Shashank Mishra

6/27/2021

Doctor Fees prediction



**Table of contents-:**

* **Problem definition**
* **Data analysis**
* **Exploratory data analysis**
* **Data pre-processing**
* **Model building**
* **Conclusion**

**Problem Definition-:**

We have all been in situation where we go to a doctor in emergency and find that the consultation fees are too high, in that situation we questioned ourselves the most that weather doctor is more qualified or he/she is matter of expert, or how people rated this doctor.

You probably in question sometimes when you find high prices for the doctor and rating and qualification is not more.

The solution can be having the model/software at the end which actually can predict the basic fees of the doctor.

Basically, by the dataset we are able to predict the valid doctor fees which he/she must charge according to the dataset there are 7 features in which one is “Fees” which is target variable and the feature are-:

* Qualification: Qualification and degrees held by the doctor
* Experience: Experience of the doctor in number of years
* Rating: Rating given by patients
* Profile: Type of the doctor
* Miscellaneous Info: Extra information about the doctor
* Fees: Fees charged by the doctor (Target Variable)
* Place: Area and the city where the doctor is located

**Data Analysis-:**

**Importing libraries**-:

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

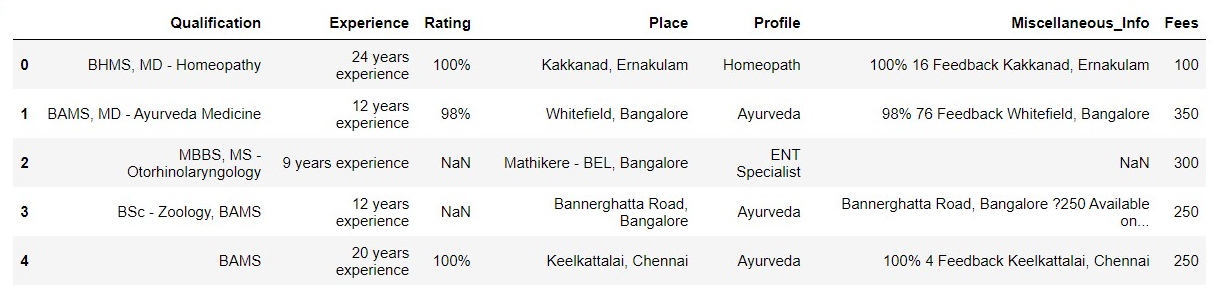
import seaborn as sns

import warnings

warnings.filterwarnings('ignore')

**Importing dataset -:**

fee = pd.read\_csv('fees\_data.csv')



**analysing the data-:**

fee.info()

**output-:** <class 'pandas.core.frame.DataFrame'>

RangeIndex: 5961 entries, 0 to 5960

Data columns (total 7 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 Qualification 5961 non-null object

1 Experience 5961 non-null object

2 Rating 2659 non-null object

3 Place 5936 non-null object

4 Profile 5961 non-null object

5 Miscellaneous\_Info 3341 non-null object

6 Fees 5961 non-null int64

dtypes: int64(1), object(6)

memory usage: 326.1+ KB

**Observation-:**

**data is present in object format, we need to encode every column for the model training.**

**we have plenty of data in which plenty of qualifications, plenty of places, plenty of profile, plenty of feedbacks**

**checking unique variables-:**

for i in fee.columns:

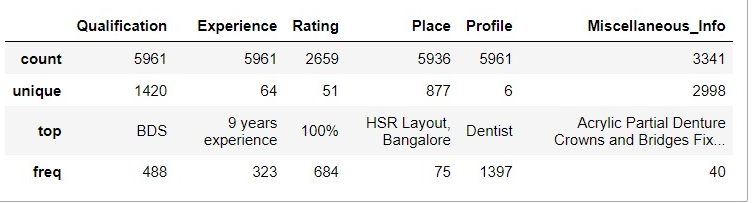
print(fee[i].unique())

print('\n')

this command will show you the unique variables for each columns and it would be more n more in number because the variables have objects in it and there are multiple and different objects.

