Name-Raja Kumar

Batch-DS2312

Project-Zomato Restaurant Machine Learning Project

1.Problem Statement-

In this dataset predict 2 things –

1) Average Cost for two

2) Price range

So in this project we have to predict the average cost for two which is a regression task and the other problem is predicting the price range which is a classification task.

So let us proceed further .

2.Data Analysis:-

This dataset has 9551 rows and 21 columns. We have another dataset as country code.

So at first we merged these two datasets on the common columns country code which is common in both of the datasets.

After merging the dataset looks like this-

	Restaurant ID	Restaurant Name	Country Code	City	Address	Locality	Locality Verbose	Longitude	Latitude	Cuisines	Average Cost for two	Currency	Has Table booking	Has Online delivery
0	6317637	Le Petit Souffle	162	Makati City	Third Floor, Century City Mall, Kalayaan Avenu...	Century City Mall, Poblacion, Makati City	Century City Mall, Poblacion, Makati City, Mak...	121.027535	14.565443	French, Japanese, Desserts	1100	Botswana Pula(P)	Yes	No
1	6304287	Izakaya Kikufuji	162	Makati City	Little Tokyo, 2277 Chino Roces Avenue, Legaspi...	Little Tokyo, Legaspi Village, Makati City	Little Tokyo, Legaspi Village, Makati City, Ma...	121.014101	14.553708	Japanese	1200	Botswana Pula(P)	Yes	No
2	6300002	Heat - Edsa Shangri-La	162	Mandaluyong City	Edsa Shangri-La, 1 Garden Way, Ortigas, Mandal...	Edsa Shangri-La, Ortigas, Mandaluyong City	Edsa Shangri-La, Ortigas, Mandaluyong City, Ma...	121.056831	14.581404	Seafood, Asian, Filipino, Indian	4000	Botswana Pula(P)	Yes	No
3	6318506	Ooma	162	Mandaluyong City	Third Floor, Mega Fashion Hall, SM Megamall, O...	SM Megamall, Ortigas, Mandaluyong City	SM Megamall, Ortigas, Mandaluyong City, Mandal...	121.056475	14.585318	Japanese, Sushi	1500	Botswana Pula(P)	No	No
4	6314302	Sambo Kojin	162	Mandaluyong City	Third Floor, Mega Atrium, SM Megamall, Ortigas...	SM Megamall, Ortigas, Mandaluyong City	SM Megamall, Ortigas, Mandaluyong City, Mandal...	121.057508	14.584450	Japanese, Korean	1500	Botswana Pula(P)	Yes	No

This dataset has no duplicate values and has nulls in one column only.

Let us see this in the picture.

```
data.isna().sum()
```

```
Restaurant ID         0
Restaurant Name       0
Country Code          0
City                  0
Address               0
Locality              0
Locality Verbose      0
Longitude             0
Latitude              0
Cuisines              9
Average Cost for two  0
Currency              0
Has Table booking     0
Has Online delivery   0
Is delivering now     0
Switch to order menu  0
Price range           0
Aggregate rating      0
Rating color          0
Rating text           0
Votes                 0
Country               0
dtype: int64
```

There are no null values in this column do it is a good thing.

Data analysis of various columns.

#Restaurant_id- Altough the restaurant id has different Id for each row. Even then we did not deleted it and we will use this in our model building

#Restaurant_name-

We replaced restaurant name with the value counts of each of the restaurant. Most of the restaurants were appearing only once. So in this way we represented each restaurant with a unique number without loosing too much of information.

#Country

Most of the restaurants were from india and many had 1,2 restaurants.

So we combined the restaurants with very less vaule counts together and later one-hot encoded them.

After the encoding the data looks like.

	code1	code2	code3	Restaurant ID	Restaurant Name	City	Address	Locality	Locality Verbose	Longitude	Latitude	Cuisines	Average Cost for two	Currency	1 boc
0	0.0	0.0	1.0	6317637	Le Petit Souffle	Makati City	Third Floor, Century City Mall, Kalayaan Avenu...	Century City Mall, Poblacion, Makati City	Century City Mall, Poblacion, Makati City, Mak...	121.027535	14.565443	French, Japanese, Desserts	1100	Botswana Pula(P)	
1	0.0	0.0	1.0	6304287	Izakaya Kikufuji	Makati City	Little Tokyo, 2277 Chino Roces Avenue, Legaspi...	Little Tokyo, Legaspi Village, Makati City	Little Tokyo, Legaspi Village, Makati City, Ma...	121.014101	14.553708	Japanese	1200	Botswana Pula(P)	
2	0.0	0.0	1.0	6300002	Heat - Edsa Shangri-La	Mandaluyong City	Edsa Shangri-La, 1 Garden Way, Ortigas, Mandal...	Edsa Shangri-La, Ortigas, Mandaluyong City	Edsa Shangri-La, Ortigas, Mandaluyong City, Ma...	121.056831	14.581404	Seafood, Asian, Filipino, Indian	4000	Botswana Pula(P)	
3	0.0	0.0	1.0	6318506	Ooma	Mandaluyong City	Third Floor, Mega Fashion Hall, SM Megamall, O...	SM Megamall, Ortigas, Mandaluyong City	SM Megamall, Ortigas, Mandaluyong City, Mandal...	121.056475	14.585318	Japanese, Sushi	1500	Botswana Pula(P)	
4	0.0	0.0	1.0	6314302	Sambo Kojin	Mandaluyong City	Third Floor, Mega Atrium, SM Megamall, Ortigas...	SM Megamall, Ortigas, Mandaluyong City	SM Megamall, Ortigas, Mandaluyong City, Mandal...	121.057508	14.584450	Japanese, Korean	1500	Botswana Pula(P)	

#City.

