

▼ Рубежный контроль 1


Стадник Елена, 15 вариант, задание 2, номер датасета 7

▼ Задание

▼ Загрузка датасета

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from google.colab import files
%matplotlib inline
sns.set(style="ticks")
```

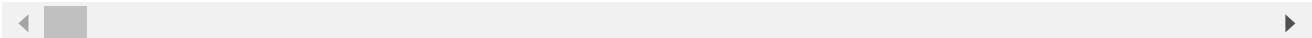
```
files.upload()
```

 states_all...ended.csv

- **states_all_extended.csv**(text/csv) - 1275647 bytes, last modified: 13.04.2020 - 100% done

Saving states_all_extended.csv to states_all_extended.csv

{'states_all_extended.csv': b'PRIMARY_KEY,STATE,YEAR,ENROLL,TOTAL_REVENUE,FEDERAL_RE\



```
data = pd.read_csv('states_all_extended.csv')
```

▼ Первичный анализ датасета

Первые 5 строк датасета:

```
data.head()
```

	PRIMARY_KEY	STATE	YEAR	ENROLL	TOTAL_REVENUE	FEDERAL_REVENUE	STAT
0	1992_ALABAMA	ALABAMA	1992	NaN	2678885.0	304177.0	
1	1992_ALASKA	ALASKA	1992	NaN	1040501.0	106780.0	

Определим размер датасета:

```
data.shape
```

```
(1715, 266)
```

Определим типы данных:

```
data.dtypes
```

```
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

Проверим наличие 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
```

Датасет невероятно огромный, работать с ним в таком виде нет смысла. Тем более в нём слишком много пропусков в некоторых колонках, которые никак невозможно заполнить, полагаясь на простейшие встроенные алгоритмы. В связи с этим выберем те

данные, с которым будем работать - будем искать зависимость между оценками по чтению и математике у студентов 8 класса 4 национальностей - азиатов, афроамериканцев, латино-американцев и белых.

```
data_new = data[["YEAR", "G08_AS_A_READING", "G08_AS_A_MATHEMATICS", "G08_HI_A_READING", "
```

```
data_new.isnull().sum()
```

```
YEAR          0
G08_AS_A_READING    1562
G08_AS_A_MATHEMATICS 1558
G08_HI_A_READING    1469
G08_HI_A_MATHEMATICS 1467
G08_WH_A_READING    1450
G08_WH_A_MATHEMATICS 1450
G08_BL_A_READING    1493
G08_BL_A_MATHEMATICS 1494
dtype: int64
```

Пропущенные значения всё ещё есть - удаляем их.

```
data_new = data_new.dropna(axis=0, how='any')
```

```
data_new.shape
```

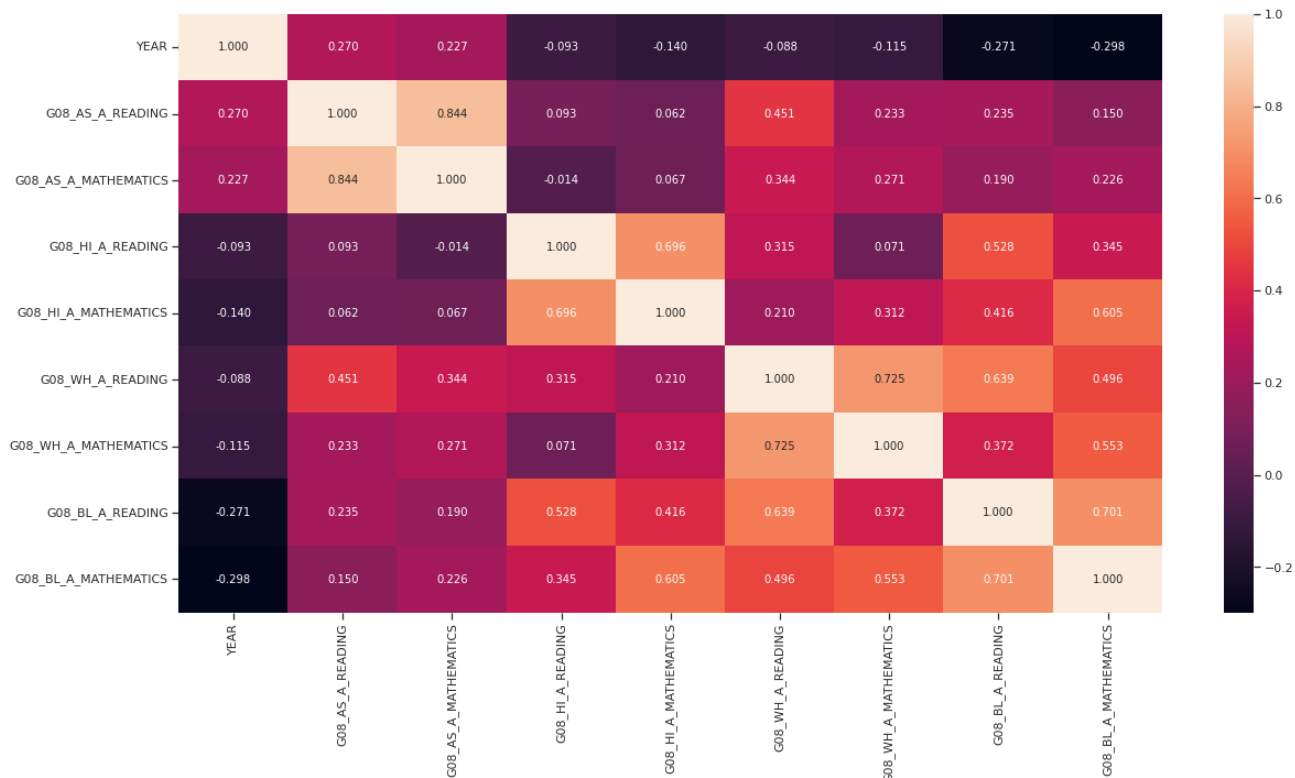
```
(134, 9)
```

В датасете имеется больше 100 значений, будем продолжать работу.

▼ Корреляционная матрица