You can use this command also to get the described table of object type variables-:

Fee.describe(include-‘O’)



Now, we going to deal with each variable as per the frequency/value counts.

**Data preprocessing pipeline-:**

1. **First we need to convert the experience column in the data set as it has string values.**

exp = []

for i in fee['Experience']:

exp.append(int(i.split()[0]))

assigning the extraced variable to the dataframe variable

fee['Experience'] = exp

1. Second we encode Profile column on one hot encoding method by pd.get\_dummies method

profile\_df = pd.get\_dummies(fee['Profile'])

then we need to concat the profile\_df dataframe to the fee dataframe.

fee = pd.concat([fee,profile\_df], axis =1 )

1. Now the rating column will be converted as it also contains string values in it.

test['Rating'] = test['Rating'].fillna('0')

rating = []

for i in fee['Rating']:

rating.append(int(i.split('%')[0])/10)

assigning the extracted rating values to the fee dataframe.

fee['Rating'] = rating

1. **The place column comes at this point for extracting the cities from the whole string value.**

fee['Place'] = fee['Place'].fillna('unknown,unknown')

place=[]

for i in fee['Place']:

place.append(i.split(',').pop())

assigning extracted values to the fee dataframe and replacing some unwanted values

fee['Place']=place

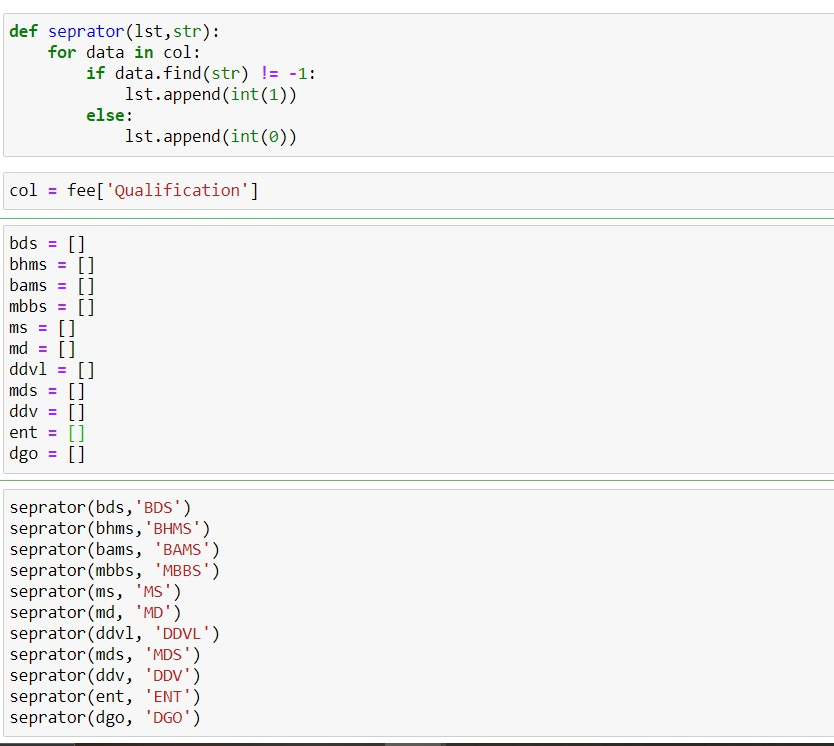
fee['Place'].replace(to\_replace='e',value='unknown',inplace=True)