In this also we replaced the values with the value counts of the city.

#Address

In the address also we replaced the address with value counts and later transformed them because the default dataset has very large number of outliers.

#Locality-

In the locality also we also replaced the values with the value counts of the locality columns and later transformed them with using the power transformer because this had also a large number of outilers.

#Locality Verbose- This was the same as the locality column so we deleted it because having two columns of the same type does not makes any sense for the model and it can result into an over fitting.

#Longitude-

This is a numerical column but it had a large number of outliers and the skewness was also very high so we transformed the values using the power transformer and this also resulted into a incease in the correlation with the label also.

```
pd.concat([transformed['Transformed_longi'],data['Average Cost for two'],data['Longitude']],axis=1).corr()
```

	Transformed_longi	Average Cost for two	Longitude
Transformed_longi	1.000000	0.070720	0.915869
Average Cost for two	0.070720	1.000000	0.045891
Longitude	0.915869	0.045891	1.000000

The transformation increased the relation so we will definately use this

The transformation decreased the outliers and also increased the relation with the label so we will be using this transformed longitude for our model.

#Latitude-

For the latitude also the transformation increased the relation with the label and also reduced the outliers by a large number so in this case also we used the power transformed latitude column instead of the default latitude column.

#Cuisines-

This is the only column in this dataset with nulls so we filled the nulls first using the pandas fillna method.

Now the nulls are filled.

```
data.isna().sum()
```

```
code1                    0
code2                    0
code3                    0
Restaurant ID            0
Restaurant Name          0
City                     0
Address                  0
Locality                 0
Longitude                0
Latitude                 0
Cuisines                 0
Average Cost for two     0
Currency                 0
Has Table booking        0
Has Online delivery      0
Is delivering now        0
Switch to order menu     0
Price range              0
Aggregate rating         0
Rating color             0
Rating text              0
Votes                    0
Country                  0
Transformed_outler_no    0
dtype: int64
```

For this column what we did is we used the number of cuisines a restaurant has because each restaurant has different types of cuisines.

```
Cuisines=pd.DataFrame(Cuisines,columns=['No_of_cuisines'])
```

```
Cuisines
```

	No_of_cuisines
0	3
1	1
2	4
3	2
4	2
...	...
9546	1
9547	3
9548	2
9549	1
9550	1

9551 rows × 1 columns

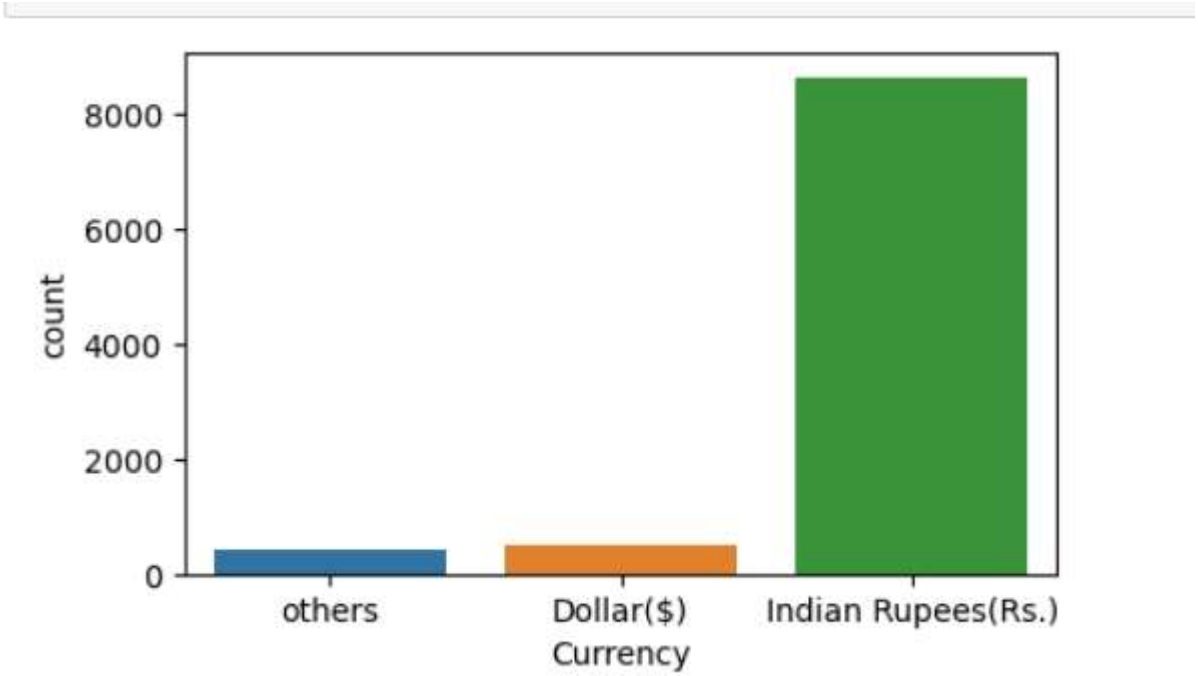
And later we one hot encoded the cuisines column on the basis of number of cuisinses a restautant has.

