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_____(ИУ)___

КАФЕДРА ____СИСТЕМЫ ОБРАБОТКИ ИНФОРМАЦИИ И УПРАВЛЕНИЯ___(ИУ5)__

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ

НА ТЕМУ:

Исследование факторов влияющих на успеваемость студентов высших учебных заведений США

Студент ___**ИУ5-33М**____ _____ __**Болотин А.С.**_____
(Группа) (Подпись, дата) (И.О.Фамилия)

Руководитель _____ __**Гапанюк Ю.Е.** ___
(Подпись, дата) (И.О.Фамилия)

Консультант _____ _____
(Подпись, дата) (И.О.Фамилия)

2021 г.

Цель работы

Рассмотрим данные об успеваемости американских студентов. Из них был сформирован датасет, состоящий из 8 столбцов и тысячи строк. В датасете учтены такие параметры, как пол, уровень образования, раса, наличие курсов подготовки к тесту и наличие платных обедов в учебном заведении. Проведём исследование, чтобы выявить, какие факторы оказывают решающее влияние на итоговый результат и разработать модель, которая могла бы предсказать успешную сдачу экзаменов по имеющемуся набору характеристик.

Решение задачи

Датасет был исследован на наличие пропусков, которых не было обнаружено. После был закодирован целевой признак gender (женскому полу было присвоено значение 0, мужскому – 1). Остальные категориальные признаки были закодированы с помощью TargetEncoder

```
student = pd.read_csv('C:/archive/StudentsPerformance.csv')
student
```

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score
0	female	group B	bachelor's degree	standard	none	72	72	74
1	female	group C	some college	standard	completed	69	90	88
2	female	group B	master's degree	standard	none	90	95	93
3	male	group A	associate's degree	free/reduced	none	47	57	44
4	male	group C	some college	standard	none	76	78	75
5	female	group B	associate's degree	standard	none	71	83	78
6	female	group B	some college	standard	completed	88	95	92
7	male	group B	some college	free/reduced	none	40	43	39
8	male	group D	high school	free/reduced	completed	64	64	67

```
data_features = list(zip(
# признаки
[i for i in student.columns],
zip(
    # типы колонок
    [str(i) for i in student.dtypes],
    # проверим есть ли пропущенные значения
    [i for i in student.isnull().sum()]
)))
# Признаки с типом данных и количеством пропусков
data_features
```

```
[('gender', ('object', 0)),
 ('race/ethnicity', ('object', 0)),
 ('parental level of education', ('object', 0)),
 ('lunch', ('object', 0)),
 ('test preparation course', ('object', 0)),
 ('math score', ('int64', 0)),
 ('reading score', ('int64', 0)),
 ('writing score', ('int64', 0))]
```

```
dct = {'female': 0, 'male': 1}
student['gender'] = student['gender'].map(dct)
student.head()
```

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score
0	0	group B	bachelor's degree	standard	none	72	72	74
1	0	group C	some college	standard	completed	69	90	88
2	0	group B	master's degree	standard	none	90	95	93
3	1	group A	associate's degree	free/reduced	none	47	57	44
4	1	group C	some college	standard	none	76	78	75

```
ce_TargetEncoder1 = ce_TargetEncoder()
student_MEAN_ENC = ce_TargetEncoder1.fit_transform(student[student.columns.difference(['gender'])], student['gender'])
student_MEAN_ENC.head()
```