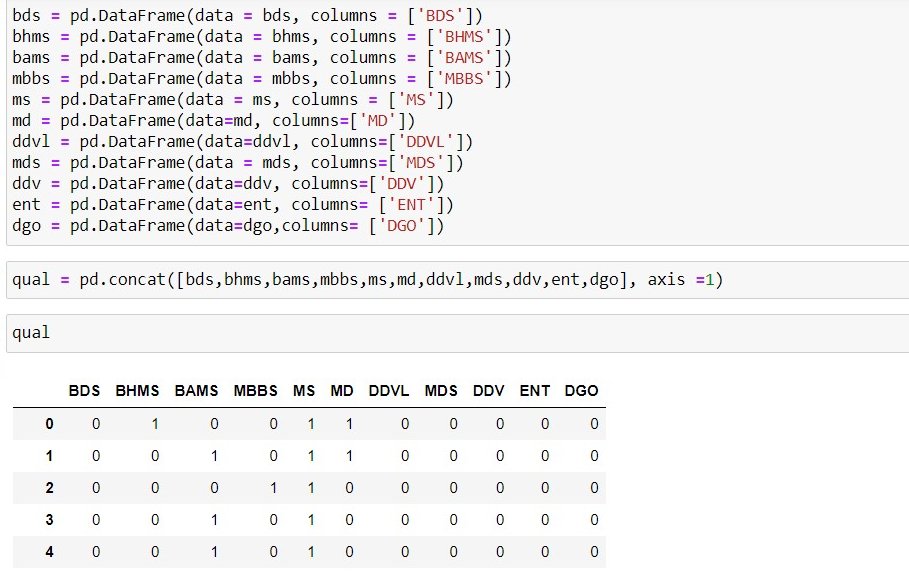
place\_ = pd.get\_dummies(fee['Place'])

fee\_new = pd.concat([fee,place\_], axis =1 )

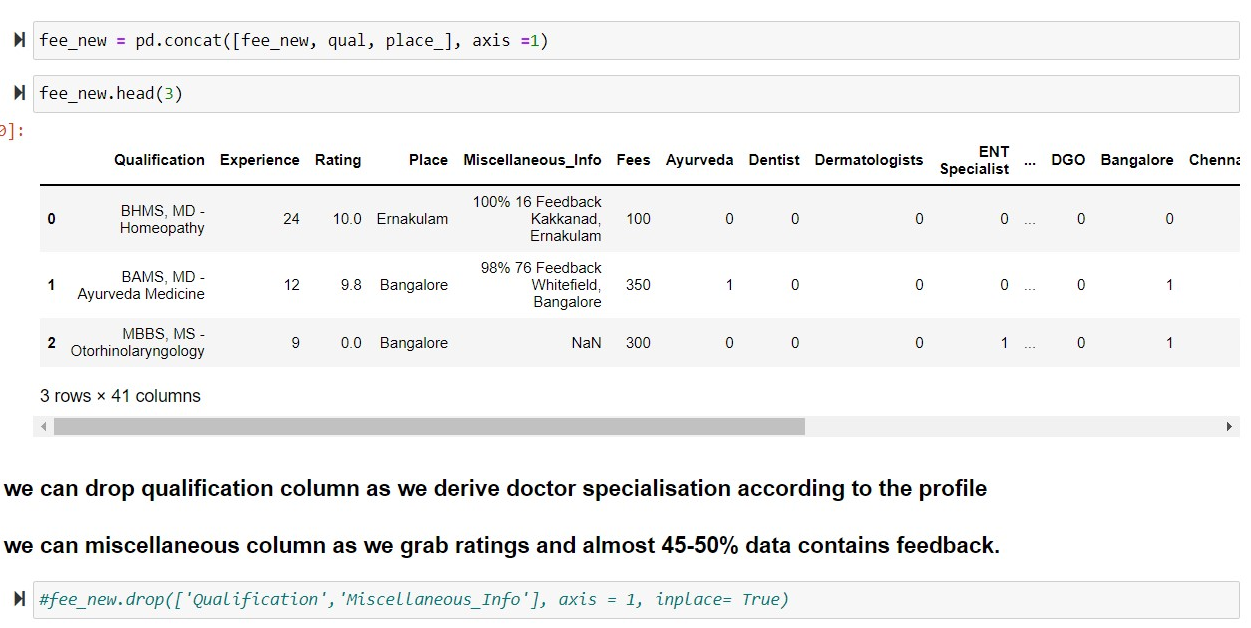
1. **Now we need to convert Qualification variable as per the most frequent values**

* **'BDS', 'BHMS', 'BAMS', 'MBBS', 'MS', 'MD', 'DDVL', 'MDS', 'DDV','ENT' , 'DGO'**
* **these are thee most common qualification count present and we will use these to make features to make better model prediction.**
* **we will make function to separate and build features**