After encoding the dataset looks like this.

	c1	c2	c3	c4	c5	c6	code1	code2	code3	Restaurant ID	Restaurant Name	City	Address	Locality	Longitude	Latitude	Cuisines	Average Cost for two	Currency	T boo
0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0	6317637	Le Petit Souffle	2	-0.343878	1	2.498033	-1.602771	3	1100	Botswana Pula(P)	
1	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	6304287	Izakaya Kikufuji	2	-0.343878	1	2.497306	-1.603989	1	1200	Botswana Pula(P)	
2	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	6300002	Heat - Edsa Shangri-La	4	-0.343878	1	2.499617	-1.601115	4	4000	Botswana Pula(P)	
3	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	6318506	Ooma	4	-0.343878	3	2.499598	-1.600709	2	1500	Botswana Pula(P)	
4	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	6314302	Sambo Kojin	4	-0.343878	3	2.499653	-1.600799	2	1500	Botswana Pula(P)	

#Average cost for two- As this is the label for the dataset we will not be making any changes int his column at all.

#Currency- This column also has a very large imbalance so we grouped the currency with less value counts together .



We one hot encoded this currency column also and this looks like this.

```
#ONe hot encoding
a=one_hot.fit_transform(data[['Currency']])
a=pd.DataFrame(a,columns=['cu1','cu2','cu3'])
data=pd.concat([a,data],axis=1)
data=data.drop(columns='Currency')
data.head()
```

	cu1	cu2	cu3	c1	c2	c3	c4	c5	c6	code1	code2	code3	Restaurant ID	Restaurant Name	City	Address	Locality	Longitude	Latitude	Average Cost for two	Has Table booking
0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0	6317637	Le Petit Souffle	2	-0.343878	1	2.498033	-1.602771	1100	Yes
1	0.0	0.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	6304287	Izakaya Kikufuji	2	-0.343878	1	2.497306	-1.603989	1200	Yes
2	0.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	6300002	Heat - Edsa Shangri-La	4	-0.343878	1	2.499617	-1.601115	4000	Yes
3	0.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	6318506	Ooma	4	-0.343878	3	2.499598	-1.600709	1500	No
4	0.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	6314302	Sambo Kojin	4	-0.343878	3	2.499653	-1.600799	1500	Yes

#Has online delivery and has online booking-

There are columns with Boolean values so simply we encoded them and this looks like this after the encoding

	b1	b2	d1	d2	cu1	cu2	cu3	c1	c2	c3	c4	c5	c6	code1	code2	code3	Restaurant ID	Restaurant Name	City	Address	Locality	Longitude	Latitude
0	0.0	1.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0	6317637	Le Petit Souffle	2	-0.343878	1	2.498033	-1.602771
1	0.0	1.0	1.0	0.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	6304287	Izakaya Kikufuji	2	-0.343878	1	2.497306	-1.603989
2	0.0	1.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	6300002	Heat - Edsa Shangri-La	4	-0.343878	1	2.499617	-1.601115
3	1.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	6318506	Ooma	4	-0.343878	3	2.499598	-1.600709
4	0.0	1.0	1.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	6314302	Sambo Kojin	4	-0.343878	3	2.499653	-1.600799

#Is delivering now-This is also a column with Boolean value and we simply encoded this using the one hot encoder.

#Switch to order menu-

This columns has all rows same so we deleted this column.

```
data['Switch to order menu'].value_counts()
```

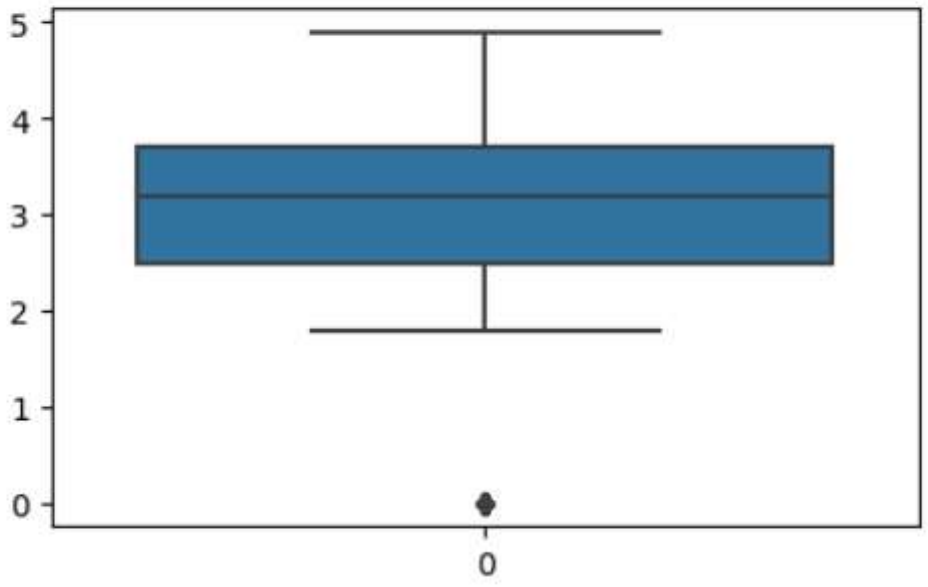
```
Switch to order menu
No    9551
Name: count, dtype: int64
```

#Price Range-This is also one of our label so we made no changes here also.

