
- ENROLL - количество учащихся в штате по данным U.S. Census Bureau
- TOTAL_REVENUE - общий доход
- FEDERAL_REVENUE - федеральный доход
- STATE_REVENUE - доход штата
- LOCAL_REVENUE - доход местного правительства
- TOTAL_EXPENDITURE - общий расход
- INSTRUCTION_EXPENDITURE - расходы на обучение
- SUPPORT_SERVICES_EXPENDITURE - расходы на вспомогательные услуги
- CAPITAL_OUTLAY_EXPENDITURE - капитальные расходы
- OTHER_EXPENDITURE - другие расходы

- A_A_A - общее количество учащихся по данным NCES

- G01_A_A - G12_A_A - количество учащихся по годам обучения
- KG_A_A - количество воспитанников старших групп детских садов
- PK_A_A - количество воспитанников младших групп детских садов
- G01-G08_A_A - общее количество учащихся 1-8 классов

- G01_AM_F - PK_WH_M - количество учащихся по ступеням образования, расовой принадлежности и полу
- G04_A_A_READING - средний балл по чтению среди всех учащихся 4 класса
- G04_A_A_MATHEMATICS - средний балл по математике среди всех учащихся 4 класса
- G04_A_M_READING - средний балл по чтению среди мальчиков 4 класса
- G04_A_M_MATHEMATICS - средний балл по математике среди мальчиков 4 класса
- G04_A_F_READING - средний балл по чтению среди девочек 4 класса
- G04_A_F_MATHEMATICS - средний балл по математике среди девочек 4 класса
- G04_WH_A_READING - G04_TR_A_MATHEMATICS - средний балл среди всех учащихся 4 класса с разделением по расовой принадлежности

- остальные колонки содержат аналогичную информацию по тестам среди учеников 8 классов

In [1]:
```
# Импорт библиотек
import numpy as np
import pandas as pd
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, median_absolute_error
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
sns.set(style="ticks")

# Загрузка датасета
data = pd.read_csv('states_all_extended.csv')
```

In [2]:
```
# Первые 5 строк датасета
data.head()
```

Out[2]:

	PRIMARY_KEY	STATE	YEAR	ENROLL	TOTAL_REVENUE	FEDERAL_REVENUE	STATE_REVE
0	1992_ALABAMA	ALABAMA	1992	NaN	2678885.0	304177.0	16590
1	1992_ALASKA	ALASKA	1992	NaN	1049591.0	106780.0	7207
2	1992_ARIZONA	ARIZONA	1992	NaN	3258079.0	297888.0	13698
3	1992_ARKANSAS	ARKANSAS	1992	NaN	1711959.0	178571.0	9587
4	1992_CALIFORNIA	CALIFORNIA	1992	NaN	26260025.0	2072470.0	165465

5 rows × 266 columns

In [3]:
```
# Размер датасета
print('Строк: {}'.format(data.shape[0]))
print('Колонок: {}'.format(data.shape[1]))
```

```
Строк: 1715
Колонок: 266
```

In [4]:
```
```

```
# Типы колонок
data.dtypes
```

Out[4]:

```
PRIMARY_KEY              object
STATE                    object
YEAR                      int64
ENROLL                  float64
TOTAL_REVENUE           float64
                         ...
G08_AM_A_MATHEMATICS    float64
G08_HP_A_READING        float64
G08_HP_A_MATHEMATICS    float64
G08_TR_A_READING        float64
G08_TR_A_MATHEMATICS    float64
Length: 266, dtype: object
```

Обработка пропусков и кодирование признаков

Проверим количество пропусков в колонках:

In [5]:

```
#Типы колонок и количество пропусков в них
print('{:30}  {:10} {}'.format('Колонка', 'Тип', 'Количество пустых значений'))
mis_cols = []
# Цикл по колонкам датасета
for col in data.columns:
    # Количество пустых значений
    temp_null_count = data[data[col].isnull()].shape[0]
    if temp_null_count>0:
        temp_perc = round((temp_null_count / data.shape[0]) * 100.0, 2)
        if temp_perc > 76:
            mis_cols.append(col)
        print('{:30}  {:10} {} ({}%)'.format(col, str(data[col].dtype), temp_null_co
```

```
Колонка                        Тип         Количество пустых значений
ENROLL                         float64     491 (28.63%)
TOTAL_REVENUE                  float64     440 (25.66%)
FEDERAL_REVENUE                float64     440 (25.66%)
STATE_REVENUE                  float64     440 (25.66%)
LOCAL_REVENUE                  float64     440 (25.66%)
TOTAL_EXPENDITURE              float64     440 (25.66%)
INSTRUCTION_EXPENDITURE        float64     440 (25.66%)
SUPPORT_SERVICES_EXPENDITURE   float64     440 (25.66%)
OTHER_EXPENDITURE              float64     491 (28.63%)
CAPITAL_OUTLAY_EXPENDITURE     float64     440 (25.66%)
A_A_A                          float64     83 (4.84%)
G01_A_A                        float64     83 (4.84%)
G02_A_A                        float64     83 (4.84%)
G03_A_A                        float64     83 (4.84%)
G04_A_A                        float64     83 (4.84%)
G05_A_A                        float64     83 (4.84%)
G06_A_A                        float64     83 (4.84%)
G07_A_A                        float64     83 (4.84%)
G08_A_A                        float64     83 (4.84%)
G09_A_A                        float64     83 (4.84%)
G10_A_A                        float64     83 (4.84%)
G11_A_A                        float64     83 (4.84%)
G12_A_A                        float64     83 (4.84%)
KG_A_A                         float64     83 (4.84%)
PK_A_A                         float64     173 (10.09%)
G01-G08_A_A                    float64     695 (40.52%)
G09-G12_A_A                    float64     644 (37.55%)
G01_AM_F                       float64     1308 (76.27%)
```

G01_AM_M	float64	1307 (76.21%)
G01_AS_F	float64	1307 (76.21%)
G01_AS_M	float64	1307 (76.21%)
G01_BL_F	float64	1307 (76.21%)
G01_BL_M	float64	1307 (76.21%)
G01_HI_F	float64	1308 (76.27%)
G01_HI_M	float64	1307 (76.21%)
G01_HP_F	float64	1351 (78.78%)
G01_HP_M	float64	1352 (78.83%)
G01_TR_F	float64	1344 (78.37%)
G01_TR_M	float64	1344 (78.37%)
G01_WH_F	float64	1307 (76.21%)
G01_WH_M	float64	1307 (76.21%)
G02_AM_F	float64	1309 (76.33%)
G02_AM_M	float64	1308 (76.27%)
G02_AS_F	float64	1307 (76.21%)
G02_AS_M	float64	1307 (76.21%)
G02_BL_F	float64	1307 (76.21%)
G02_BL_M	float64	1307 (76.21%)
G02_HI_F	float64	1307 (76.21%)
G02_HI_M	float64	1307 (76.21%)
G02_HP_F	float64	1351 (78.78%)
G02_HP_M	float64	1351 (78.78%)
G02_TR_F	float64	1344 (78.37%)
G02_TR_M	float64	1344 (78.37%)
G02_WH_F	float64	1307 (76.21%)
G02_WH_M	float64	1307 (76.21%)
G03_AM_F	float64	1308 (76.27%)
G03_AM_M	float64	1309 (76.33%)
G03_AS_F	float64	1307 (76.21%)
G03_AS_M	float64	1307 (76.21%)
G03_BL_F	float64	1307 (76.21%)
G03_BL_M	float64	1307 (76.21%)
G03_HI_F	float64	1307 (76.21%)
G03_HI_M	float64	1307 (76.21%)
G03_HP_F	float64	1352 (78.83%)
G03_HP_M	float64	1349 (78.66%)
G03_TR_F	float64	1344 (78.37%)
G03_TR_M	float64	1344 (78.37%)
G03_WH_F	float64	1307 (76.21%)
G03_WH_M	float64	1307 (76.21%)
G04_AM_F	float64	1308 (76.27%)
G04_AM_M	float64	1308 (76.27%)
G04_AS_F	float64	1307 (76.21%)
G04_AS_M	float64	1307 (76.21%)
G04_BL_F	float64	1307 (76.21%)
G04_BL_M	float64	1307 (76.21%)
G04_HI_F	float64	1307 (76.21%)
G04_HI_M	float64	1307 (76.21%)
G04_HP_F	float64	1351 (78.78%)
G04_HP_M	float64	1351 (78.78%)
G04_TR_F	float64	1344 (78.37%)
G04_TR_M	float64	1344 (78.37%)
G04_WH_F	float64	1307 (76.21%)
G04_WH_M	float64	1307 (76.21%)
G05_AM_F	float64	1308 (76.27%)
G05_AM_M	float64	1308 (76.27%)
G05_AS_F	float64	1307 (76.21%)
G05_AS_M	float64	1307 (76.21%)
G05_BL_F	float64	1307 (76.21%)
G05_BL_M	float64	1307 (76.21%)
G05_HI_F	float64	1307 (76.21%)
G05_HI_M	float64	1307 (76.21%)
G05_HP_F	float64	1352 (78.83%)