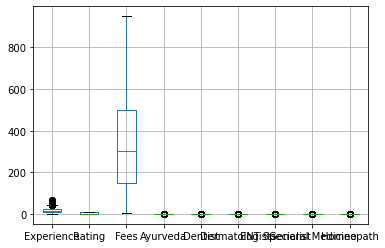


1. **Now we can make our final dataframe by concatenating other and after that removing unwanted variables.**



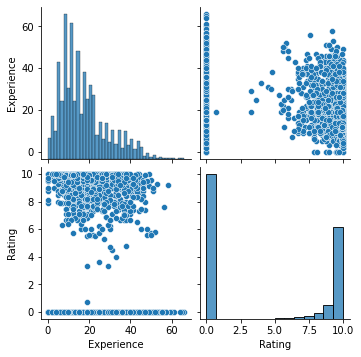
**Exploratory data analysis/data visualisation-:**

1. **Boxplot of the data frame**



1. **Pairplot for two variable (as they have continuous values)**

**sns.pairplot(fee\_new.iloc[:,:4])**



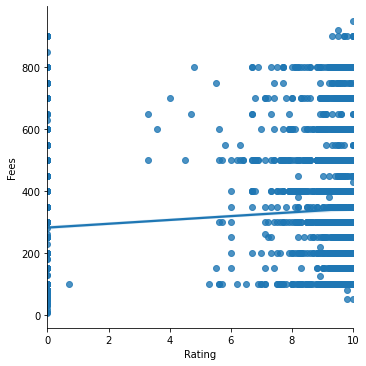
**Observation-:**

**In pairplot we can see that points are distributed in a class form but this problem actually relats to regression problem.**