#Aggregate rating-

This is a continuous column with a little bit of outliers as per the box plot

```
]: plt.figure(figsize=(5,3))
   sns.boxplot(data['Aggregate rating'])
   plt.show()
```



#Rating colour-

This is a categorical column with a bit of imbalance so what we did is we combined the classes with very little value counts together and later one hot encoded the column and after the encoding it looks like this

```
a=one_hot.fit_transform(data[['Rating color']])
a=pd.DataFrame(a,columns=['col1','col2','col3','col4','col5'])
data=pd.concat([a,data],axis=1)
data=data.drop(columns='Rating color')
data.head()
```

	col1	col2	col3	col4	col5	de1	de2	b1	b2	d1	d2	cu1	cu2	cu3	c1	c2	c3	c4	c5	c6	code1	code2	code3	Restaurant ID	Restaurant Name	City	A
0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0	6317637	Le Petit Souffle	2	-0
1	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	6304287	Izakaya Kikufuji	2	-0
2	0.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	6300002	Heat - Edsa Shangri-La	4	-0
3	1.0	0.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	6318506	Ooma	4	-0
4	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	6314302	Sambo Kojin	4	-0

#Rating text- This is also a categorical column with some imbalance so we combined classes with less value counts together and later encoded them.

#Votes- This is continuous column but it has a very large number of outliers so we tried transforming it and this resulted in a less number of outliers along with a better relation with the label so we replaced the default column with the transformed column .

```
]: pd.concat([data['Average Cost for two'],transformed['VOtes_trans'],data['Votes']],axis=1).corr()
```

]:

	Average Cost for two	VOtes_trans	Votes
Average Cost for two	1.000000	0.070303	0.067783
VOtes_trans	0.070303	1.000000	0.556147
Votes	0.067783	0.556147	1.000000

#So we have treated all the feature columns successfully and the final dataset looks like this after all the transformation and the encodings.

	ra1	ra2	ra3	ra4	ra5	ra6	col1	col2	col3	col4	col5	de1	de2	b1	b2	d1	d2	cu1	cu2	cu3	c1	c2	c3	c4	c5	c6	code1	code2	code3	Re
0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0	
1	0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	
2	0.0	0.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	
3	0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	
4	0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	

3.Preprocessing and outlier detection-

To delete the outliers we first selected the continuous columns from our dataset because here we are only concerned about the continuous columns not categorical.

```
']: continous_columns=['Restaurant ID', 'City', 'Address', 'Locality', 'Longitude', 'Latitude',
       'Average Cost for two', 'Price range', 'Aggregate rating', 'Votes',
       'Transformed_outler_no', 'Restaurant Name']
```

```
]: data[continous_columns]
```

]:

	Restaurant ID	City	Address	Locality	Longitude	Latitude	Average Cost for two	Price range	Aggregate rating	Votes	Transformed_outler_no	Restaurant Name
0	6317637	2	-0.343878	1	2.498033	-1.602771	1100	3	4.8	1.191885	-0.634799	-0.624754
1	6304287	2	-0.343878	1	2.497306	-1.603989	1200	3	4.5	1.507786	-0.634799	-0.624754
2	6300002	4	-0.343878	1	2.499617	-1.601115	4000	4	4.4	1.116765	-0.634799	-0.624754
3	6318506	4	-0.343878	3	2.499598	-1.600709	1500	4	4.9	1.266889	-0.634799	-0.624754
4	6314302	4	-0.343878	3	2.499653	-1.600799	1500	4	4.8	1.034953	-0.634799	-0.624754
...	...	...	...	...	...	...	...	...	...	...	...	...
9546	5915730	14	-0.343878	4	-1.640540	2.270905	80	3	4.1	1.652086	-0.634799	-0.624754
9547	5908749	14	-0.343878	1	-1.638445	2.268528	105	3	4.2	1.788687	-0.634799	-0.624754
9548	5915807	14	-0.343878	2	-1.638664	2.276969	170	4	3.7	1.563891	-0.634799	-0.624754
9549	5916112	14	-0.343878	2	-1.638619	2.277366	120	4	4.0	1.719424	-0.634799	-0.624754
9550	5927402	14	-0.343878	1	-1.638946	2.263927	55	2	4.0	1.507786	-0.634799	-0.624754

9551 rows × 12 columns

#Detecting and deleting the outliers.

```
]: Zscore=np.abs(zscore(data[continous_columns]))

]: index=np.where(Zscore>3)[0]

]: len(index)

]: 169

]: #Onlt 169 rows with outliers we can surely delete it
```

Only 169 rows with the outliers have been detected so we will be deleting it.