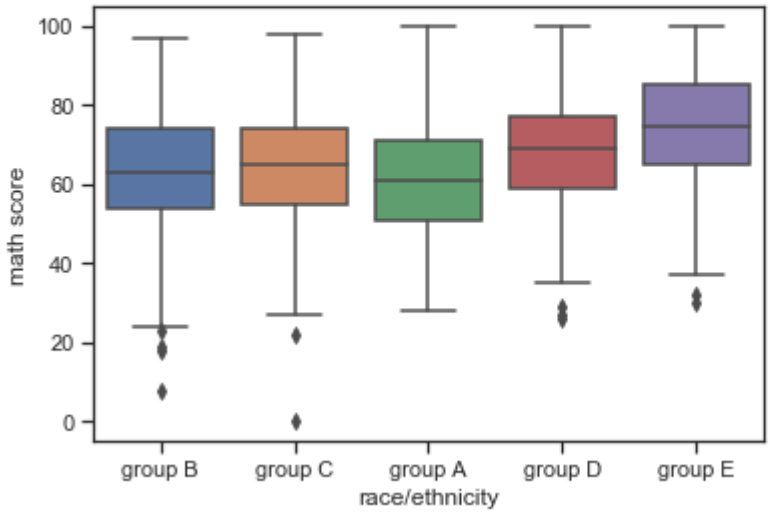
	lunch	math score	parental level of education	race/ethnicity	reading score	test preparation course	writing score
0	0.489922	72	0.466102	0.452632	72	0.479751	74
1	0.489922	69	0.477876	0.435737	90	0.486034	88
2	0.489922	90	0.389831	0.452632	95	0.479751	93
3	0.467606	47	0.477477	0.595506	57	0.479751	44
4	0.489922	76	0.477876	0.435737	78	0.479751	75

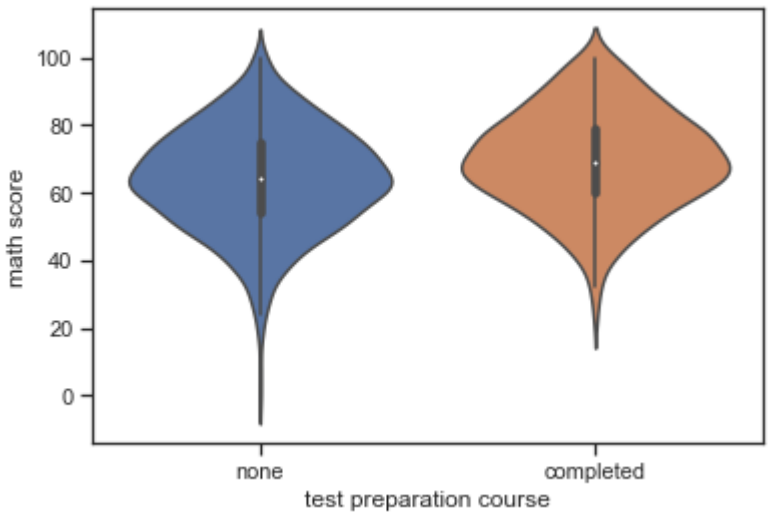
Построим несколько графиков, чтобы посмотреть влияние параметров. Видно, что этническая группа E справляется с тестом по математике лучше всех, пройденный подготовительный тест на высокую оценку влияет незначительно, хотя среди тех, кто тест прошёл, низких баллов гораздо меньше, результаты по математике и письму с чтением различаются, и те, кто получил высокий балл за второе, с большой вероятностью получит такой же и за третье. Также видно, что женщин гораздо больше, чем мужчин, а подготовку к тесту не прошло в два

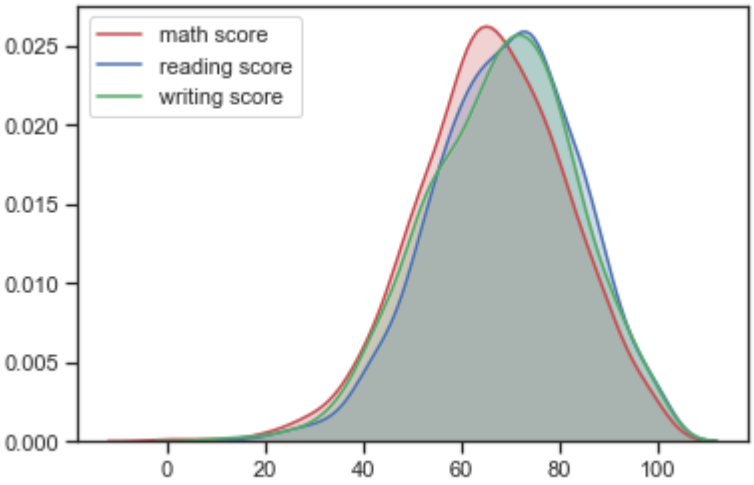
раза больше учеников, чем прошло. Связь с образованием очевидна, чем оно выше, тем больше высоких баллов. Также высокую корреляцию с хорошими результатами показало наличие платных обедов, что можно объяснить так: позволить их могли состоятельные семьи, в которых дети росли в лучших условиях и имели возможность получать доп образование.

