МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по индивидуальному домашнему заданию

по дисциплине «Машинное обучение»

ТЕМА: Задача регрессии для предсказания оценок студентов

Студенты гр. 6307 Кичерова А. Д.

Медведев Е. Р.

Преподаватель Жангиров Т. Р.

Санкт-Петербург

2020

Цель работы

Разработать модель МО для решения задачи предсказания оценки студента на основе информации о нем методом регрессии.

Ход работы

Описание датасета

Данные загружены из csv файла, столбцы, содержащие бинарные признаки и признаки времени, исключены.

```
[30]: data = pd.read_csv('student-mat.csv')
      data
```

[30]:

	school	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	Fjob	...	famrel
0	GP	F	18	U	GT3	A	4	4	at_home	teacher	...	4
1	GP	F	17	U	GT3	T	1	1	at_home	other	...	5
2	GP	F	15	U	LE3	T	1	1	at_home	other	...	4
3	GP	F	15	U	GT3	T	4	2	health	services	...	3
4	GP	F	16	U	GT3	T	3	3	other	other	...	4
...	...	...	...	...	...	...	...	...	...	...	...	...
390	MS	M	20	U	LE3	A	2	2	services	services	...	5
391	MS	M	17	U	LE3	T	3	1	services	services	...	2
392	MS	M	21	R	GT3	T	1	1	other	other	...	5
393	MS	M	18	R	LE3	T	3	2	services	other	...	4
394	MS	M	19	U	LE3	T	1	1	other	at_home	...	3

395 rows × 33 columns

Рисунок 1 – Данные в датасете

В датасете приведено 395 наблюдений с 33 переменными. Каждая строка представляет собой учащегося, а каждый столбец содержит характеристику.

Признаки, содержащиеся в датасете:

school - школа ученика (binary: GP - Габриэль Перейра или MS - Мусиньо да Силвейра)

sex - пол ученика (binary: 'F' - женский или 'M' - мужской)

age - возраст студента (numeric: от 15 до 22)

address - тип домашнего адреса студента (binary: 'U' - городской или 'R' - сельский)

famsize - размер семьи (binary: 'LE3' - меньше или равно 3 или 'GT3' - больше 3)

Pstatus - статус сожительства родителей (binary: «T» - проживают вместе или «A» - отдельно)

Medu - образование матери (numeric: 0 - нет, 1 - начальное образование (4-й класс), 2 - 5-9 классы, 3 - среднее образование или 4 - высшее образование)

Fedu - образование отца (numeric: 0 - нет, 1 - начальное образование (4 класс), 2–5–9 классы, 3 - среднее образование или 4 - высшее образование)

Mjob - работа матери (nominal: «учитель», «медицинское обслуживание», гражданские «службы» (например, административные или полицейские), «at_home» или «другое»)

Fjob - работа отца (nominal: «учитель», «медицинское обслуживание», гражданские «услуги» (например, административные или полицейские), «at_home» или «другое»)

reason - причина выбрать эту школу (nominal: близко к «дому», «репутация» школы, «предпочтение по курсу» или «другое»)

guardian - опекун ученика (nominal: «мать», «отец» или «другой»)

время в пути - время в пути от дома до школы (numeric: 1–1 час)

Studytime - еженедельное учебное время (numeric: 1-10 часов)

сбои - количество прошлых сбоев класса (numeric: n, если 1 <= n <3, иначе 4)

Schoolup - дополнительная образовательная поддержка (binary: да или нет)

famsup - семейная образовательная поддержка (binary: да или нет)

paid - дополнительные платные занятия по предмету курса (математика или португальский) (binary: да или нет)

activities - внеклассные мероприятия (binary: да или нет)

nursery - посещал детский сад (binary: да или нет)

higher - хочет получить высшее образование (двоичное: да или нет)

internet - доступ в Интернет дома (binary: да или нет)

romantic - с романтическими отношениями (binary: да или нет)

famrel - качество семейных отношений (numeric: от 1 - очень плохо до 5 - отлично)

freetime - свободное время после школы (numeric: от 1 - очень мало до 5 - очень высоко)

goout - встреча с друзьями (numeric: от 1 - очень низкий до 5 - очень высокий)

Dalc - потребление алкоголя в течение рабочего дня (numeric: от 1 - очень низкий до 5 - очень высокий)

Walc - потребление алкоголя в выходные дни (numeric: от 1 - очень низкий до 5 - очень высокий)

health - текущее состояние здоровья (numeric: от 1 - очень плохо до 5 - очень хорошо)

absences - количество пропусков в школе (numeric: от 0 до 93)

G1 – оценка за первый семестр (numeric: от 0 до 20)

G2 - оценка за второй семестр (numeric: от 0 до 20)

G3 - итоговая оценка (numeric: от 0 до 20, выходное значение)

