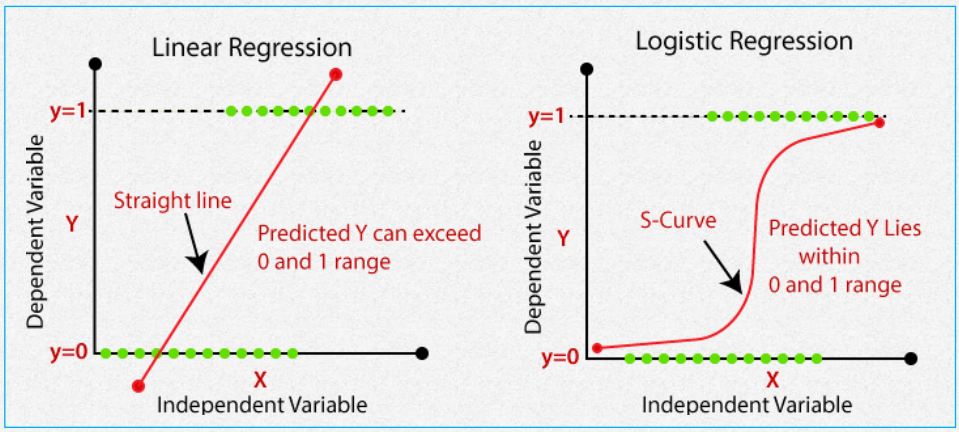
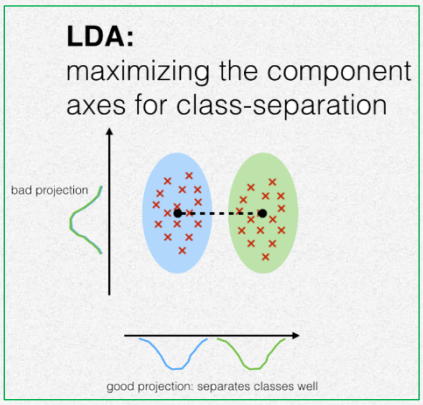
PREDICTIVE MODELING

Group Assignment

(Linear Regression, Logistic Regression & LDA)





Lavanya

Nikhil

Rekha

Rajiv

Submission:

April, 2021

GREAT LAKES
INSTITUTE OF MANAGEMENT

greatlearning
Power Ahead

About Us

This section will cover brief introduction about us. We are seasoned professionals with diverse experience.

Rajiv

Am a CA and currently working as **Vice President & Group Head of Internal Audits** for Flipkart, Myntra, PhonePe, e-Kart and Walmart India. Prior to that I have worked in different audit and governance roles in PwC, Coke and Diageo.

Rekha

Overall **20+ years** of corporate and management experience. Currently (from 2018) working as **Strategy Consultant (Independently)** for firms and companies that range from hospitality, manufacturing, and retail industries. Before moving to an independent role was with **KPMG Global Services for 12 years**. Hands-on manager with expertise in accounting systems development, fiscal management. and financial reporting. Proven record of developing and implementing financial and operational controls that improve P&L scenario and competitively position firms.

Nikhil

Overall **13+ years of cross functional experience** in FMCG and Alcobev industry. Working with **Diageo PLC** since 2017, looking after **Data Analytics CoE for Global Audit & Risk department**. Before to that worked with **Coca-Cola for 10 years** and completed stint in operations, supply chain (direct & indirect), Master data maintenance (SAP cross functional role) and Data analytics (Internal Audit).

Lavanya

Working with **Novo Nordisk** as a Manager, having **13 years of techno-functional experience** in SAP Fiori, Embedded Analytics,SCP, ABAP on HANA and also having SAP Consulting experience on UX design strategy, RFP's/Pursuits and solutioning. During this tenure, worked in **various SAP ERP implementation projects spanning across Manufacturing, Healthcare, Retail and FMCG domains**. Prior to this, I have worked with **Accenture, Caterpillar and Wipro**.

Table of Contents:

Problem1:

You are hired by a company named Gem Stones Co Ltd, which is a cubic zirconia manufacturer. You are provided with the dataset containing the prices and other attributes of approximately 27,000 pieces of cubic zirconia (which is an inexpensive synthesized diamond alternative with similar qualities of a diamond).