G05_HP_M	float64	1349 (78.66%)
G05_TR_F	float64	1344 (78.37%)
G05_TR_M	float64	1344 (78.37%)
G05_WH_F	float64	1307 (76.21%)
G05_WH_M	float64	1307 (76.21%)
G06_AM_F	float64	1307 (76.21%)
G06_AM_M	float64	1308 (76.27%)
G06_AS_F	float64	1307 (76.21%)
G06_AS_M	float64	1307 (76.21%)
G06_BL_F	float64	1307 (76.21%)
G06_BL_M	float64	1307 (76.21%)
G06_HI_F	float64	1307 (76.21%)
G06_HI_M	float64	1307 (76.21%)
G06_HP_F	float64	1351 (78.78%)
G06_HP_M	float64	1349 (78.66%)
G06_TR_F	float64	1344 (78.37%)
G06_TR_M	float64	1344 (78.37%)
G06_WH_F	float64	1307 (76.21%)
G06_WH_M	float64	1307 (76.21%)
G07_AM_F	float64	1307 (76.21%)
G07_AM_M	float64	1308 (76.27%)
G07_AS_F	float64	1307 (76.21%)
G07_AS_M	float64	1307 (76.21%)
G07_BL_F	float64	1307 (76.21%)
G07_BL_M	float64	1307 (76.21%)
G07_HI_F	float64	1307 (76.21%)
G07_HI_M	float64	1307 (76.21%)
G07_HP_F	float64	1350 (78.72%)
G07_HP_M	float64	1352 (78.83%)
G07_TR_F	float64	1344 (78.37%)
G07_TR_M	float64	1344 (78.37%)
G07_WH_F	float64	1307 (76.21%)
G07_WH_M	float64	1307 (76.21%)
G08_AM_F	float64	1308 (76.27%)
G08_AM_M	float64	1307 (76.21%)
G08_AS_F	float64	1307 (76.21%)
G08_AS_M	float64	1307 (76.21%)
G08_BL_F	float64	1307 (76.21%)
G08_BL_M	float64	1307 (76.21%)
G08_HI_F	float64	1307 (76.21%)
G08_HI_M	float64	1307 (76.21%)
G08_HP_F	float64	1350 (78.72%)
G08_HP_M	float64	1349 (78.66%)
G08_TR_F	float64	1344 (78.37%)
G08_TR_M	float64	1344 (78.37%)
G08_WH_F	float64	1307 (76.21%)
G08_WH_M	float64	1307 (76.21%)
G09_AM_F	float64	1307 (76.21%)
G09_AM_M	float64	1308 (76.27%)
G09_AS_F	float64	1307 (76.21%)
G09_AS_M	float64	1307 (76.21%)
G09_BL_F	float64	1307 (76.21%)
G09_BL_M	float64	1307 (76.21%)
G09_HI_F	float64	1307 (76.21%)
G09_HI_M	float64	1307 (76.21%)
G09_HP_F	float64	1352 (78.83%)
G09_HP_M	float64	1351 (78.78%)
G09_TR_F	float64	1344 (78.37%)
G09_TR_M	float64	1344 (78.37%)
G09_WH_F	float64	1307 (76.21%)
G09_WH_M	float64	1307 (76.21%)
G10_AM_F	float64	1307 (76.21%)
G10_AM_M	float64	1308 (76.27%)
G10_AS_F	float64	1307 (76.21%)

