**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.

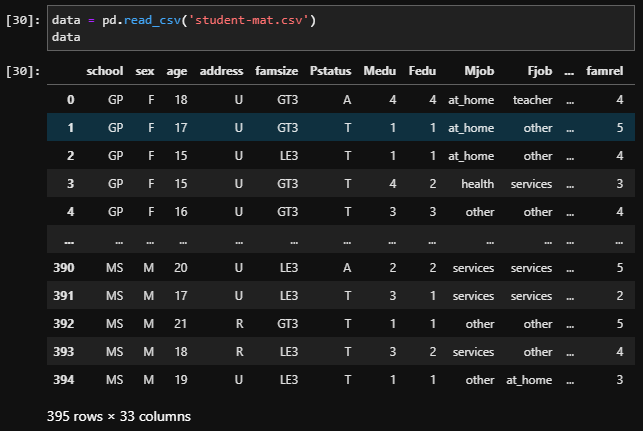


Рисунок 1 – Данные в датасете

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)

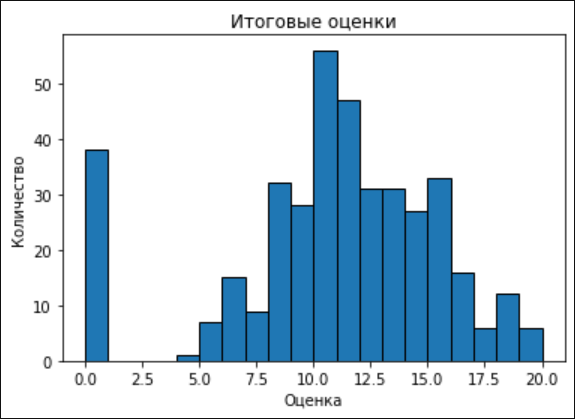


Рисунок 2 – Распределение итоговых оценок студентов

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.

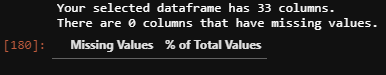


Рисунок 3 – Проверка данных

Пустых столбцов в датафрейме не обнаружено.

## Выбор опорных признаков

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

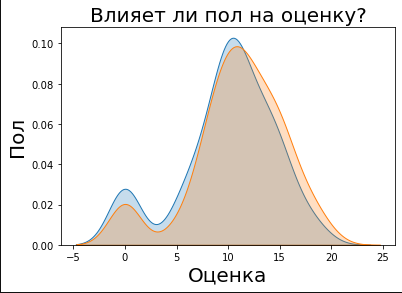


Рисунок 4 – График влияния пола на оценку

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

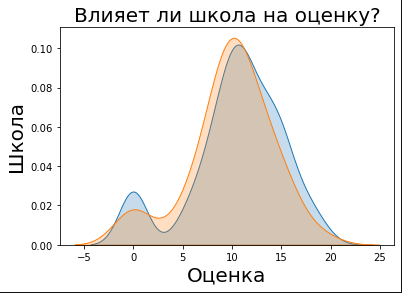


Рисунок 5 – График влияния школы студента на оценку

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

Таким же образом проанализированы следующие признаки:

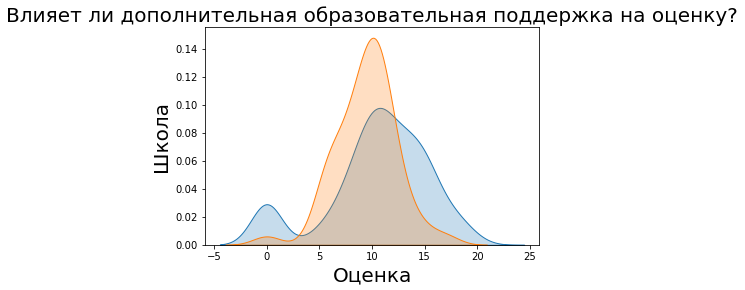


Рисунок 6 – График влияния школы

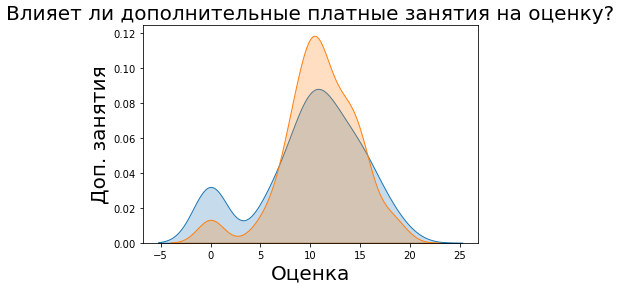


Рисунок 7 – График влияния посещения платных занятий на оценку

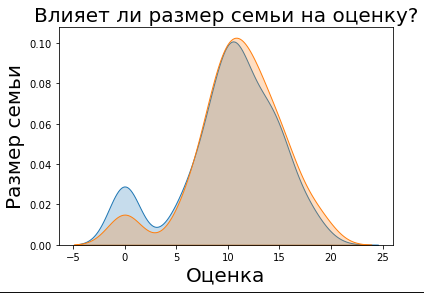


Рисунок 8 – График влияния размера семьи на оценку

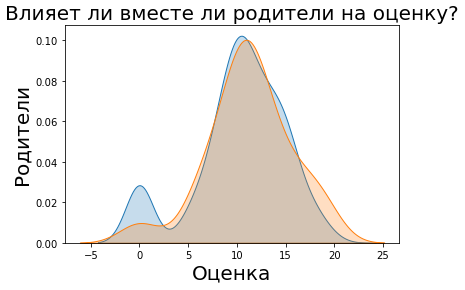


Рисунок 9 – График влияния отношений родителей на оценку

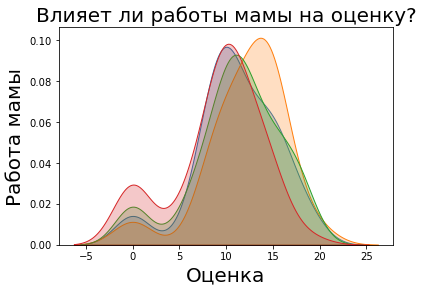


Рисунок 10 - График влияния работы мамы на оценку

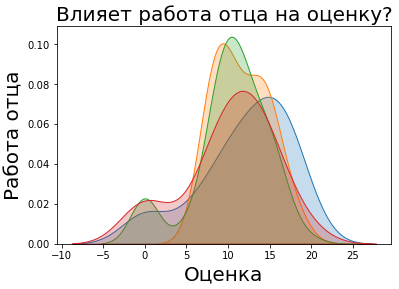


Рисунок 11 – График влияния работы отца на оценку

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

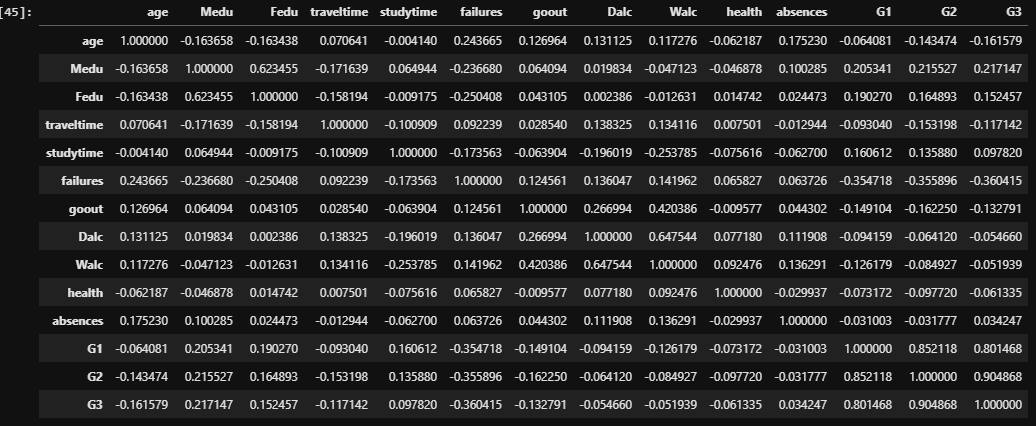


Рисунок 12 – Корреляционная матрица признаков

А также ее графическое представление:

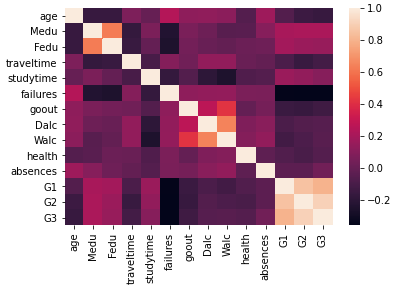


Рисунок 13 – Графическое представление корреляции признаков

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:

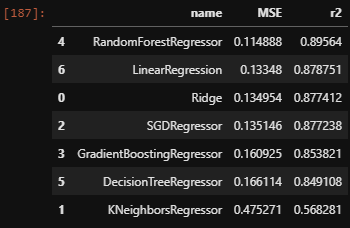


Рисунок 14 – Сравнение результатов обучения

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:



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.

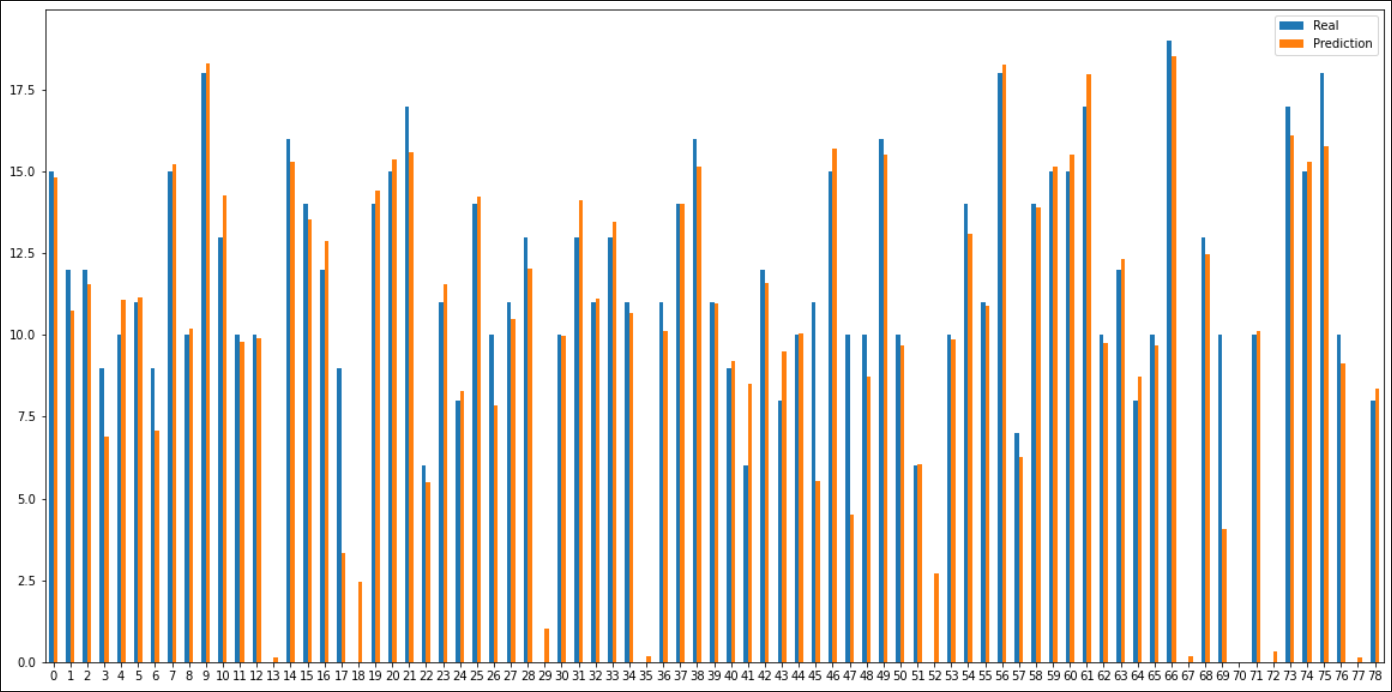


Рисунок 16 – Предсказание оценок студентов

# 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))