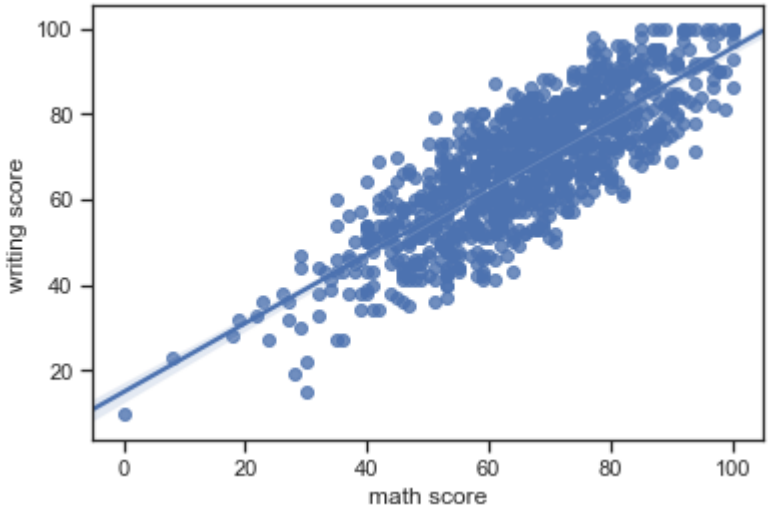
```
sns.boxplot( x=student["race/ethnicity"], y=student["math score"] )
```

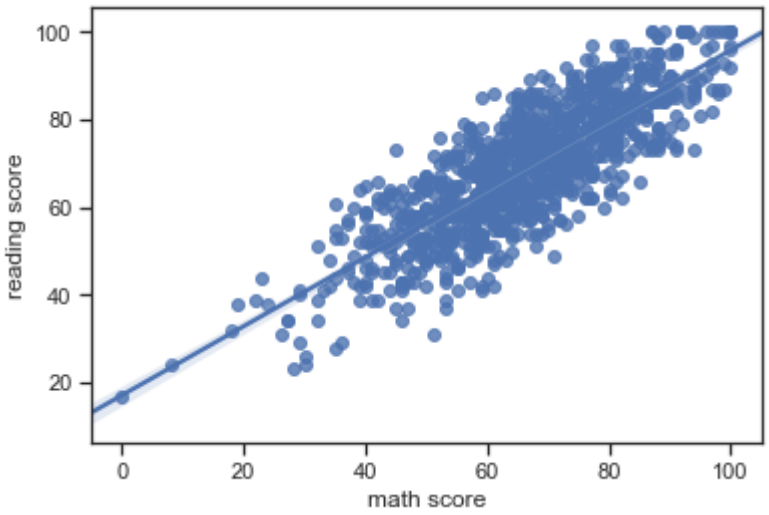
```
<matplotlib.axes._subplots.AxesSubplot at 0x13810f0>
```