G10_AS_M	float64	1307 (76.21%)
G10_BL_F	float64	1307 (76.21%)
G10_BL_M	float64	1307 (76.21%)
G10_HI_F	float64	1307 (76.21%)
G10_HI_M	float64	1307 (76.21%)
G10_HP_F	float64	1352 (78.83%)
G10_HP_M	float64	1351 (78.78%)
G10_TR_F	float64	1344 (78.37%)
G10_TR_M	float64	1344 (78.37%)
G10_WH_F	float64	1307 (76.21%)
G10_WH_M	float64	1307 (76.21%)
G11_AM_F	float64	1307 (76.21%)
G11_AM_M	float64	1307 (76.21%)
G11_AS_F	float64	1307 (76.21%)
G11_AS_M	float64	1307 (76.21%)
G11_BL_F	float64	1307 (76.21%)
G11_BL_M	float64	1307 (76.21%)
G11_HI_F	float64	1308 (76.27%)
G11_HI_M	float64	1307 (76.21%)
G11_HP_F	float64	1352 (78.83%)
G11_HP_M	float64	1354 (78.95%)
G11_TR_F	float64	1344 (78.37%)
G11_TR_M	float64	1344 (78.37%)
G11_WH_F	float64	1307 (76.21%)
G11_WH_M	float64	1307 (76.21%)
G12_AM_F	float64	1309 (76.33%)
G12_AM_M	float64	1310 (76.38%)
G12_AS_F	float64	1307 (76.21%)
G12_AS_M	float64	1307 (76.21%)
G12_BL_F	float64	1307 (76.21%)
G12_BL_M	float64	1307 (76.21%)
G12_HI_F	float64	1307 (76.21%)
G12_HI_M	float64	1307 (76.21%)
G12_HP_F	float64	1352 (78.83%)
G12_HP_M	float64	1352 (78.83%)
G12_TR_F	float64	1344 (78.37%)
G12_TR_M	float64	1344 (78.37%)
G12_WH_F	float64	1307 (76.21%)
G12_WH_M	float64	1307 (76.21%)
KG_AM_F	float64	1307 (76.21%)
KG_AM_M	float64	1308 (76.27%)
KG_AS_F	float64	1307 (76.21%)
KG_AS_M	float64	1307 (76.21%)
KG_BL_F	float64	1307 (76.21%)
KG_BL_M	float64	1307 (76.21%)
KG_HI_F	float64	1308 (76.27%)
KG_HI_M	float64	1307 (76.21%)
KG_HP_F	float64	1349 (78.66%)
KG_HP_M	float64	1350 (78.72%)
KG_TR_F	float64	1344 (78.37%)
KG_TR_M	float64	1344 (78.37%)
KG_WH_F	float64	1307 (76.21%)
KG_WH_M	float64	1307 (76.21%)
PK_AM_F	float64	1332 (77.67%)
PK_AM_M	float64	1321 (77.03%)
PK_AS_F	float64	1321 (77.03%)
PK_AS_M	float64	1323 (77.14%)
PK_BL_F	float64	1321 (77.03%)
PK_BL_M	float64	1321 (77.03%)
PK_HI_F	float64	1321 (77.03%)
PK_HI_M	float64	1321 (77.03%)
PK_HP_F	float64	1387 (80.87%)
PK_HP_M	float64	1384 (80.7%)
PK_TR_F	float64	1357 (79.13%)

```
PK_TR_M                         float64    1357 (79.13%)
PK_WH_F                         float64    1321 (77.03%)
PK_WH_M                         float64    1321 (77.03%)
G04_A_A_READING                 float64    1065 (62.1%)