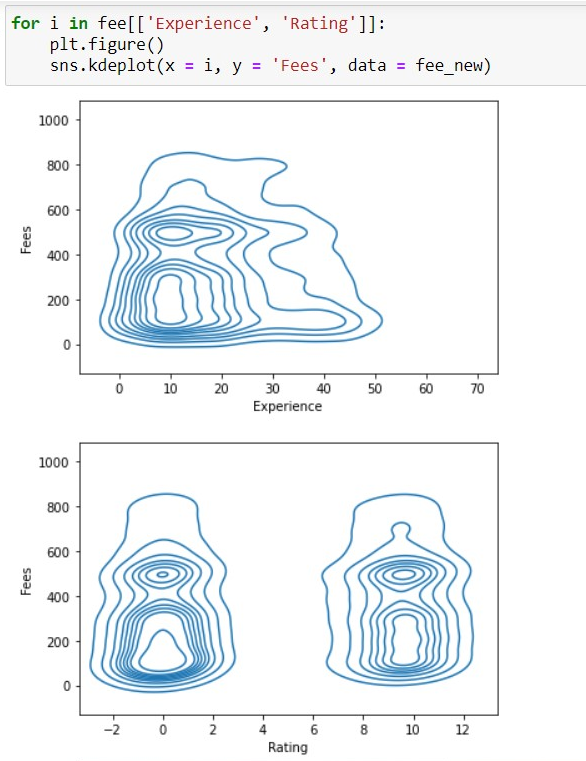
1. **Lmplot for experience**



1. **Lmplot for rating**



1. **Kde plot**

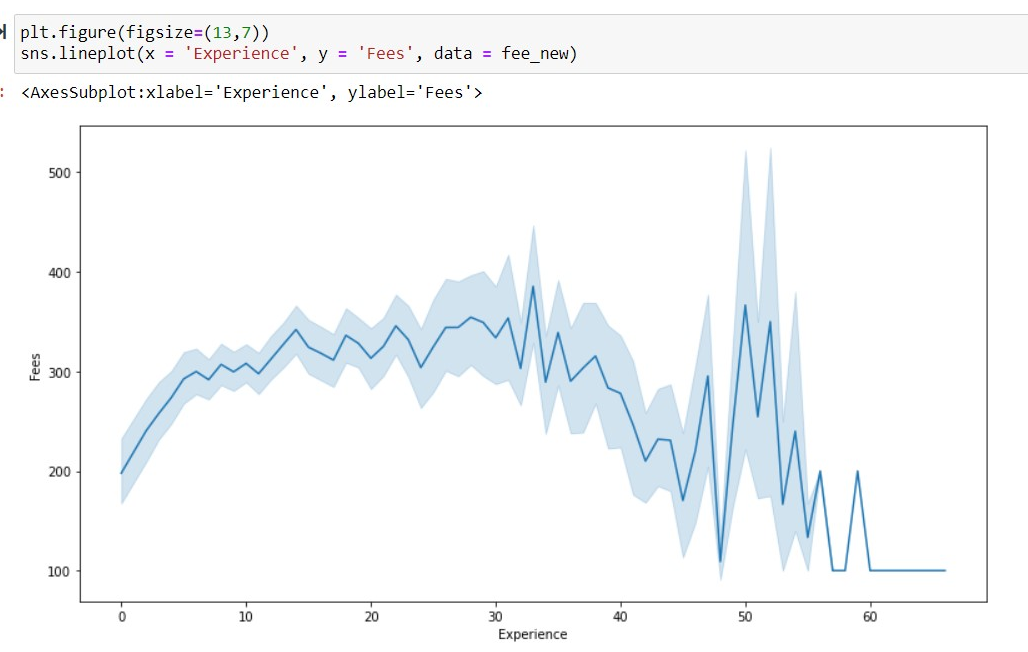


**Observations-:**

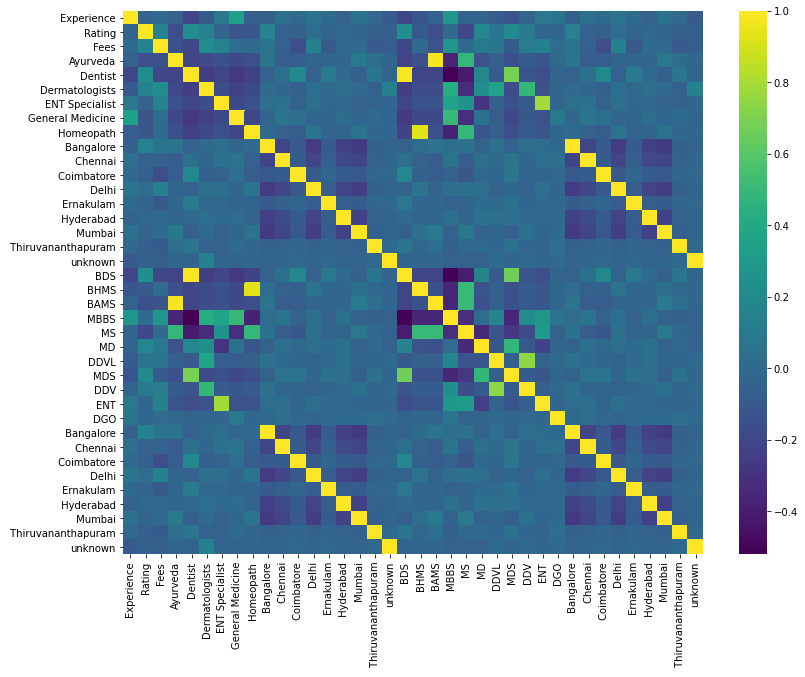
linear regression wont work in this situation as the regression nearly showing 0 or neutral regression due to presence of classes.

* most ratings lies between 0 to 2 and 7-10 and although the fee is same density estimation.
* most experienced doctors charging less fees as they are probably have less qualification or may not be a specialist or tend to charge low.