Your objective is to accurately predict prices of the zircon pieces. Since the company profits at a different rate at different price levels, for revenue management, it is important that prices are predicted as accurately as possible. At the same time, it is important to understand which of the predictors are more important in determining the price.

The data dictionary is given below:

Variable Name	Description
Carat	Carat weight of the cubic zirconia.
Cut	Describes the cut quality of the cubic zirconia. Quality is in increasing order: Fair, Good, Very Good, Premium, Ideal.
Colour	Colour of the cubic zirconia.
Clarity	Cubic zirconia Clarity refers to the absence of the Inclusions and Blemishes. (In order from Best to Worst, FL = flawless, I3= level 3 inclusions) FL, IF, VVS1, VVS2, VS1, VS2, SI1, SI2, I1, I2, I3
Depth	The Height of a cubic zirconia piece, measured from the Culet to the table, divided by its average Girdle Diameter.
Table	The Width of the cubic zirconia's Table expressed as a Percentage of its Average Diameter.
Price	Price of the cubic zirconia.
X	Length of the cubic zirconia in mm.
Y	Width of the cubic zirconia in mm.
Z	Height of the cubic zirconia in mm.

Remarks: All the questions of problem1 which explained here, are also performed in python as well. Please refer python notebook "PM_GA_Problem1_Linear Regression".

1.1 The very first step of any data analysis assignment is to do the exploratory data analysis (EDA).
Once you have understood the nature of all the variables, identified the response and the predictors, apply appropriate methods to determine whether there is any duplicate observation or missing data and whether the variables have symmetric or skewed distribution. Note that data may contain various types of attributes and numerical and/or visual data summarization techniques need to be appropriately decided. Both univariate and bivariate analyses and pre-processing of data are important. Check for outliers and comment on removing or keeping them while model building. Since this is a regression problem, the dependence of the response on the predictors needs to be thoroughly investigated.

Exploratory Data Analysis (EDA) : EDA is performed to understand the data first and try to gather as many insights from it. In short EDA is all about checking the dataset before the same is used to make any predictive model. EDA is a critical process for performing initial investigations on data so as to discover patterns, to spot anomalies and to check assumptions with the help of summary statistics and graphical representations.

Below is the Exploratory Data Analysis of the data set "cubic_zirconia" of our problem statement.

Variable Summary and Status:

Steps and Results:

- Import libraries: as first step, we imported all possible libraries which may be required for the entire project, viz:
  - Numpy, pandas, seaborn
  - Scipy, pyplot
  - Sklearn- train test split, standard scaler.
- Load and read the data set "cubic_zirconia.csv".
- Understand the dataset by displaying the first few rows (**code: df.head()**).

	Unnamed: 0	carat	cut	color	clarity	depth	table	x	y	z	price
0	1	0.30	Ideal	E	SI1	62.1	58.0	4.27	4.29	2.66	499
1	2	0.33	Premium	G	IF	60.8	58.0	4.42	4.46	2.70	984
2	3	0.90	Very Good	E	VVS2	62.2	60.0	6.04	6.12	3.78	6289
3	4	0.42	Ideal	F	VS1	61.6	56.0	4.82	4.80	2.96	1082
4	5	0.31	Ideal	F	VVS1	60.4	59.0	4.35	4.43	2.65	779

- Analysing the dimension of the dataset and the data type of variables in the dataset (**code: df.shape() & df.info()**)

```
The number of columns (variables) in the dataset is 11
The number of rows (observations per variable) in the dataset is 26967
```

```
Data columns (total 11 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Unnamed: 0  26967 non-null  int64
 1   carat       26967 non-null  float64
 2   cut         26967 non-null  object
 3   color       26967 non-null  object
 4   clarity     26967 non-null  object
 5   depth       26270 non-null  float64
 6   table       26967 non-null  float64
 7   x           26967 non-null  float64
 8   y           26967 non-null  float64
 9   z           26967 non-null  float64
 10  price       26967 non-null  int64
dtypes: float64(6), int64(2), object(3)
memory usage: 2.3+ MB
```