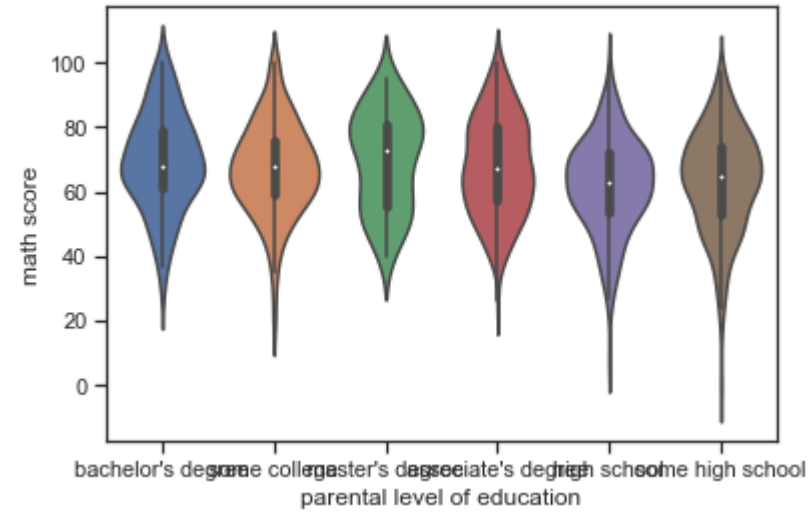


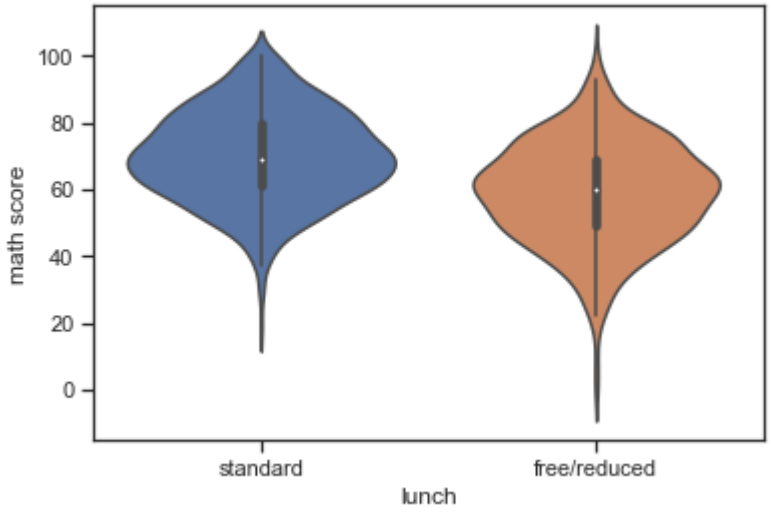



math score
reading score
writing score
0.025
0.020
0.015
0.010
0.005
0.000
0
20
40
60
80
100


writing score
math score
100
80
60
40
20
0
20
40
60
80
100


reading score
math score
100
80
60
40
20
0
20
40
60
80
100





Проведем обучение модели с различными вариантами, построим графики метрики качества модели и модель с использованием AutoML

```
def arr_to_df(arr_scaled):
    res = pd.DataFrame(arr_scaled, columns=student_MEAN_ENC.columns)
    return res
```

```
X_train, X_test, y_train, y_test = train_test_split(student_MEAN_ENC, student['gender'],
                                                    test_size=0.2,
                                                    random_state=1)
# Преобразуем массивы в DataFrame
X_train_df = arr_to_df(X_train)
X_test_df = arr_to_df(X_test)

X_train_df.shape, X_test_df.shape
```

```
((800, 7), (200, 7))
```

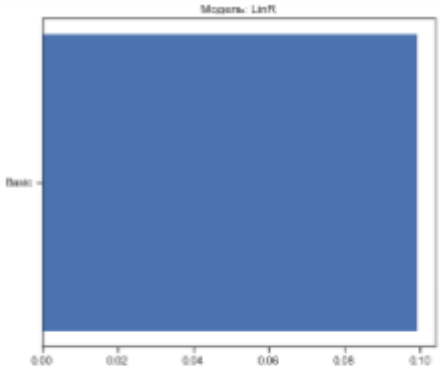
```
class MetricLogger:

    def __init__(self):
        self.df = pd.DataFrame(
            {'metric': pd.Series([], dtype='str'),
            'alg': pd.Series([], dtype='str'),
            'value': pd.Series([], dtype='float')})

    def add(self, metric, alg, value):
        """
        Добавление значения
        """
        # Удаление значения если оно уже было ранее добавлено
        self.df.drop(self.df[(self.df['metric']==metric)&(self.df['alg']==alg)].index, inplace = True)
        # Добавление нового значения
        temp = [{'metric':metric, 'alg':alg, 'value':value}]
        self.df = self.df.append(temp, ignore_index=True)

    def get_data_for_metric(self, metric, ascending=True):
        """
        Формирование данных с фильтром по метрике
        """
        temp_data = self.df[self.df['metric']==metric]
        temp_data_2 = temp_data.sort_values(by='value', ascending=ascending)
        return temp_data_2['alg'].values, temp_data_2['value'].values

    def plot(self, str_header, metric, ascending=True, figsize=(5, 5)):
        """
        Вывод графика
        """
        array_labels, array_metric = self.get_data_for_metric(metric, ascending)
        fig, ax1 = plt.subplots(figsize=figsize)
        pos = np.arange(len(array_metric))
        rects = ax1.barh(pos, array_metric,
                         align='center',
                         height=0.5,
                         tick_label=array_labels)
        ax1.set_title(str_header)
        for a,b in zip(pos, array_metric):
            plt.text(0.5, a-0.05, str(round(b,3)), color='white')
        plt.show()
```