G04_A_A_MATHEMATICS             float64    1150 (67.06%)
G04_A_M_READING                 float64    1065 (62.1%)
G04_A_M_MATHEMATICS             float64    1150 (67.06%)
G04_A_F_READING                 float64    1065 (62.1%)
G04_A_F_MATHEMATICS             float64    1150 (67.06%)
G04_WH_A_READING                float64    1450 (84.55%)
G04_WH_A_MATHEMATICS            float64    1450 (84.55%)
G04_BL_A_READING                float64    1489 (86.82%)
G04_BL_A_MATHEMATICS            float64    1486 (86.65%)
G04_HI_A_READING                float64    1465 (85.42%)
G04_HI_A_MATHEMATICS            float64    1465 (85.42%)
G04_AS_A_READING                float64    1551 (90.44%)
G04_AS_A_MATHEMATICS            float64    1547 (90.2%)
G04_AM_A_READING                float64    1651 (96.27%)
G04_AM_A_MATHEMATICS            float64    1652 (96.33%)
G04_HP_A_READING                float64    1699 (99.07%)
G04_HP_A_MATHEMATICS            float64    1700 (99.13%)
G04_TR_A_READING                float64    1532 (89.33%)
G04_TR_A_MATHEMATICS            float64    1532 (89.33%)
G08_A_A_READING                 float64    1153 (67.23%)
G08_A_A_MATHEMATICS             float64    1113 (64.9%)
G08_A_M_READING                 float64    1153 (67.23%)
G08_A_M_MATHEMATICS             float64    1113 (64.9%)
G08_A_F_READING                 float64    1153 (67.23%)
G08_A_F_MATHEMATICS             float64    1113 (64.9%)
G08_WH_A_READING                float64    1450 (84.55%)
G08_WH_A_MATHEMATICS            float64    1450 (84.55%)
G08_BL_A_READING                float64    1493 (87.06%)
G08_BL_A_MATHEMATICS            float64    1494 (87.11%)
G08_HI_A_READING                float64    1469 (85.66%)
G08_HI_A_MATHEMATICS            float64    1467 (85.54%)
G08_AS_A_READING                float64    1562 (91.08%)
G08_AS_A_MATHEMATICS            float64    1558 (90.85%)
G08_AM_A_READING                float64    1654 (96.44%)
G08_AM_A_MATHEMATICS            float64    1655 (96.5%)
G08_HP_A_READING                float64    1701 (99.18%)
G08_HP_A_MATHEMATICS            float64    1702 (99.24%)
G08_TR_A_READING                float64    1574 (91.78%)
G08_TR_A_MATHEMATICS            float64    1570 (91.55%)
```