#After deleting the dataset looks like this

```
: #Let us delete the outliers
  data=data.drop(index=index)
  data.reset_index(drop=True,inplace=True)
  data.head()
```

:

	ra1	ra2	ra3	ra4	ra5	ra6	col1	col2	col3	col4	col5	de1	de2	b1	b2	d1	d2	cu1	cu2	cu3	c1	c2	c3	c4	c5	c6	code1	code2	code3	Re
0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0	
1	0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	
2	0.0	0.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	
3	0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	
4	0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	

#At first we will be making the classification model we divide the features and labels according to this.

```
#Let us make the classification model
x=data.drop(columns='Price range')
y=data['Price range']
```

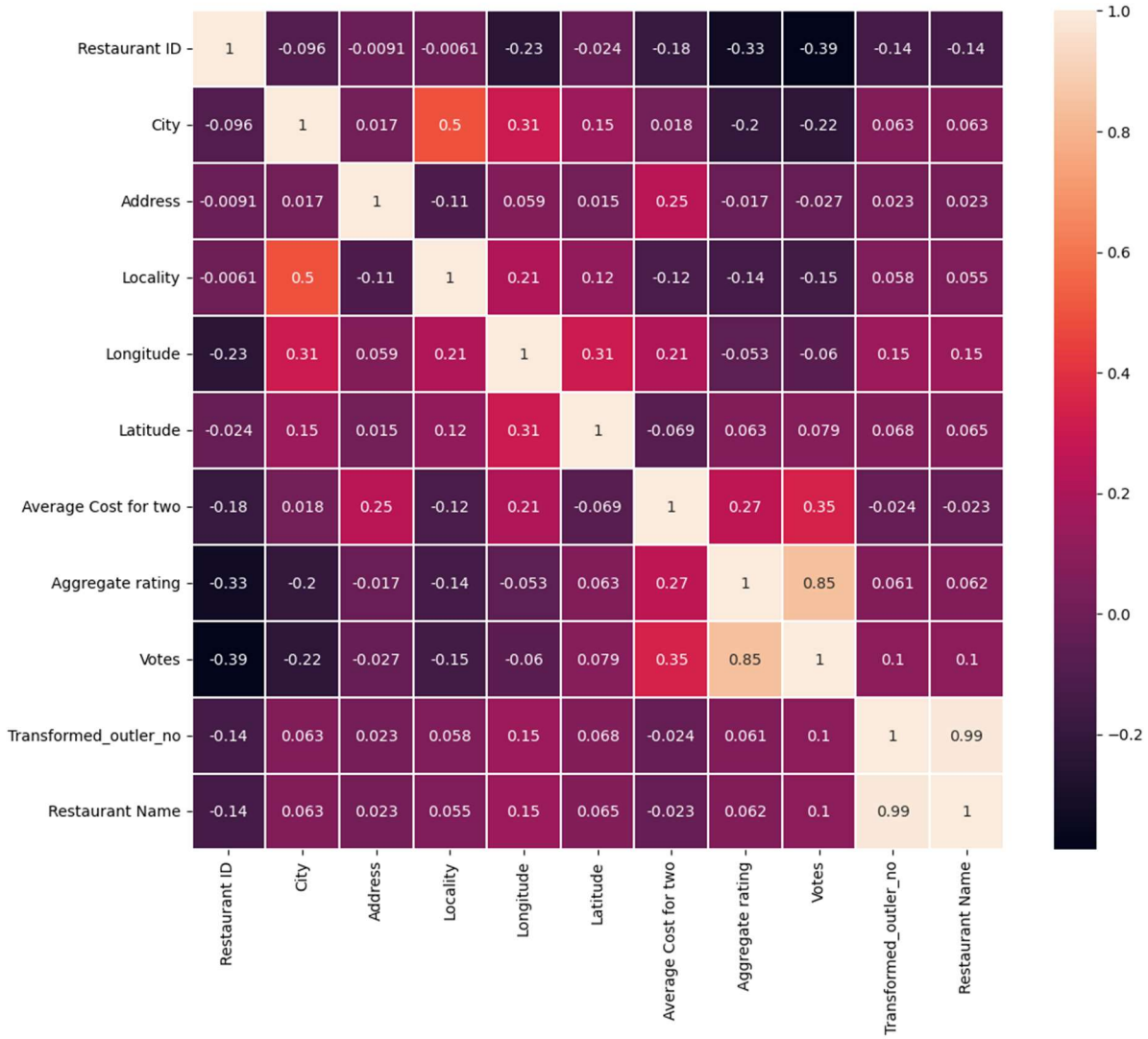
```
x
```

	ra1	ra2	ra3	ra4	ra5	ra6	col1	col2	col3	col4	col5	de1	de2	b1	b2	d1	d2	cu1	cu2	cu3	c1	c2	c3	c4	c5	c6	code1	code2	code3
0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0
1	0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
2	0.0	0.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0
3	0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
4	0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
9377	0.0	0.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
9378	0.0	0.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0
9379	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
9380	0.0	0.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
9381	0.0	0.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0

9382 rows × 40 columns

The average cost had very large number of outliers so what we did is we used the transformed columns for this also rather than the default one.