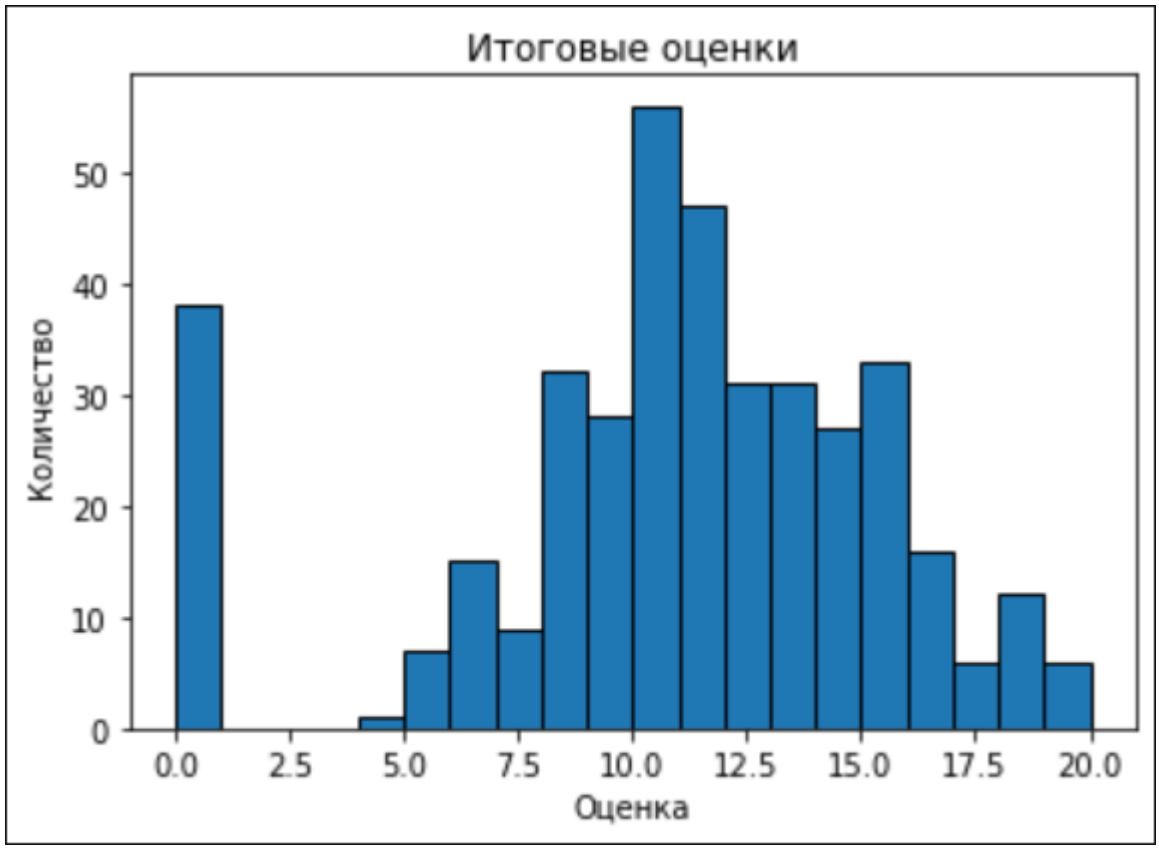

Рисунок 2 – Распределение итоговых оценок студентов

За исключением большого числа студентов, получивших 0 баллов, распределение нормальное. Возможно 0 использовался для тех, кто не пришел на экзамен или вместо отсутствия данных. Нужно проверить, есть ли пропущенные переменные.

```
Your selected dataframe has 33 columns.
There are 0 columns that have missing values.
[180]:    Missing Values  % of Total Values
```

Рисунок 3 – Проверка данных

Пустых столбцов в датафрейме не обнаружено.

Выбор опорных признаков

Для выбора признаков, наиболее влияющих на итоговую оценку, был произведен анализ корреляции признаков.

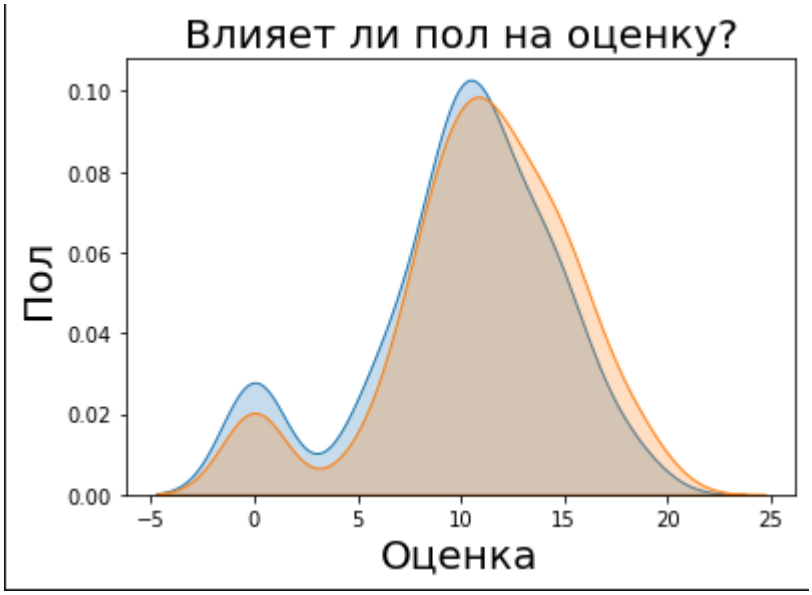

Рисунок 4 – График влияния пола на оценку

График показывает, что нет сильных различий между оценками девушек и парней.