- Then we **dropped first column (Unnamed: 0) using. drop function**, as this column is not required. And we checked the dataset again using. head function

	carat	cut	color	clarity	depth	table	x	y	z	price
0	0.30	Ideal	E	SI1	62.1	58.0	4.27	4.29	2.66	499
1	0.33	Premium	G	IF	60.8	58.0	4.42	4.46	2.70	984
2	0.90	Very Good	E	VVS2	62.2	60.0	6.04	6.12	3.78	6289
3	0.42	Ideal	F	VS1	61.6	56.0	4.82	4.80	2.96	1082
4	0.31	Ideal	F	VVS1	60.4	59.0	4.35	4.43	2.65	779

- We checked the existence of duplicate records in the df using. duplicate function and identified **34 duplicate records**:

```
There are 34 records in the dataset which are duplicates
```

- We then dropped these duplicate records using**. drop_duplicates** function and checked the remaining number of rows and variables using. shape function:

```
The number of columns (variables) in the dataset is 10
The number of rows (observations per variable after removing duplicates) in the dataset is 26933
```

- Checking for null value in the dataset **(code: df.isnull().sum())**

  There **are 697 rows (i.e., ~2.6%) where "depth" column values are null.**

```
carat        0
cut          0
color        0
clarity      0
depth      697
table        0
x            0
y            0
z            0
price        0
dtype: int64
```

There are different ways to replace null values of depth. We performed some web research on the factors on which the depth of zirconia is dependent (attached in along with the Business Report submission) and came up with two possibly better approaches for imputation VS simply replacing NULL with median or mean of the depth, in general.

Data Imputation (variable : depth)

Approach 1:

- Merge **three columns (i.e., cut, color and clarity)**
- Take the **mean** of the three columns
- Replace **mean values of these with null values of above combination**
- Using group by function .**groupby() replace null values** of "merge_col" combination with the mean depth value of "merge_col".

	carat	cut	color	clarity	depth	table	x	y	z	price	merge_col
0	0.30	Ideal	E	SI1	62.1	58.0	4.27	4.29	2.66	499	IdealESI1
1	0.33	Premium	G	IF	60.8	58.0	4.42	4.46	2.70	984	PremiumGIF
2	0.90	Very Good	E	VVS2	62.2	60.0	6.04	6.12	3.78	6289	Very GoodEVVS2
3	0.42	Ideal	F	VS1	61.6	56.0	4.82	4.80	2.96	1082	IdealFVS1
4	0.31	Ideal	F	VVS1	60.4	59.0	4.35	4.43	2.65	779	IdealFVVS1

- Please find the below output after applying the Approach 1

  Still, we could see **one row which has null depth value**

```
carat         0
cut           0
color         0
clarity       0
depth         1
table         0
x             0
y             0
z             0
price         0
merge_col     0
dtype: int64
```

Approach 2:

- Now, merge **two columns (i.e., cut and clarity)**
- Take the **mean** of the two columns (cut and clarity)
- Replace **mean values of that with remaining null values of above combination**
- Save merge_col2 value into (**df_replace**) dataset.
- Check top five records whether the column is added

	carat	cut	color	clarity	depth	table	x	y	z	price	merge_col	merge_col2
0	0.30	Ideal	E	SI1	62.1	58.0	4.27	4.29	2.66	499	IdealESI1	IdealSI1
1	0.33	Premium	G	IF	60.8	58.0	4.42	4.46	2.70	984	PremiumGIF	PremiumIF
2	0.90	Very Good	E	VVS2	62.2	60.0	6.04	6.12	3.78	6289	Very GoodEVVS2	Very GoodVVS2
3	0.42	Ideal	F	VS1	61.6	56.0	4.82	4.80	2.96	1082	IdealFVS1	IdealVS1
4	0.31	Ideal	F	VVS1	60.4	59.0	4.35	4.43	2.65	779	IdealFVVS1	IdealVVS1

into the dataset or not using**. head()**

Using group by function. **groupby() replace null values of "merge_col2" combination with the mean depth value of "merge_col2"**. We get following **ZERO count of null values after applying Approach 2**.

```
carat         0
cut           0
color         0
clarity       0
depth         0
table         0
x             0
y             0
z             0
price         0
merge_col     0
merge_col2    0
dtype: int64
```