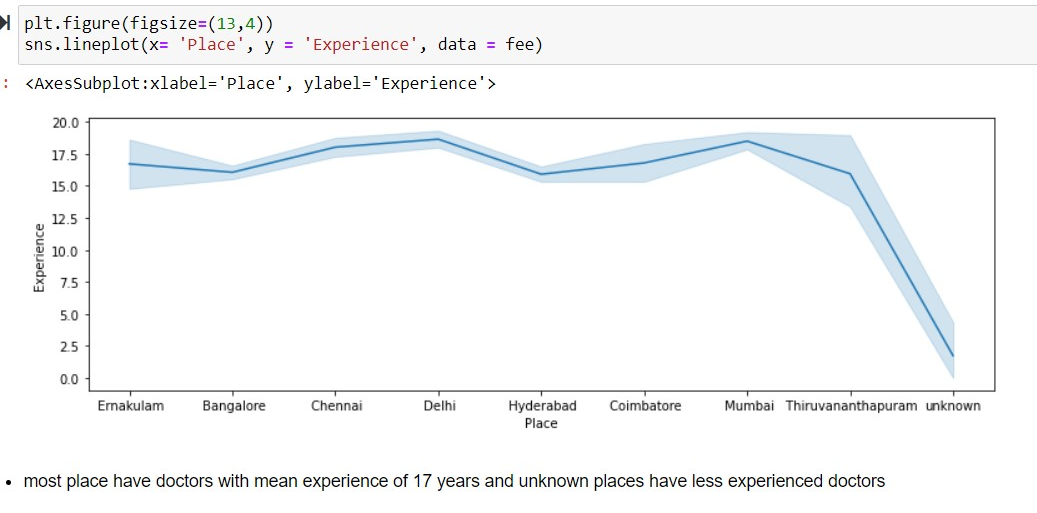
1. **Lineplot**

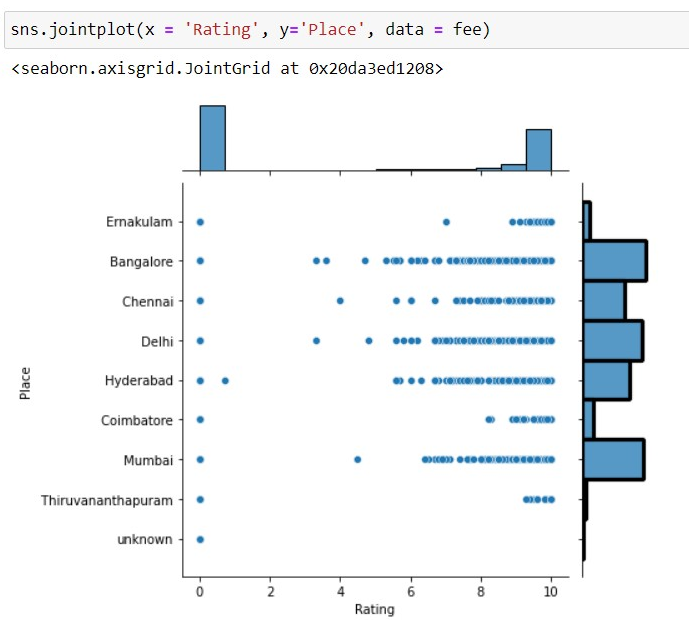


1. **Heatmap of the whole dataframe to see the correlation**



**Bivariate analysis**





**Model Building**

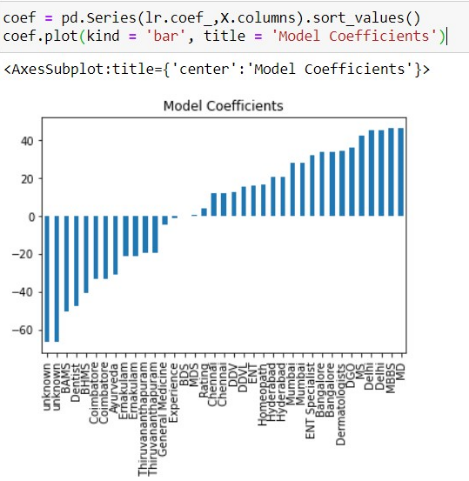
* 1. **Setting the X and y variable and scaling the X data, using train test split and then importing the libraries**