Рисунок 5 – График влияния школы студента на оценку

Школа, которую посещал студент тоже не сильно влияет на итоговую оценку.

Таким же образом проанализированы следующие признаки:

Влияет ли дополнительная образовательная поддержка на оценку?

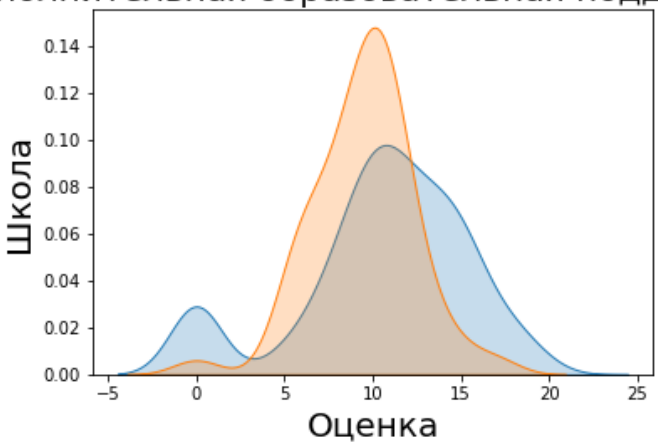

Рисунок 6 – График влияния школы

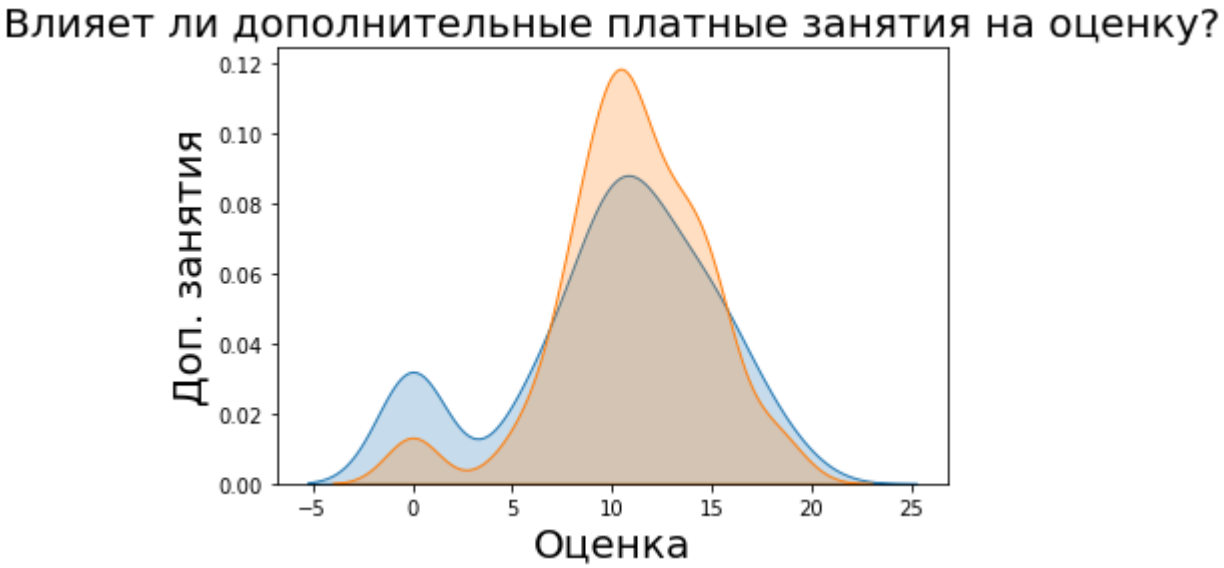

Рисунок 7 – График влияния посещения платных занятий на оценку

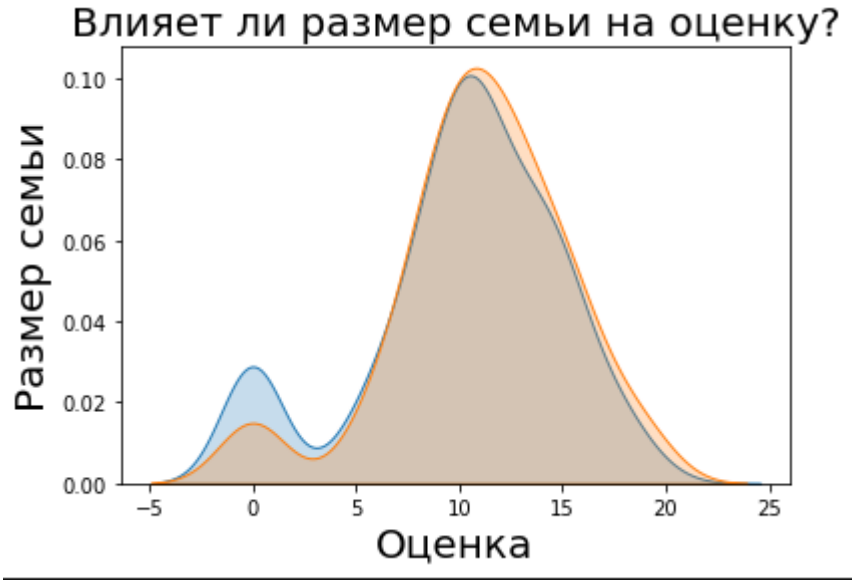

Рисунок 8 – График влияния размера семьи на оценку

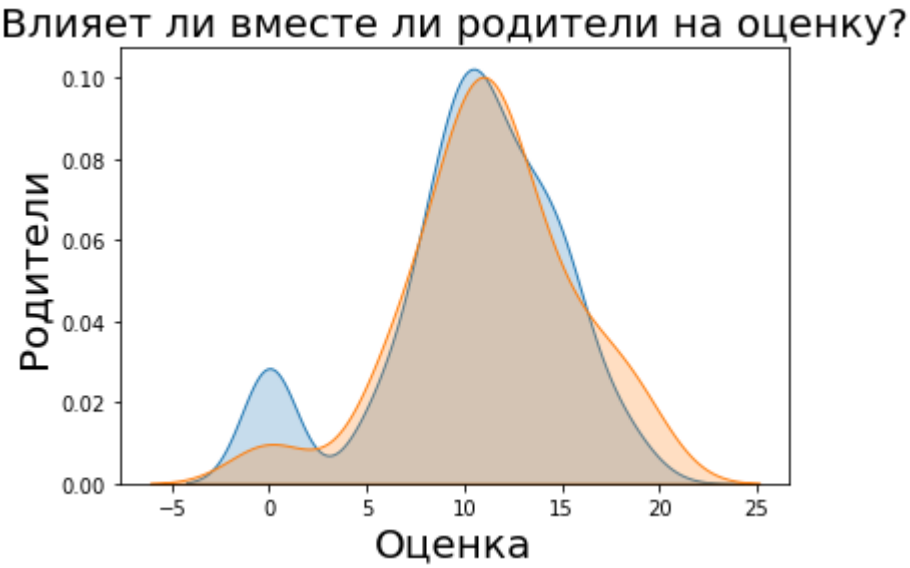

Рисунок 9 – График влияния отношений родителей на оценку

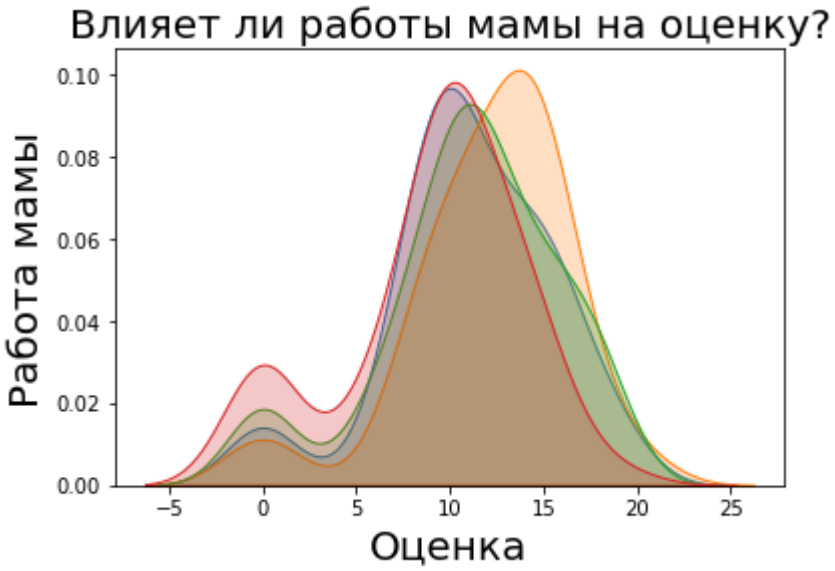

Рисунок 10 - График влияния работы мамы на оценку

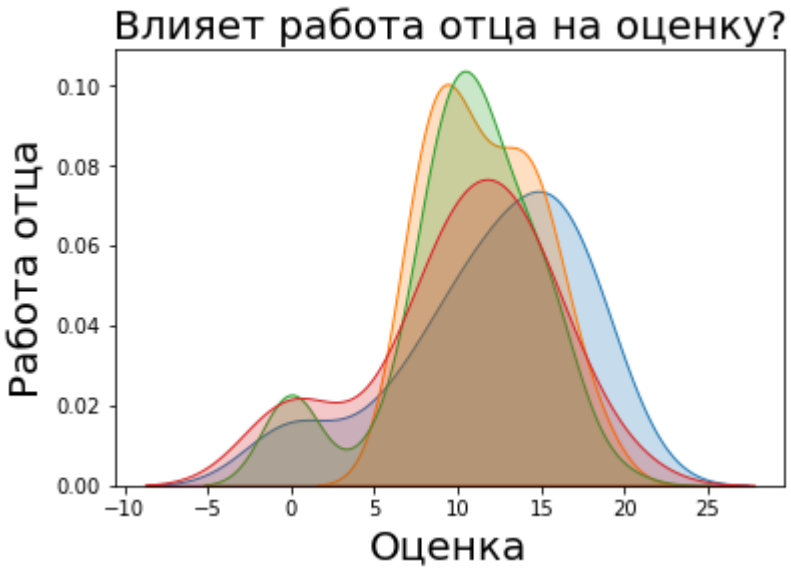

Рисунок 11 – График влияния работы отца на оценку

Таким образом, вышеперечисленные признаки не несут значимого влияния на итоговую оценку. По признакам, имеющим более высокую корреляцию с параметром G3 составлена корреляционная матрица:

[45]:

	age	Medu	Fedu	traveltime	studytime	failures	goout	Dalc	Walc	health	absences	G1	G2	G3
age	1.000000	-0.163658	-0.163438	0.070641	-0.004140	0.243665	0.126964	0.131125	0.117276	-0.062187	0.175230	-0.064081	-0.143474	-0.161579
Medu	-0.163658	1.000000	0.623455	-0.171639	0.064944	-0.236680	0.064094	0.019834	-0.047123	-0.046878	0.100285	0.205341	0.215527	0.217147
Fedu	-0.163438	0.623455	1.000000	-0.158194	-0.009175	-0.250408	0.043105	0.002386	-0.012631	0.014742	0.024473	0.190270	0.164893	0.152457
traveltime	0.070641	-0.171639	-0.158194	1.000000	-0.100909	0.092239	0.028540	0.138325	0.134116	0.007501	-0.012944	-0.093040	-0.153198	-0.117142
studytime	-0.004140	0.064944	-0.009175	-0.100909	1.000000	-0.173563	-0.063904	-0.196019	-0.253785	-0.075616	-0.062700	0.160612	0.135880	0.097820
failures	0.243665	-0.236680	-0.250408	0.092239	-0.173563	1.000000	0.124561	0.136047	0.141962	0.065827	0.063726	-0.354718	-0.355896	-0.360415
goout	0.126964	0.064094	0.043105	0.028540	-0.063904	0.124561	1.000000	0.266994	0.420386	-0.009577	0.044302	-0.149104	-0.162250	-0.132791