Further, let's check is there any **ZERO values in the numerical variable's columns** using

(df_replace == 0.sum)

```
carat          0
cut            0
color          0
clarity        0
depth          0
table          0
x              2
y              2
z              8
price          0
merge_col      0
merge_col2     0
dtype: int64
```

There are **three columns (x, y and z) which have ZERO values** in the Dataset, where x, y, z represents the Length, Width and Height of the cubic zirconia respectively.

Let's take the alike approach as above to replace zero values.

Approach1:

- Merge **four columns (i.e., carat, cut, color and clarity**) and
- Take the **mean of that column**.
- Replace **mean values of these with null values of above combination**.
- Check top five records whether the column "merge_col3" is added into the dataset or not using**. head()**

```
carat          0
cut            0
color          0
clarity        0
depth          0
table          0
x              1
y              1
z              3
price          0
merge_col      0
merge_col2     0
merge_col3     0
dtype: int64
```

From the above output we could still see the **five values** which has ZERO values in the dataset.

Approach 2:

- Merge **three columns (i.e., carat, color and clarity)**
- Take the **mean** of that
- Replace the **mean values of these with null values of above combination**.
- save "merge_col4" value **into df_replace** dataset
- Using group by function (code: groupby()) replace null values of "merge_col4" combination with the mean x, y, z value of "merge_col4".
- Check top five records whether the column is added into the dataset or not using. head()

	carat	cut	color	clarity	depth	table	x	y	z	price	merge_col	merge_col2	merge_col3	merge_col4
0	0.3	Ideal	E	SI1	62.1	58.0	4.27	4.29	2.66	499	IdealESI1	IdealSI1	0.3IdealESI1	0.3ESI1
1	0.33	Premium	G	IF	60.8	58.0	4.42	4.46	2.70	984	PremiumGIF	PremiumIF	0.33PremiumGIF	0.33GIF
2	0.9	Very Good	E	VVS2	62.2	60.0	6.04	6.12	3.78	6289	Very GoodEVVS2	Very GoodVVS2	0.9Very GoodEVVS2	0.9EVVS2
3	0.42	Ideal	F	VS1	61.6	56.0	4.82	4.80	2.96	1082	IdealFVS1	IdealVS1	0.42IdealFVS1	0.42FVS1
4	0.31	Ideal	F	VVS1	60.4	59.0	4.35	4.43	2.65	779	IdealFVVS1	IdealVVS1	0.31IdealFVVS1	0.31FVVS1

- check is there any null data in the dataset using the function **isnull(). sum ()**

  We could still see **one value as zero** in the dataset**.**

```
carat         0
cut           0
color         0
clarity       0
depth         0
table         0
x             0
y             0
z             1
price         0
merge_col     0
merge_col2    0
merge_col3    0
merge_col4    0
dtype: int64
```

Approach3:

- Merge **three columns (i.e., carat, cut and clarity)**
- Take the **mean** of that.
- Replace **mean values of these with null values of above combination**.
- Merge column “merge_col5” value into **df_replace dataset**.
- Check top five records whether the column is added into the dataset or not using**. head()**

	carat	cut	color	clarity	depth	table	x	y	z	price	merge_col	merge_col2	merge_col3	merge_col4	merge_col5
0	0.3	Ideal	E	SI1	62.1	58.0	4.27	4.29	2.66	499	IdealESI1	IdealSI1	0.3IdealESI1	0.3ESI1	0.3IdealSI1
1	0.33	Premium	G	IF	60.8	58.0	4.42	4.46	2.70	984	PremiumGIF	PremiumIF	0.33PremiumGIF	0.33GIF	0.33PremiumIF
2	0.9	Very Good	E	VVS2	62.2	60.0	6.04	6.12	3.78	6289	Very GoodEVVS2	Very GoodVVS2	0.9Very GoodEVVS2	0.9EVVS2	0.9Very GoodVVS2
3	0.42	Ideal	F	VS1	61.6	56.0	4.82	4.80	2.96	1082	IdealFVS1	IdealVS1	0.42IdealFVS1	0.42FVS1	0.42IdealVS1
4	0.31	Ideal	F	VVS1	60.4	59.0	4.35	4.43	2.65	779	IdealFVVS1	IdealVVS1	0.31IdealFVVS1	0.31FVVS1	0.31IdealVVS1