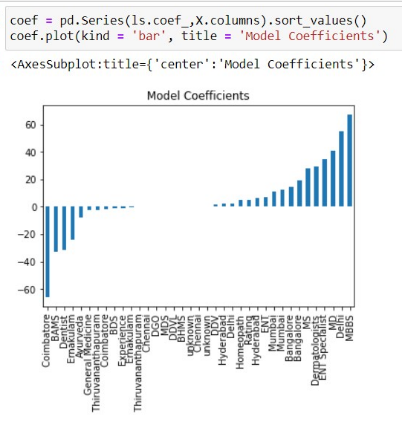


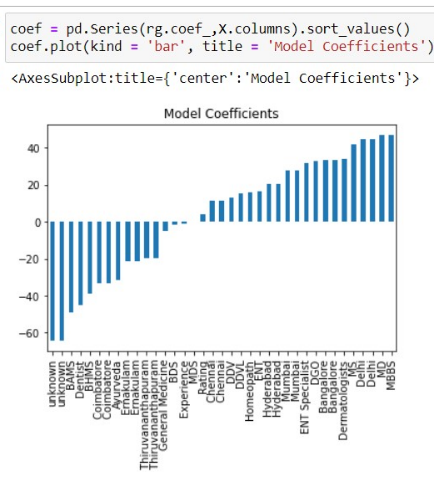
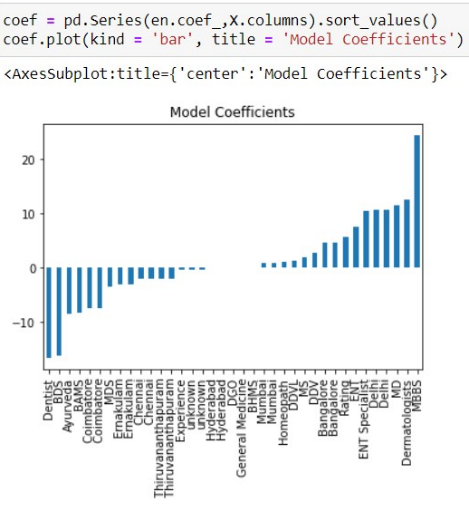
* 1. **Making the model instances and running the models in the for loop and will choose the best performing model from the for loop**



* 1. **Checking model coefficients for selected models**

****

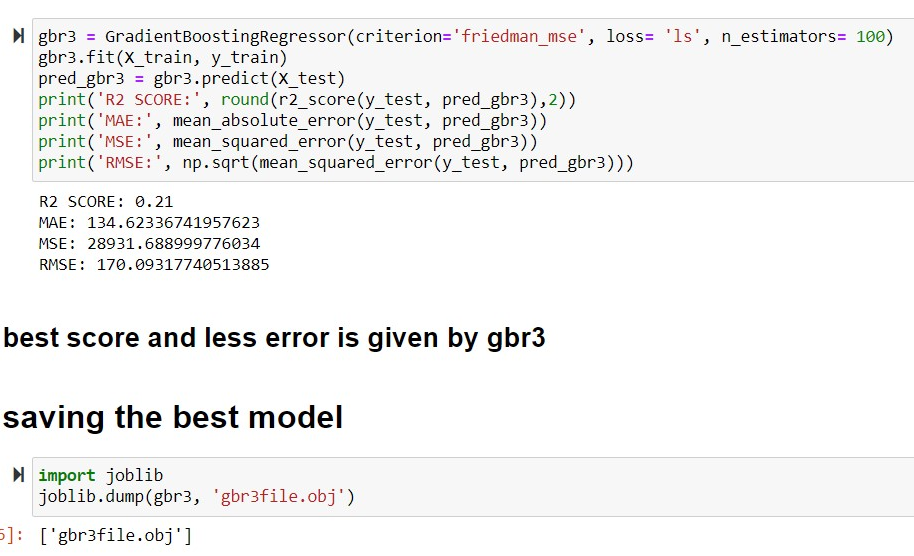




* 1. **Evaluating model using grid search CV**



* 1. **As we didn’t get much better results we had evaluated one more model and achieved good results and better prediction and saved the best model.**



**Conclusion-:**

Yet we has various aspects of our data and yet we had scaled data but somewhat we get low r2 score, but the r2 score generally can be low and high so the model we selected is gradient boost regressor and its predicting good from other models.