Dalc	0.131125	0.019834	0.002386	0.138325	-0.196019	0.136047	0.266994	1.000000	0.647544	0.077180	0.111908	-0.094159	-0.064120	-0.054660
Walc	0.117276	-0.047123	-0.012631	0.134116	-0.253785	0.141962	0.420386	0.647544	1.000000	0.092476	0.136291	-0.126179	-0.084927	-0.051939
health	-0.062187	-0.046878	0.014742	0.007501	-0.075616	0.065827	-0.009577	0.077180	0.092476	1.000000	-0.029937	-0.073172	-0.097720	-0.061335
absences	0.175230	0.100285	0.024473	-0.012944	-0.062700	0.063726	0.044302	0.111908	0.136291	-0.029937	1.000000	-0.031003	-0.031777	0.034247
G1	-0.064081	0.205341	0.190270	-0.093040	0.160612	-0.354718	-0.149104	-0.094159	-0.126179	-0.073172	-0.031003	1.000000	0.852118	0.801468
G2	-0.143474	0.215527	0.164893	-0.153198	0.135880	-0.355896	-0.162250	-0.064120	-0.084927	-0.097720	-0.031777	0.852118	1.000000	0.904868
G3	-0.161579	0.217147	0.152457	-0.117142	0.097820	-0.360415	-0.132791	-0.054660	-0.051939	-0.061335	0.034247	0.801468	0.904868	1.000000

Рисунок 12 – Корреляционная матрица признаков

А также ее графическое представление:

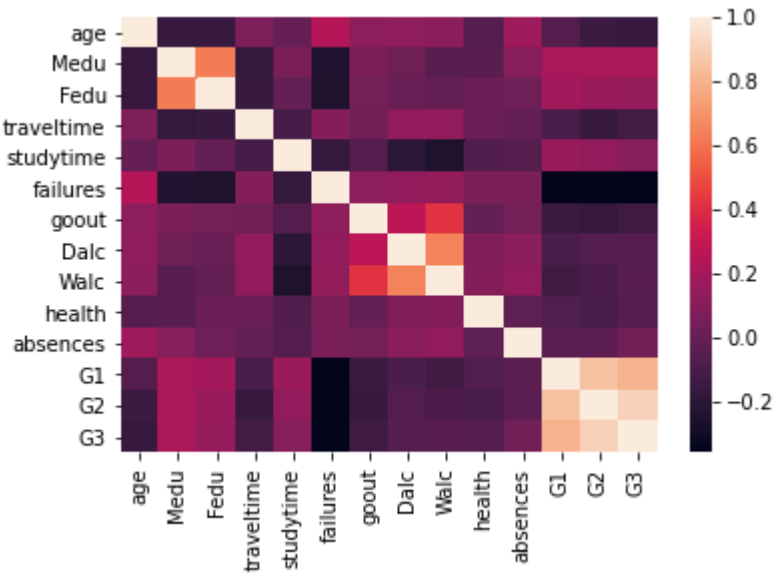

Рисунок 13 – Графическое представление корреляции признаков

Видно, что наибольшую корреляцию итоговая оценка, помимо оценок за 1 и 2 периоды, имеет с образованием родителей и временем, потраченным на учебу, возрастом, временем на путешествия, уровень здоровья и т.д. Для регрессии будем использовать данные, которые наиболее сильно коррелируют с G3.

Анализ моделей МО

Были рассмотрены следующие модели регрессии: LinearRegression, Ridge, KNeighborsRegressor, SGDRegressor, GradientBoostingRegressor, RandomForestRegressor, DecisionTreeRegressor.

Датасет был разбит на обучающее и тестовое подмножество. Размер тестового подмножества составляет 20% датасета, обучающего – 80%.

Результаты оценки обучения выбранными моделями представлены в таблице:

[187]:

	name	MSE	r2
4	RandomForestRegressor	0.114888	0.89564
6	LinearRegression	0.13348	0.878751
0	Ridge	0.134954	0.877412
2	SGDRegressor	0.135146	0.877238
3	GradientBoostingRegressor	0.160925	0.853821
5	DecisionTreeRegressor	0.166114	0.849108
1	KNeighborsRegressor	0.475271	0.568281

Рисунок 14 – Сравнение результатов обучения

Проведена проверка моделей на разных наборах тестовых и обучающих данных в серии из 10 тестов, где результаты вычислялись как средние значения показателей MSE и R2:

[189]:

	name	MSE	r2
4	RandomForestRegressor	0.129783	0.87087
6	LinearRegression	0.180791	0.818698
0	Ridge	0.180993	0.81853
2	SGDRegressor	0.187969	0.81122
3	GradientBoostingRegressor	0.154543	0.845122
5	DecisionTreeRegressor	0.20626	0.792933
1	KNeighborsRegressor	0.530036	0.469126

Рисунок 15 – Сравнение моделей на 10 тестах

Видно, что качество обучения осталось прежним независимо от входного набора данных.

В ходе исследования результатов работы моделей была выбрана модель, обученная RandomForestRegressor.

На рис. 16 представлены оценки студентов, предсказанные моделью, обученной RandomForestRegressor оранжевым цветом и реальные оценки студентов синим цветом. Можно заметить, что предсказанные оценки близки к истинным значениям, что означает хорошую применимость машинного обучения для решения данной задачи.