- Using **group by function** replace null values of "merge_col4" combination with the mean ‘z’ value of "merge_col5".
- check is there **any null values in the dataset using the function .isnull().sum()**

```
carat         0
cut           0
color         0
clarity       0
depth         0
table         0
x             0
y             0
z             0
price         0
merge_col     0
merge_col2    0
merge_col3    0
merge_col4    0
merge_col5    0
dtype: int64
```

- Convert back **carat as float type using the function (code: astype(float))**
- Analysing the dimension of the dataset and the data type of variables in the dataset **(code: df.info ())**

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 26933 entries, 0 to 26966
Data columns (total 15 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   carat       26933 non-null  float64
 1   cut         26933 non-null  object
 2   color       26933 non-null  object
 3   clarity     26933 non-null  object
 4   depth       26933 non-null  float64
 5   table       26933 non-null  float64
 6   x           26933 non-null  float64
 7   y           26933 non-null  float64
 8   z           26933 non-null  float64
 9   price       26933 non-null  int64
 10  merge_col   26933 non-null  object
 11  merge_col2  26933 non-null  object
 12  merge_col3  26933 non-null  object
 13  merge_col4  26933 non-null  object
 14  merge_col5  26933 non-null  object
dtypes: float64(6), int64(1), object(8)
memory usage: 3.3+ MB
```

- Drop the columns **'merge_col','merge_col2','merge_col3','merge_col4','merge_col5' from the dataset (code: df_replace. drop ()).**

	carat	cut	color	clarity	depth	table	x	y	z	price
0	0.30	Ideal	E	SI1	62.1	58.0	4.27	4.29	2.66	499
1	0.33	Premium	G	IF	60.8	58.0	4.42	4.46	2.70	984
2	0.90	Very Good	E	VVS2	62.2	60.0	6.04	6.12	3.78	6289
3	0.42	Ideal	F	VS1	61.6	56.0	4.82	4.80	2.96	1082
4	0.31	Ideal	F	VVS1	60.4	59.0	4.35	4.43	2.65	779

- Check if the dataset has null values after processing **(code: df.isnull().sum()).**

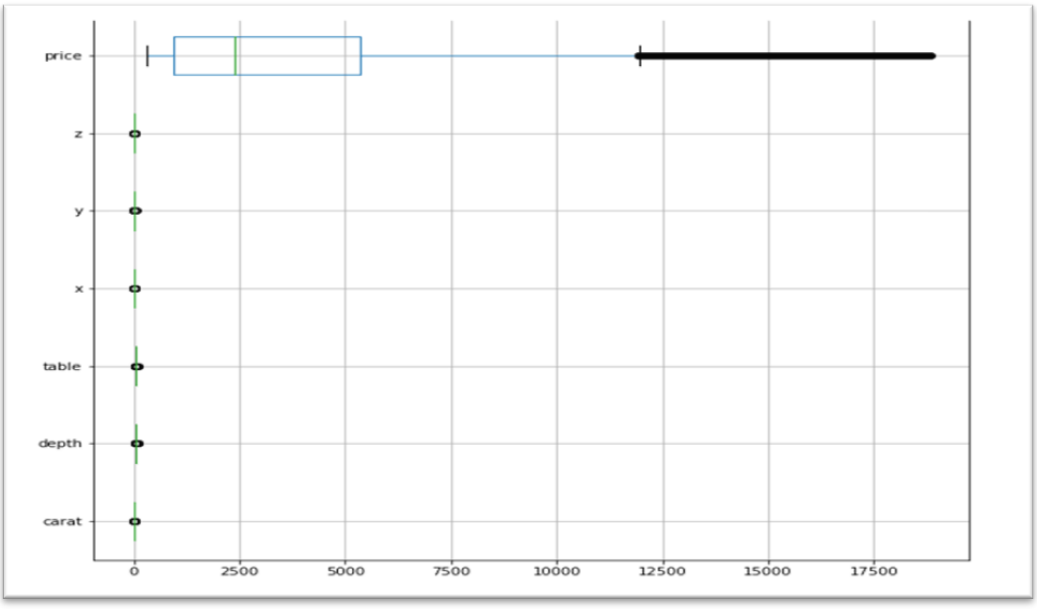
```
carat      0
cut        0
color      0
clarity    0
depth      0
table      0
x          0
y          0
z          0
price      0
dtype: int64
```

Now, we don't have any null values in x, y and z after using Approach 3 for imputation.

- Analysing the dimension of the dataset and the data type of variables in the dataset **(code: df.info())**

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 26933 entries, 0 to 26966
Data columns (total 10 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   carat    26933 non-null  float64
 1   cut      26933 non-null  object
 2   color    26933 non-null  object
 3   clarity  26933 non-null  object
 4   depth    26933 non-null  float64
 5   table    26933 non-null  float64
 6   x        26933 non-null  float64
 7   y        26933 non-null  float64
 8   z        26933 non-null  float64
 9   price    26933 non-null  int64
dtypes: float64(6), int64(1), object(3)
memory usage: 2.3+ MB
```