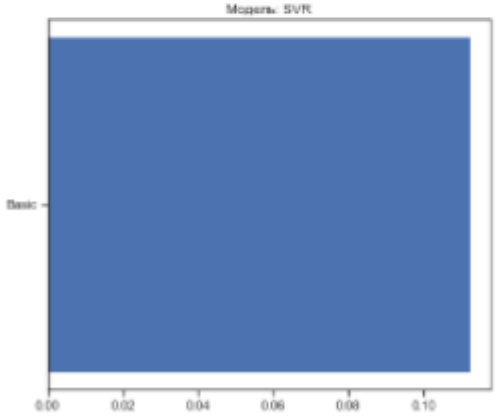
```
clas_models_dict = {'LinR': LinearRegression(),
                    'SVR': SVR(),
                    'KNN_5':KNeighborsRegressor(n_neighbors=5),
                    'Tree':DecisionTreeRegressor(random_state=1),
                    'GB': GradientBoostingRegressor(random_state=1),
                    'RF':RandomForestRegressor(n_estimators=50, random_state=1)}
```

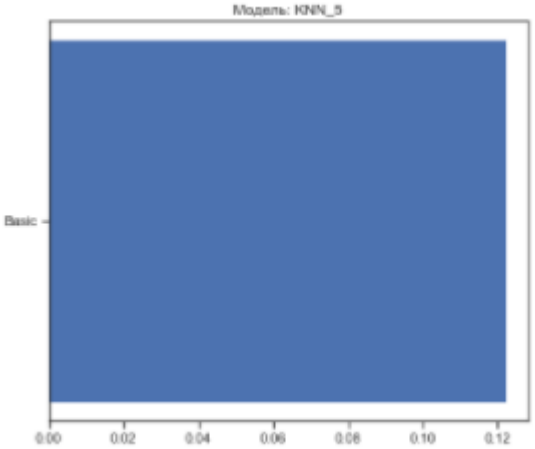
```
X_data_dict = {'Basic': (X_train_df, X_test_df)}
```

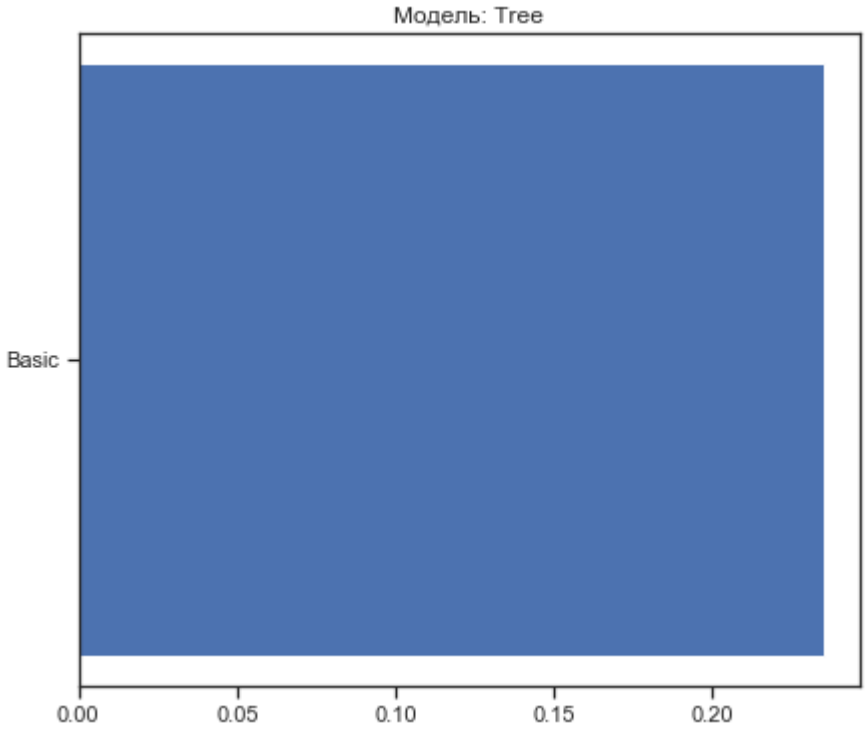
```
def test_models(clas_models_dict, X_train, X_test, y_train, y_test):

    logger = MetricLogger()

    for model_name, model in clas_models_dict.items():
        model.fit(X_train, y_train)
        y_pred = model.predict(X_test)
        mse = mean_squared_error(y_test, y_pred)
        logger.add(model_name, 'Basic', mse)

    return logger
```