#Correlation of the continuous columns.



#After that we standardized the continuous columns and we did the analysis of the variance inflation factor score.

After the standardization the continuous columns looks like this.

```
continous_scaled=scaler.fit_transform(data[continous_columns])
continous_scaled=pd.DataFrame(continous_scaled,columns=continous_columns)
continous_scaled
```

	Restaurant ID	City	Address	Locality	Longitude	Latitude	Average Cost for two	Aggregate rating	Votes	Transformed_outlier_no	Restaurant Name
0	-0.307891	-1.437433	-0.346946	-1.346158	2.832662	-1.830443	0.844802	1.424337	1.205833	-0.639646	-0.629505
1	-0.309400	-1.437433	-0.346946	-1.346158	2.831843	-1.831849	1.010658	1.226423	1.521714	-0.639646	-0.629505
2	-0.309884	-1.436602	-0.346946	-1.346158	2.834448	-1.828530	5.654641	1.160452	1.130718	-0.639646	-0.629505
3	-0.307793	-1.436602	-0.346946	-1.275613	2.834426	-1.828061	1.508228	1.490308	1.280833	-0.639646	-0.629505
4	-0.308268	-1.436602	-0.346946	-1.275613	2.834489	-1.828165	1.508228	1.424337	1.048912	-0.639646	-0.629505
...	...	...	...	...	...	...	...	...	...	...	...
9377	-0.353311	-1.432445	-0.346946	-1.240340	-1.832694	2.644274	-0.846935	0.962538	1.666005	-0.639646	-0.629505
9378	-0.354100	-1.432445	-0.346946	-1.346158	-1.830333	2.641529	-0.805471	1.028510	1.802598	-0.639646	-0.629505
9379	-0.353302	-1.432445	-0.346946	-1.310886	-1.830579	2.651279	-0.697664	0.698654	1.577816	-0.639646	-0.629505
9380	-0.353268	-1.432445	-0.346946	-1.310886	-1.830528	2.651737	-0.780592	0.896567	1.733339	-0.639646	-0.629505
9381	-0.351992	-1.432445	-0.346946	-1.346158	-1.830898	2.636214	-0.888399	0.896567	1.521714	-0.639646	-0.629505

9382 rows × 11 columns

#We saw the variance score of the continuous columns but some were having a high score.

```
for i in range(11):
    print(variance_inflation_factor(continous_scaled,i))
    print(continous_columns[i])
    print('*'*10)
```

```
1.3346787507966495
Restaurant ID
**********
1.5149459638108413
City
**********
1.098712636230603
Address
**********
1.394834085079976
Locality
**********
1.4298239968847437
Longitude
**********
1.1913125357503738
Latitude
**********
1.4131349591242222
Average Cost for two
**********
3.526054944107172
Aggregate rating
**********
4.225504385486358
Votes
**********
55.249737953831236
Transformed_outler_no
**********
55.29520375964021
Restaurant Name
**********
```

Altough some columns were having a high variance but we did not deleted them and made the firs model using all the columns.

After this we joined the main continuous scaled columns to the main featue columns and the preprocessing is done for the classification model we will make.

4.Model building for classification task.

#The best random state

We used the logistic regression to know the best random state for the train test split.

```
maxACC=0
maxrs=0
for i in range(1,200):
    lr=LogisticRegression()
    x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25,random_state=i)
    lr.fit(x_train,y_train)
    pred=lr.predict(x_test)
    acc=accuracy_score(y_test,pred)
    if acc>maxACC:
        maxACC=acc
        maxrs=i
print(maxACC,maxrs)
```

```
0.9364876385336743 92
```

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25,random_state=maxrs)
```

We made the models using other algorithms also and some model had very high accuracy also. The accuracy of other model are below:-

```
RandomForestClassifier()    0.9688832054560955
ExtraTreesClassifier()    0.8712702472293266
LogisticRegression()    0.9364876385336743
SVC()    0.9045183290707587
DecisionTreeClassifier()    0.985080988917306
GradientBoostingClassifier()    0.9825234441602728
AdaBoostClassifier()    0.9254049445865302
BaggingClassifier()    0.9872122762148338
```

The accuracy after the cross validation is also below

```
RandomForestClassifier()    0.9236985594896361
ExtraTreesClassifier()    0.8144554834473648
LogisticRegression()    0.9044046265362432
SVC()    0.8805284029657633
DecisionTreeClassifier()    0.9275369101671792
GradientBoostingClassifier()    0.951084671020421
AdaBoostClassifier()    0.7321483381479087
BaggingClassifier()    0.9251933687222612
```

The gradient boost is the best performing model till now.

After that we deleted the restaurant id to se if the accuracy increases or decreases but it resulted In a decrease in the accuracy .