- **Check Outliers in the dataset using boxplot (code: df.boxplot()).**



❖ Check the **shape of the dataset (code: df.shape()**).

```
The number of columns (variables) in the dataset is 10
The number of rows (observations per variable) in the dataset is 26933
```

Insights from the above Results:

1. **34 records** were duplicate and considering the application of the dataset, we have **removed the same**. So, **final no. of records would be 26933**
2. **697 records** in numeric column name **"depth"** had the blank valued. by using **group by function**, we have replaced the null value with appropriate mean values (based on the appropriate combination, as explained above)
3. Further, we found **ZERO values in col x, y, z two, two and eight records respectively**. As, these columns are numeric in nature and represents length, width and height of the cubic zirconia. We have replaced these ZERO values with appropriate value. For which we used **group by function** we have replaced the null value with appropriate mean values (based on the appropriate combination, as explained above)
4. We have outliers in the dataset but **let's not remove at this point in time**.

Let's check for the basic measures of descriptive statistics:

Steps for analysis:

❖ Generating the descriptive statistics summary to include both numerical and categorical. **(code : df.describe(include='all').T)**

	count	unique	top	freq	mean	std	min	25%	50%	75%	max
carat	26933.0	NaN	NaN	NaN	0.79801	0.477237	0.2	0.4	0.7	1.05	4.5
cut	26933	5	Ideal	10805	NaN	NaN	NaN	NaN	NaN	NaN	NaN
color	26933	7	G	5653	NaN	NaN	NaN	NaN	NaN	NaN	NaN
clarity	26933	8	SI1	6565	NaN	NaN	NaN	NaN	NaN	NaN	NaN
depth	26933.0	NaN	NaN	NaN	61.745339	1.397077	50.8	61.1	61.8	62.5	73.6
table	26933.0	NaN	NaN	NaN	57.45595	2.232156	49.0	56.0	57.0	59.0	79.0
x	26933.0	NaN	NaN	NaN	5.729805	1.126301	3.73	4.71	5.69	6.55	10.23
y	26933.0	NaN	NaN	NaN	5.73356	1.164006	3.71	4.72	5.7	6.54	58.9
z	26933.0	NaN	NaN	NaN	3.539071	0.717594	1.07	2.9	3.52	4.04	31.8
price	26933.0	NaN	NaN	NaN	3937.52612	4022.551862	326.0	945.0	2375.0	5356.0	18818.0