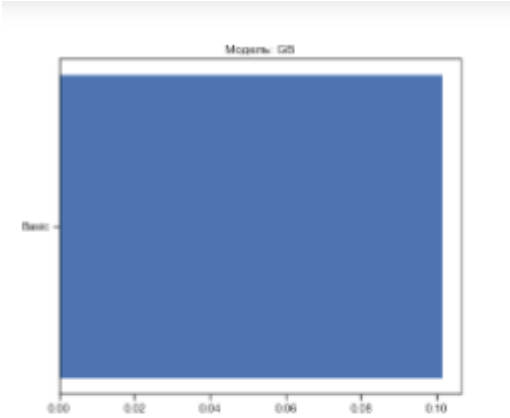
```
logger = test_models(clas_models_dict, X_train_df, X_test_df, y_train, y_test)
```

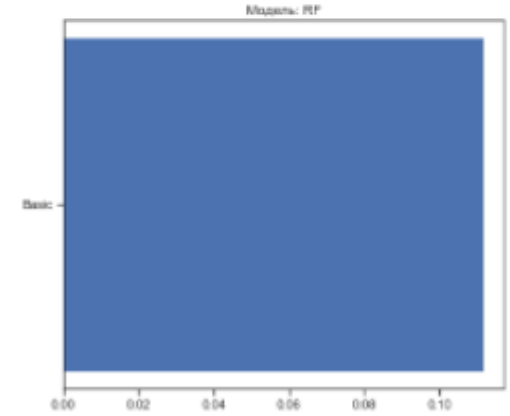





Модель: KNN_5
Basic
0.00
0.02
0.04
0.06
0.08
0.10
0.12


Модель: Tree
Basic
0.00
0.05
0.10
0.15
0.20






```
automl = AutoML()
```

```
automl.fit(train[train.columns[2:-3]], train['math score'])
```

```
AutoML directory: AutoML_1
The task is regression with evaluation metric rmse
AutoML will use algorithms: ['Baseline', 'Linear', 'Decision Tree', 'Random Forest', 'Xgboost', 'Neural Network']
AutoML will ensemble availabe models
AutoML steps: ['simple_algorithms', 'default_algorithms', 'ensemble']
* Step simple_algorithms will try to check up to 3 models
1_Baseline rmse 30.041484 trained in 0.42 seconds
2_DecisionTree rmse 9.566765 trained in 13.15 seconds
3_Linear rmse 36.208657 trained in 3.66 seconds
* Step default_algorithms will try to check up to 3 models
4_Default_Xgboost rmse 1.103866 trained in 7.6 seconds
5_Default_NeuralNetwork rmse 2.882078 trained in 1.61 seconds
6_Default_RandomForest rmse 6.037609 trained in 9.99 seconds
* Step ensemble will try to check up to 1 model
Ensemble rmse 1.103866 trained in 0.37 seconds
/root/.local/lib/python3.7/site-packages/numpy/lib/function_base.py:2642: RuntimeWarning:

invalid value encountered in true_divide

/root/.local/lib/python3.7/site-packages/numpy/lib/function_base.py:2643: RuntimeWarning:

invalid value encountered in true_divide

AutoML fit time: 49.05 seconds
AutoML best model: 4_Default_Xgboost
AutoML()
```