```
RandomForestClassifier()    0.9667519181585678
ExtraTreesClassifier()    0.8878942881500427
LogisticRegression()    0.9386189258312021
SVC()    0.907928388746803
DecisionTreeClassifier()    0.9808184143222506
GradientBoostingClassifier()    0.9838022165387894
AdaBoostClassifier()    0.9254049445865302
BaggingClassifier()    0.985080988917306
```

So we will be doing the tuning for the gradient boost and using all the feature columns because it had the best accuracy and the best accuracy after the cross validation also.

After the tuning the gradient boost had the accuracy of 98.

We also used different random state to know the best random state at which it has the best accuracy and after this the gradient boost had an accuracy of 98.2

```
maxACC=0
maxrs=0
for i in range(1,200):
    model_classi=GradientBoostingClassifier(learning_rate=0.1,max_depth=4,max_features='sqrt',
                                            min_samples_leaf=2,min_samples_split=5,
                                            n_estimators=200)
    model_classi.fit(X_train,Y_train)
    pred=model_classi.predict(X_test)
    acc=accuracy_score(Y_test,pred)
    if acc>maxACC:
        maxACC=acc
        maxrs=i
print(maxACC,maxrs)
```

```
KeyboardInterrupt
```

```
maxACC
```

```
0.9829497016197783
```

So this concludes the classification task now let us proceed to the regression task.

Now for the regression task the average price is the label the price range is the feature column so we have to encode the feature column respectively and after the encoding the feature column looks like this.

```
#Here firs we need to encode the price range because this time it is not a feature but a label
a=one_hot.fit_transform(x[['Price range']])
a=pd.DataFrame(a,columns=['p1','p2','p3','p4'])
x=pd.concat([a,x],axis=1)
x=x.drop(columns='Price range')
```

	ra1	ra2	ra3	ra4	ra5	ra6	col1	col2	col3	col4	col5	de1	de2	b1	b2	d1	d2	cu1	cu2	cu3	c1	c2	c3	c4	c5	c6	code1	code2	code3	Re
0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0	
1	0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	
2	0.0	0.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	
3	0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	
4	0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	

We dropped the continuous columns from the dataset and replaced them with the standardized continuous columns.

```
continous_scales=scaler.fit_transform(x[continous_columns])
continous_scales=pd.DataFrame(continous_scales,columns=continous_columns)
continous_scales
```

	Restaurant ID	City	Address	Locality	Longitude	Latitude	Aggregate rating	Votes	Transformed_outler_no	Restaurant Name
0	-0.307891	-1.437433	-0.346946	-1.346158	2.832662	-1.830443	1.424337	1.205833	-0.639646	-0.629505
1	-0.309400	-1.437433	-0.346946	-1.346158	2.831843	-1.831849	1.226423	1.521714	-0.639646	-0.629505
2	-0.309884	-1.436602	-0.346946	-1.346158	2.834448	-1.828530	1.160452	1.130718	-0.639646	-0.629505
3	-0.307793	-1.436602	-0.346946	-1.275613	2.834426	-1.828061	1.490308	1.280833	-0.639646	-0.629505
4	-0.308268	-1.436602	-0.346946	-1.275613	2.834489	-1.828165	1.424337	1.048912	-0.639646	-0.629505
...	...	...	...	...	...	...	...	...	...	...
9377	-0.353311	-1.432445	-0.346946	-1.240340	-1.832694	2.644274	0.962538	1.666005	-0.639646	-0.629505
9378	-0.354100	-1.432445	-0.346946	-1.346158	-1.830333	2.641529	1.028510	1.802598	-0.639646	-0.629505
9379	-0.353302	-1.432445	-0.346946	-1.310886	-1.830579	2.651279	0.698654	1.577816	-0.639646	-0.629505
9380	-0.353268	-1.432445	-0.346946	-1.310886	-1.830528	2.651737	0.896567	1.733339	-0.639646	-0.629505
9381	-0.351992	-1.432445	-0.346946	-1.346158	-1.830898	2.636214	0.896567	1.521714	-0.639646	-0.629505

9382 rows × 10 columns

At first we will be using all the feature columns and the default label also.

#Finding the best random state for the train test split

```
maxACC=0
maxrs=0
for i in range(1,200):
    lr=LinearRegression()
    x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=i)
    lr.fit(x_train,y_train)
    pred=lr.predict(x_test)
    acc=r2_score(y_test,pred)
    if acc>maxACC:
        maxACC=acc
        maxrs=i
print(maxACC,maxrs)
```

```
0.7779700236146141 133
```

So we divided the feature and the label on the basis of the best random state.

Accuracy of different models on the basis of this split:-

```
{RandomForestRegressor(): 0.8796356265200589,
 ExtraTreesRegressor(): 0.8366966619790703,
 LinearRegression(): 0.7779700236146141,
 GradientBoostingRegressor(): 0.8647851071306186,
 AdaBoostRegressor(): 0.5654095267348944,
 BaggingRegressor(): 0.8756289801255848,
 Ridge(): 0.7790536915580564,
 LassoCV(): 0.7798168208938872,
 DecisionTreeRegressor(): 0.7308162028566558}
```

Cross validation score:-

```
RandomForestRegressor() , 0.6772496721460677
ExtraTreesRegressor() , 0.6032689874488231
LinearRegression() , -1.1668551429533494e+22
GradientBoostingRegressor() , 0.7313946456570332
AdaBoostRegressor() , 0.5200050388253386
BaggingRegressor() , 0.6476659443131549
Ridge() , 0.6824763918439289
LassoCV() , 0.6681603565745491
DecisionTreeRegressor() , 0.2392717014223631
```

Using this way the models are performing very poorly.

So we tried other way for this.

This time we will be using the power transformed label instead of the default label.

#Finding the best random state for the split this time:-