```
ig, ax = plt.subplots(figsize=(20,10))
sns.heatmap(data_new.corr(method='pearson'), ax=ax, annot=True, fmt='.3f')
```

<matplotlib.axes._subplots.AxesSubplot at 0x7fe837a64d90>



Видим достаточно большую корреляцию между средней оценкой азиатов по математике и по чтению.

▼ Делим выборку на обучающую и тестовую

```
from sklearn.model_selection import train_test_split
```

```
X_train, X_test, y_train, y_test = train_test_split(data_new, data_new["G08_AS_A_MATHEMATICS"],
```

Размеры обучающей выборки и тестовой выборки:

```
X_train.shape, y_train.shape, X_test.shape, y_test.shape
```

```
((100, 9), (100,), (34, 9), (34,))
```

▼ Дерево решений

```
from sklearn.tree import DecisionTreeRegressor
```

```
def stat_tree(estimator):
    n_nodes = estimator.tree_.node_count
    children_left = estimator.tree_.children_left
    children_right = estimator.tree_.children_right

    node_depth = np.zeros(shape=n_nodes, dtype=np.int64)
    is_leaves = np.zeros(shape=n_nodes, dtype=bool)
    stack = [(0, -1)] # seed is the root node id and its parent depth
    while len(stack) > 0:
        node_id, parent_depth = stack.pop()
        node_depth[node_id] = parent_depth + 1

        # If we have a test node
        if (children_left[node_id] != children_right[node_id]):
            stack.append((children_left[node_id], parent_depth + 1))
            stack.append((children_right[node_id], parent_depth + 1))
        else:
            is_leaves[node_id] = True

    print("Всего узлов:", n_nodes)
    print("Листовых узлов:", sum(is_leaves))
    print("Глубина дерева:", max(node_depth))
    print("Минимальная глубина листьев дерева:", min(node_depth[is_leaves]))
    print("Средняя глубина листьев дерева:", node_depth[is_leaves].mean())
```

Построим модель дерева с глубиной = 5:

```
regr1 = DecisionTreeRegressor(max_depth=5)
model1 = regr1.fit(X_train, y_train)
```

```
y_pred_2 = model1.predict(X_test)
```

Выведем основную статистику для дерева:

```
stat_tree(model1)

Всего узлов: 55
Листовых узлов: 28
Глубина дерева: 5
Минимальная глубина листьев дерева: 4
Средняя глубина листьев дерева: 4.857142857142857
```

▼ График важности признаков:

```
from operator import itemgetter

def draw_feature_importances(tree_model, X_dataset, figsize=(18,5)):
    """
    Вывод важности признаков в виде графика
```

```
"""
```

```
# Сортировка значений важности признаков по убыванию
list_to_sort = list(zip(X_dataset.columns.values, tree_model.feature_importances_))
sorted_list = sorted(list_to_sort, key=itemgetter(1), reverse = True)
# Названия признаков
labels = [x for x,_ in sorted_list]
# Важности признаков
data = [x for _,x in sorted_list]
# Вывод графика
fig, ax = plt.subplots(figsize=figsize)
ind = np.arange(len(labels))
plt.bar(ind, data)
plt.xticks(ind, labels, rotation='vertical')
# Вывод значений
for a,b in zip(ind, data):
    plt.text(a-0.05, b+0.01, str(round(b,3)))
plt.show()
return labels, data
```

```
list(zip(data_new.columns.values, model1.feature_importances_))
```

```
[('YEAR', 0.0),
 ('G08_AS_A_READING', 0.0006553293792491219),
 ('G08_AS_A_MATHEMATICS', 0.999087888496633),
 ('G08_HI_A_READING', 0.0),
 ('G08_HI_A_MATHEMATICS', 0.0002166599172254564),
 ('G08_WH_A_READING', 0.0),
 ('G08_WH_A_MATHEMATICS', 4.012220689256507e-05),
 ('G08_BL_A_READING', 0.0),
 ('G08_BL_A_MATHEMATICS', 0.0)]
```

```
grade_tree_cl_fl_1, grade_tree_cl_fd_1 = draw_feature_importances(model1, data_new)
```



```
from sklearn.ensemble import AdaBoostClassifier
from sklearn.metrics import accuracy_score, balanced_accuracy_score
from sklearn.metrics import precision_score, recall_score, f1_score, classification_report
from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error
```



```
r2_score(y_test, y_pred_2)
```

```
0.996213642430156
```



```
mean_squared_error(y_test, y_pred_2)
```

```
0.48674019607843205
```

```
mean_absolute_error(y_test, y_pred_2)
```

```
0.5014705882352928
```

▼ Градиентный бустинг

```
ab1 = AdaBoostClassifier()
ab1.fit(X_train, y_train)
```

```
AdaBoostClassifier()
```

```
y_pred_1 = ab1.predict(X_test)
```

```
y_pred_1_0 = ab1.predict(X_train)
```

```
accuracy_score(y_train, y_pred_1_0)
```

```
0.49
```

```
accuracy_score(y_test, y_pred_1)
```

```
0.17647058823529413
```

```
f1_score(y_test, y_pred_1, average='micro')
```

```
0.17647058823529413
```

```
f1_score(y_test, y_pred_1, average='macro')
```

```
0.10666666666666667
```

```
f1_score(y_train, y_pred_1_0, average='weighted')
```

```
0.36543675213675214
```

✓ 0 сек. выполнено в 18:33