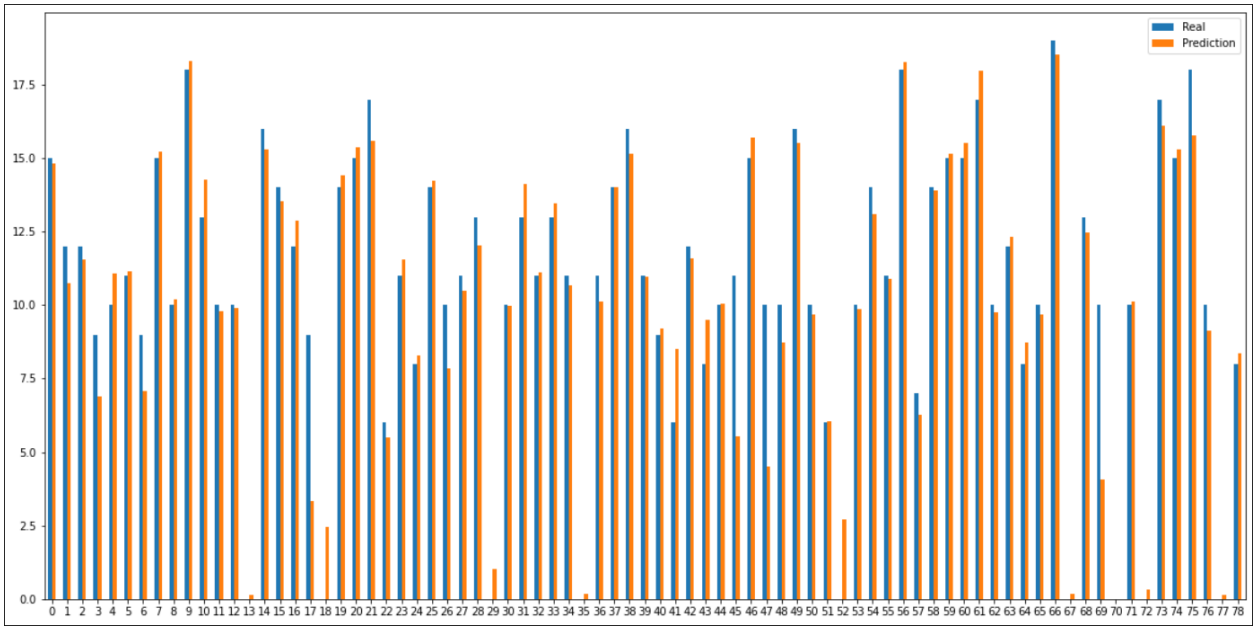

Рисунок 16 – Предсказание оценок студентов

Итоги

В данной работе была решена задача предсказания оценок. В ходе работы произведен анализ корреляции, по итогам которого был построен датасет с данными, которые коррелируют друг с другом. В результате из всех данных были выбраны те, которые играют важную роль в формировании оценки обучающегося.

Были исследованы несколько методов регрессии. Наилучший результат показал RandomForestRegressor, хотя другие регрессоры показали результат не сильно хуже.

Представлена визуализация результата, на которой видна, что регрессия дала отличные результаты: разница между реальными и истинными значениями незначительная

Проблемы, возникшие во время работы:

1. Был произведен не очень хороший анализ данных и в качестве параметров были взяты только сильно коррелирующие величины. Мы это поняли по тому, что на обучающих данных модель выдавала правильность всего в 60-70 %. Для решения этой проблемы количество параметров было увеличено, а также к ним были добавлены нечисловые параметры, что позволило добиться хороших результатов в обучении модели. Результаты сравнимы с теми, которые были бы, если бы для обучения были взяты все параметры
2. Проблема выбора регрессии из RandomForestRegressor и GradientBoostingRegressor из-за похожих результатов. Проблема была решена с помощью прогонки на нескольких разбивках на тестовое и обучающее подмножество. После нескольких прогонов стало понятно, что GradientBoostingRegressor менее хорошо себя показывает на нашей модели

Код программы

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns


from sklearn.linear_model import LinearRegression, RidgeCV, SGDRegressor,
LassoCV, Ridge

from sklearn.tree import DecisionTreeRegressor

from sklearn.neighbors import KNeighborsRegressor

from sklearn.ensemble import GradientBoostingRegressor,
RandomForestRegressor

from sklearn.model_selection import train_test_split

from sklearn import metrics

from sklearn.preprocessing import StandardScaler


data = pd.read_csv('student-mat.csv')


for col in data.columns:
    if data[col].dtype == 'object':
        print('\nColumn Name:', col,)
        print(data[col].value_counts())


data.describe()


plt.hist(data['G3'], bins=20,edgecolor = 'k', range=[0,20])
```

```
plt.xlabel('Оценка'); plt.ylabel('Количество'); plt.title('Итоговые оценки');

def missing_values_table(df):
        # Total missing values
        mis_val = df.isnull().sum()

        # Percentage of missing values
        mis_val_percent = 100 * df.isnull().sum() / len(df)

        # Make a table with the results
        mis_val_table = pd.concat([mis_val, mis_val_percent], axis=1)

        # Rename the columns
        mis_val_table_ren_columns = mis_val_table.rename(
        columns = {0 : 'Missing Values', 1 : '% of Total Values'})

        # Sort the table by percentage of missing descending
        mis_val_table_ren_columns = mis_val_table_ren_columns[
            mis_val_table_ren_columns.iloc[:,1] != 0].sort_values(
        '% of Total Values', ascending=False).round(1)

        # Print some summary information
        print ("Your selected dataframe has " + str(df.shape[1]) + " columns.\n"
            "There are " + str(mis_val_table_ren_columns.shape[0]) +
              " columns that have missing values.")
```