В колонках, содержащих информацию о количестве учащихся по ступеням образования, расовой принадлежности и полу и о результатах тестов с аналогичным разделением, имеется от 76% до 99% пропусков, поэтому эти колонки нецелесообразно использовать для построения модели и можно их удалить.

In [6]:

```
data_new = data.drop(axis=1, columns=mis_cols).copy()
print('{:30}  {:10} {}'.format('Колонка', 'Тип', 'Количество пустых значений'))
mis_rows = []
for col in data_new.columns:
    # Количество пустых значений
    temp_null_count = data_new[data_new[col].isnull()].shape[0]
    if temp_null_count>0:
        temp_perc = round((temp_null_count / data_new.shape[0]) * 100.0, 2)
        if temp_perc > 0:
            mis_rows.append(col)
        print('{:30}  {:10} {} ({}%)'.format(col, str(data_new[col].dtype), temp_nul
```

```
Колонка                         Тип        Количество пустых значений
```

```
ENROLL                          float64    491 (28.63%)
TOTAL_REVENUE                   float64    440 (25.66%)
FEDERAL_REVENUE                 float64    440 (25.66%)
STATE_REVENUE                   float64    440 (25.66%)
LOCAL_REVENUE                   float64    440 (25.66%)
TOTAL_EXPENDITURE               float64    440 (25.66%)
INSTRUCTION_EXPENDITURE         float64    440 (25.66%)
SUPPORT_SERVICES_EXPENDITURE    float64    440 (25.66%)
OTHER_EXPENDITURE               float64    491 (28.63%)
CAPITAL_OUTLAY_EXPENDITURE      float64    440 (25.66%)
A_A_A                           float64    83 (4.84%)
G01_A_A                         float64    83 (4.84%)
G02_A_A                         float64    83 (4.84%)
G03_A_A                         float64    83 (4.84%)
G04_A_A                         float64    83 (4.84%)
G05_A_A                         float64    83 (4.84%)
G06_A_A                         float64    83 (4.84%)
G07_A_A                         float64    83 (4.84%)
G08_A_A                         float64    83 (4.84%)
G09_A_A                         float64    83 (4.84%)
G10_A_A                         float64    83 (4.84%)
G11_A_A                         float64    83 (4.84%)
G12_A_A                         float64    83 (4.84%)
KG_A_A                          float64    83 (4.84%)
PK_A_A                          float64    173 (10.09%)
G01-G08_A_A                     float64    695 (40.52%)
G09-G12_A_A                     float64    644 (37.55%)
G04_A_A_READING                 float64    1065 (62.1%)
G04_A_A_MATHEMATICS             float64    1150 (67.06%)
G04_A_M_READING                 float64    1065 (62.1%)
G04_A_M_MATHEMATICS             float64    1150 (67.06%)
G04_A_F_READING                 float64    1065 (62.1%)
G04_A_F_MATHEMATICS             float64    1150 (67.06%)
G08_A_A_READING                 float64    1153 (67.23%)
G08_A_A_MATHEMATICS             float64    1113 (64.9%)
G08_A_M_READING                 float64    1153 (67.23%)
G08_A_M_MATHEMATICS             float64    1113 (64.9%)
G08_A_F_READING                 float64    1153 (67.23%)
G08_A_F_MATHEMATICS             float64    1113 (64.9%)
```

Колонки с результатами тестов содержат до 67% пропусков, но т.к. они являются целевыми, удалять их нельзя. Поэтому удалим строки с пропущенными значениями.

In [7]:
```
data_new = data_new.dropna(axis=0, subset=mis_rows).copy()
print(data_new.shape)
```

```
(355, 42)
```

Таким образом, получен фрагмент датасета без пропусков, содержащий 355 строк, что допускается по условию задачи. Также в датасете имеется один категориальный признак, требующий кодирования, - колонка STATE.