```
: #Let us build the modelmaxACC=0
maxrs=0
for i in range(1,200):
    lr=LinearRegression()
    x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=i)
    lr.fit(x_train,y_train)
    pred=lr.predict(x_test)
    acc=r2_score(y_test,pred)
    if acc>maxACC:
        maxACC=acc
        maxrs=i
print(maxACC,maxrs)
```

```
0.8656796162200379 94
```

We can clearly see a increase In the accuracy of linear regression also

The accuracy of other models based on this split and the transformed label:-

```
: {RandomForestRegressor(): 0.9072703582440925,
   ExtraTreesRegressor(): 0.8964284475331983,
   LinearRegression(): 0.8656796162200379,
   GradientBoostingRegressor(): 0.8980417651761168,
   AdaBoostRegressor(): 0.7840158658303258,
   BaggingRegressor(): 0.8967025326057595,
   Ridge(): 0.8648994843855068,
   LassoCV(): 0.8647208039918658,
   DecisionTreeRegressor(): 0.8176066037514953}
```

Clearly a very big increase in all the models using the transformed label.

The cross validation score:-

```
RandomForestRegressor() , 0.6435720293918141
ExtraTreesRegressor() , 0.6501479987459107
LinearRegression() , 0.7630559680953694
GradientBoostingRegressor() , 0.7105937314105192
AdaBoostRegressor() , 0.641619038022025
BaggingRegressor() , 0.6600597384788687
Ridge() , 0.7629439641024305
LassoCV() , -0.039634817841475954
DecisionTreeRegressor() , 0.45481768721160065
```

The gradient boost has been the best model we had done the tuning for this.

And after the tuning it had an r2_score of 0.91 and a mean absolute error Of 0.2351706166989455. And the mean_squared_error was 0.10003684855627662.

So we also made the model using other techniques for example- by deleting t he restaurant id or by deleting the columns with high correlation but it re sulted in a decreased accuracy.

So this concluded the model building.
We used the transformed label . SO the actual label and predicted label for the best model where we used all the feature columns along with the Transformed label is below:-

	Predicted	Actual
0	653.381866	600.0
1	580.841517	500.0
2	552.002155	600.0
3	4639.655717	4000.0
4	277.047680	300.0
5	600.319375	500.0
6	647.524799	700.0
7	352.038846	300.0
8	606.552714	650.0
9	239.614544	350.0
10	268.204306	300.0
11	585.489966	650.0

We can clearly see that the prediction is close to the actual and there is not a lot of difference.

This concludes the model building.

5.Conclusion:-

Conclusions.(for classification part where we predicted the price range of the restaurants):-

1.At first we need to do a lot of preprocessing of the feature columns as there were a lot of categorical columns which needed to be encoded accordingly. So at first we treated all the columns accordingly and made them suitable for the model buiding process. After that we treated the outliers for the continous columns and deleted the outliers. In the last we separated the

label and the feature columns and started building the classification model

2.For the classification task we made the models in 3 ways:-

1.In the first technique we made the model without considering the multicollinearity and the variance inflation factor score

and mde the model using all the feature columns. In this way also some models like the random forest, gradient boost gave us

a great accuracy and the best accuracy was of the gradient boost which was 96%.

2.In the second time we deleted the columns with high vif scores and this resulted in a better accuracy. This time each model

performed better than the previous one and we got the best accuracy of 98% which was of the gradient boost and after the

tuning this gave us an accuracy of 98.2% which was the best model for us for the classificatin part

3.The best model was the gradient boost classifier where we deleted the feature column with high variance inflation score and

also we used the transformed average cost column as it had a very high skewness so this is why we had to use the transformed column for this.

Conclusions.(for regression part where we predicted the average cost of the restaurants):-

1.The preprocessing was the same for this task also . They only difference was that this time the label was the average cost column and the price range column was the feature this time. So what we did is we deleted this time the average cost from the dataset and encoded the price range column accordingly and than we started building the model.

2.This time also we made the model using many techniques:-

1.At first we made the model using all the feature columns after treating and deleting the outliers. And model performed well

in this case also with randomforest regressor with the best accuracy of 87%.

2.After that we deleted the columns with high correlation and this resulted in a slight better accuracy.

3.In the last we used the transformed label because the actual default label was having a very high skewness and it had a

very high no of outliers and this resulted in a very good accuracy. The best performing model was the gradientboost

regressor with an accuracy of 90.1% after the tuning and this ws 3% more than the previous best.

3.The best performing model was the gradient boost regressor with an r2_Score of 90.1%. And in this model we deleted the

feature columns with very high correlation and we used the transformed label which resulted in a extra 3% accuracy.

this model had an mean abolute error of 0.2351706166989455 and mean squared error of 0.1000 respectively.

In the last we also converted the label and the prediction to the actual form to see the actual difference between our prediction and the actual values.

----------This concludes this project----------

****
****
THANKS
****
****