```
        # Return the dataframe with missing information
        return mis_val_table_ren_columns

missing_values_table(data)

sns.kdeplot(data.loc[data['sex'] == 'F', 'G3'], label='Женский', shade = True)
sns.kdeplot(data.loc[data['sex'] == 'M', 'G3'], label='Мужской', shade = True)
plt.title('Влияет ли пол на оценку?', fontsize = 20)
plt.xlabel('Оценка', fontsize = 20);
plt.ylabel('Пол', fontsize = 20)
plt.show()

sns.kdeplot(data.loc[data['school'] == 'GP', 'G3'], label='Gabriel Pereira', shade = True)
sns.kdeplot(data.loc[data['school'] == 'MS', 'G3'], label='Mousinho', shade = True)
plt.title('Влияет ли школа на оценку?', fontsize = 20)
plt.xlabel('Оценка', fontsize = 20);
plt.ylabel('Школа', fontsize = 20)
plt.show()

sns.kdeplot(data.loc[data['paid'] == 'no', 'G3'], label='нет', shade = True)
sns.kdeplot(data.loc[data['paid'] == 'yes', 'G3'], label='да', shade = True)
plt.title('Влияет ли дополнительные платные занятия на оценку?', fontsize = 20)
plt.xlabel('Оценка', fontsize = 20);
```

```
plt.ylabel('Доп. занятия', fontsize = 20)

plt.show()


data_num = data[['age','Medu', 'Fedu', 'traveltime', 'studytime', 'failures',
'goout','Dalc', 'Walc', 'health', 'absences', "G1","G2", "G3"]]

data_num.corr()


sns.heatmap(data_num.corr())


features = data_2.copy()

categorical_subset = pd.get_dummies(data_obj)

features = pd.concat([data_2, categorical_subset], axis = 1)


scaler = StandardScaler().fit(features)

features = pd.DataFrame(scaler.transform(features), columns=features.columns,
index=features.index)

features


x = np.array(features.drop([predict], axis=1))

y = np.array(features[predict])


X_train, X_test, Y_train, Y_test = train_test_split(x, y, test_size=0.2, random_state
= 46)


regressions = pd.DataFrame(columns = ["name",'func','MSE','r2'])

linear = LinearRegression()
```

```
linear.fit(X_train, Y_train)

predicted = linear.predict(X_test)

print ("MSE :", metrics.mean_squared_error(Y_test,predicted))

print("Правильность на обучающем наборе: {:.5f}".format(linear.score(X_train, Y_train)))

print("Правильность на тестовом наборе: {:.5f}".format(linear.score(X_test, Y_test)))


def regression(name,func,regressions, i):
    func.fit(X_train, Y_train)
    predicted = func.predict(X_test)
    regressions.loc[i, 'name'] = name
    regressions.loc[i, 'func'] = func
    regressions.loc[i, 'MSE'] = metrics.mean_squared_error(Y_test,predicted)
    regressions.loc[i, 'r2'] = func.score(X_test, Y_test)


regression("Ridge",Ridge(),regressions,0)

regression("KNeighborsRegressor",KNeighborsRegressor(n_neighbors=10, weights="distance"),regressions,1)

regression("SGDRegressor", SGDRegressor(),regressions,2)

regression("GradientBoostingRegressor", GradientBoostingRegressor(random_state=0, n_estimators=150),regressions,3)

regression("RandomForestRegressor",RandomForestRegressor(random_state=0, n_estimators=50),regressions,4)

regression("DecisionTreeRegressor",DecisionTreeRegressor(random_state=0, criterion="mae", max_depth=4),regressions,5)


best = regressions.sort_values(by='r2', ascending=False).head(1)
```

```
best.drop(columns=["func"])


def check_regression_n_times(func, N):
    r2 = 0
    MSE = 0


    for i in range(N):
        X_train, X_test, Y_train, Y_test = train_test_split(x, y, test_size=0.2,
random_state = 15+i)
        func.fit(X_train, Y_train)
        predicted = func.predict(X_test)
        MSE +=  metrics.mean_squared_error(Y_test,predicted)
        r2 += func.score(X_test, Y_test)


    best.loc[best['func'] == func, 'MSE'] = MSE/N
    best.loc[best['func'] == func, 'r2'] = r2/N


for func in best['func']:
    check_regression_n_times(func, 10)
best.drop(columns=["func"])


rfr = RandomForestRegressor(random_state=0, n_estimators=50)
rfr.fit(X_train, Y_train)
predicted = rfr.predict(X_test)
comparison = pd.DataFrame(columns=["Real","Prediction"])
```

```
f = features.drop([predict], axis=1)


tmp_df = pd.DataFrame(X_test, columns=f.columns)

tmp_df["G3"] = predicted

tmp_df = tmp_df.reindex(columns=features.columns)

tmp_df = pd.DataFrame(scaler.inverse_transform(tmp_df),
columns=tmp_df.columns, index=tmp_df.index)

comparison["Prediction"] = tmp_df["G3"]



tmp_df = pd.DataFrame(X_test, columns=f.columns)

tmp_df["G3"] = Y_test

tmp_df = tmp_df.reindex(columns=features.columns)

tmp_df = pd.DataFrame(scaler.inverse_transform(tmp_df),
columns=tmp_df.columns, index=tmp_df.index)

comparison["Real"] = tmp_df["G3"]


ax = comparison.plot.bar(rot=0, figsize=(20,10))
```