In [8]:
```
# Уникальные значения колонки STATE
data_new['STATE'].unique()
```

Out[8]:
```
array(['ALABAMA', 'ALASKA', 'ARIZONA', 'ARKANSAS', 'CALIFORNIA',
       'COLORADO', 'CONNECTICUT', 'DELAWARE', 'DISTRICT_OF_COLUMBIA',
       'FLORIDA', 'GEORGIA', 'HAWAII', 'IDAHO', 'ILLINOIS', 'INDIANA',
       'IOWA', 'KANSAS', 'KENTUCKY', 'LOUISIANA', 'MAINE', 'MARYLAND',
       'MASSACHUSETTS', 'MICHIGAN', 'MINNESOTA', 'MISSISSIPPI',
       'MISSOURI', 'MONTANA', 'NEBRASKA', 'NEVADA', 'NEW_HAMPSHIRE',
       'NEW_JERSEY', 'NEW_MEXICO', 'NEW_YORK', 'NORTH_CAROLINA',
```

```
'NORTH_DAKOTA', 'OHIO', 'OKLAHOMA', 'OREGON', 'PENNSYLVANIA',
'RHODE_ISLAND', 'SOUTH_CAROLINA', 'SOUTH_DAKOTA', 'TENNESSEE',
'TEXAS', 'UTAH', 'VERMONT', 'VIRGINIA', 'WASHINGTON',
'WEST_VIRGINIA', 'WISCONSIN', 'WYOMING'], dtype=object)
```

In [9]:
```
# Кодирование значений колонки STATE числовыми значениями
le = LabelEncoder()
data_new['STATE'] = le.fit_transform(data_new['STATE'])
```

Построение моделей

Будем исследовать зависимость оценок в 8 классах от трат на образование, поэтому удалим лишние столбцы, а также введем столбец со средним баллом за оба теста.

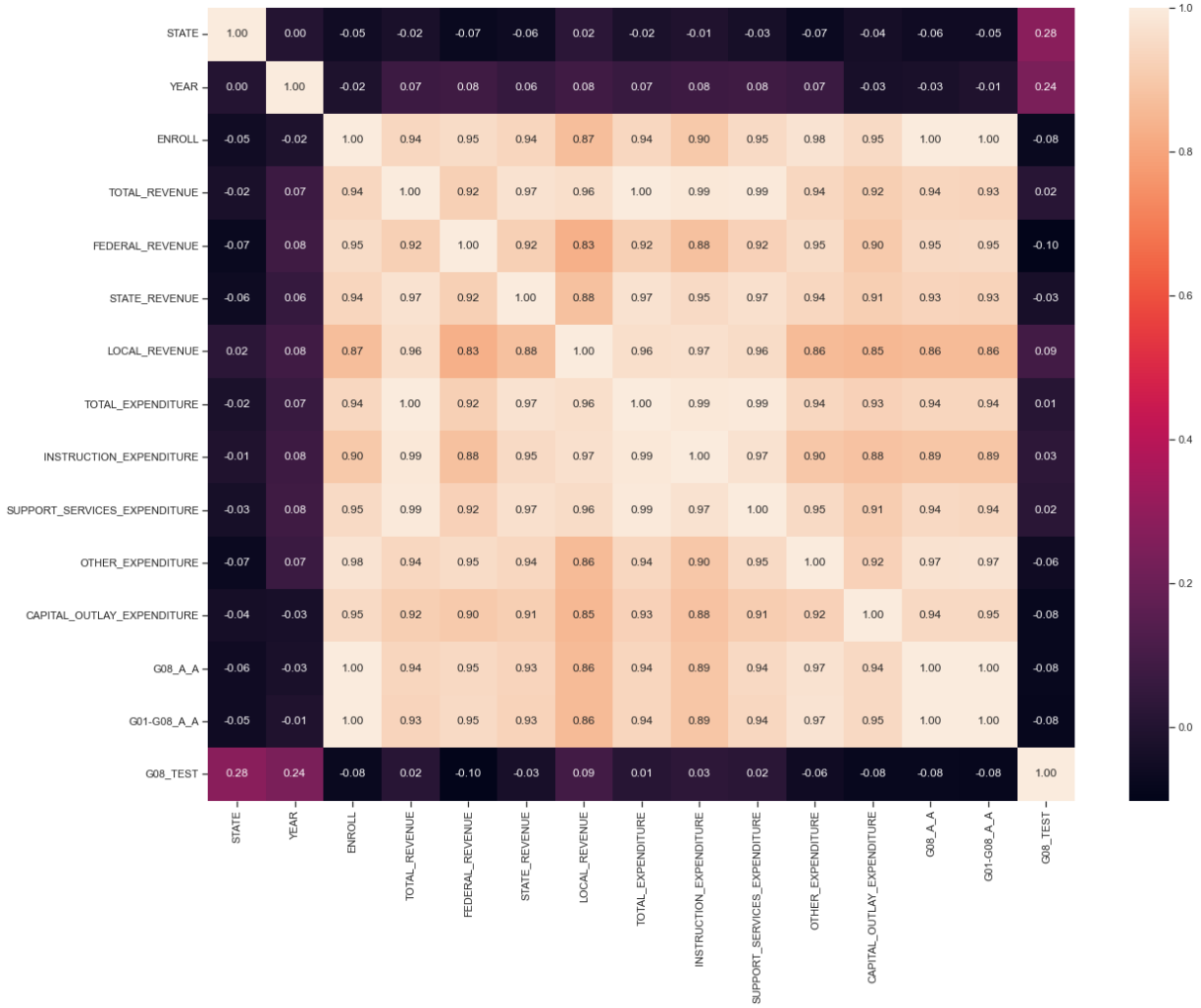
Для оценки качества моделей будем использовать следующие метрики:

- Средняя квадратичная ошибка (MSE)
- Медианная абсолютная ошибка (MedAE)

In [10]:
```
del_cols = ['PRIMARY_KEY','A_A_A', 'G09_A_A', 'G10_A_A','G11_A_A','G12_A_A','G09-G12
data_new['G08_TEST'] = data_new.apply (lambda row: (row['G08_A_A_READING']+row['G08_
data_new = data_new.drop(axis=1, columns=del_cols).copy()
```

In [11]:
```
fig, ax = plt.subplots(1, 1, sharex='col', sharey='row', figsize=(20,15))
sns.heatmap(data_new.corr(), ax=ax, annot=True, fmt='.2f')
```

Out[11]:
```
<AxesSubplot:>
```



In [12]:
```
# Разделение данных на обучающую и тестовую выборки
train_cols = data_new.columns.difference(['G08_TEST'])
data_x_train, data_x_test, data_y_train, data_y_test = train_test_split(
    data_new[train_cols], data_new['G08_TEST'], test_size=0.3, random_state=1)
```

In [13]:
```
# Размер обучающей выборки
data_x_train.shape, data_y_train.shape
```

Out[13]: ((248, 14), (248,))

In [14]:
```
# Размер тестовой выборки
data_x_test.shape, data_y_test.shape
```

Out[14]: ((107, 14), (107,))

In [15]:
```
# Линейная регрессия
lr = LinearRegression().fit(data_x_train, data_y_train)
lr_mse = mean_squared_error(data_y_test, lr.predict(data_x_test), squared = True)
lr_medae = median_absolute_error(data_y_test, lr.predict(data_x_test))
```

In [16]:
```
# Случайный лес
rf = RandomForestRegressor(random_state=1)
rf.fit(data_x_train, data_y_train)
rf_mse = mean_squared_error(data_y_test, rf.predict(data_x_test), squared = True)
rf_medae = median_absolute_error(data_y_test, rf.predict(data_x_test))
```

In [17]:

```
# Вывод метрик
print('{:5} {:>20} {:>20}'.format('','Линейная регрессия', 'Случайный лес'))
print('{:5} {:20} {:20}'.format('MSE', lr_mse, rf_mse))
print('{:5} {:20} {:20}'.format('MedAE', lr_medae, rf_medae))
```

```
        Линейная регрессия        Случайный лес
MSE      46.51973688145895    9.797897196261689
MedAE    4.997387605429481    1.704999999999984
```

Таким образом, ансамблевая модель случайного леса лучше предсказала значения, чем модель линейной регрессии.