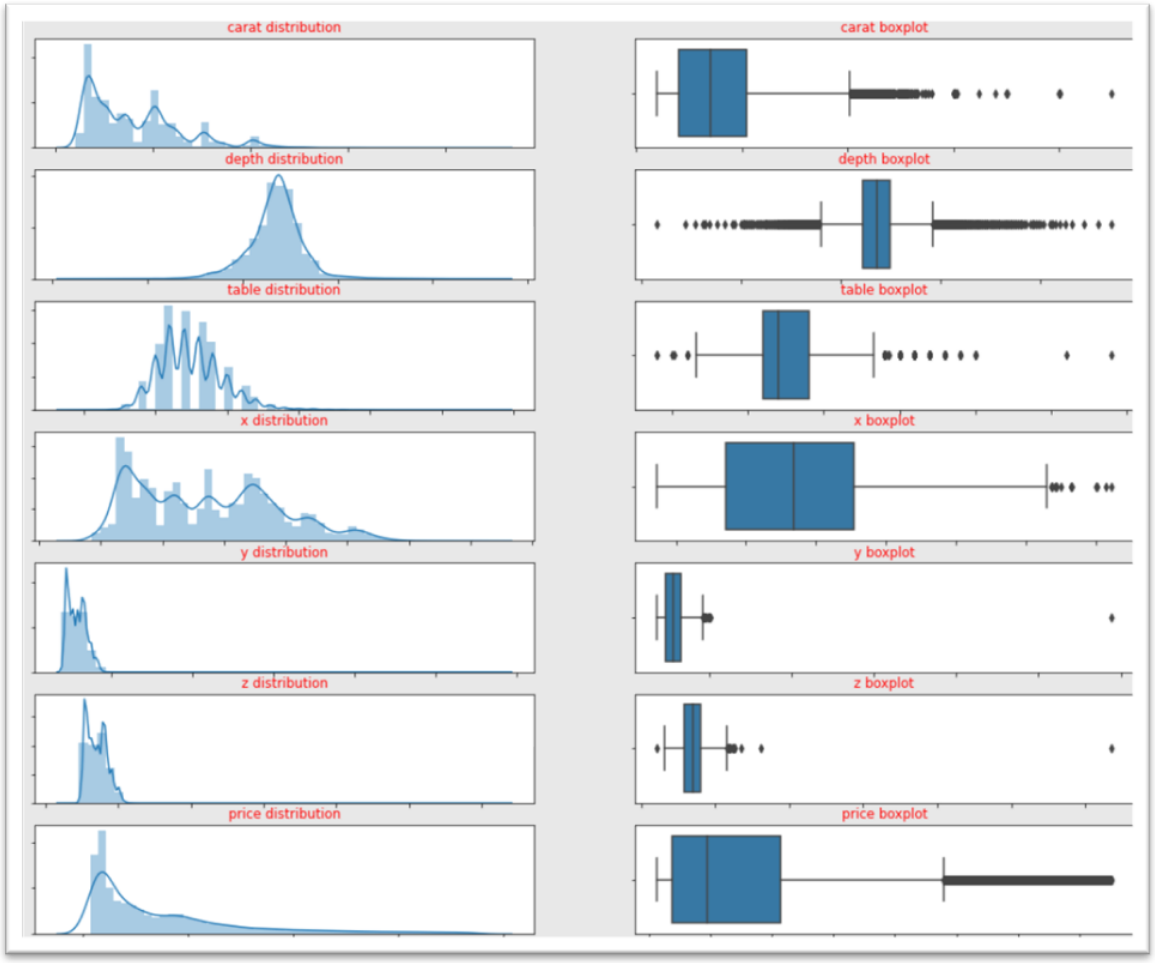
Insights from the above Results:

1. By looking at the dataset, it appears that there are outliers in the variables. The same is visible from the distribution of **5 values (min, 25%, 50%, 75% and max)**
2. Further, there **is a significant gap between mean and median for price**.
3. Cut, Color and Clarity are the categorical variables present in the dataset with **unique values of 5, 7 and 8 respectively**.

Let's have the visual representation of the distribution using

Univariate Analysis (Numerical Variables)

- Generating distribution and boxplot for **seven numerical variables**.
  (code: sns.distplot and sns.boxplot)
- Analysis of the **skewness of the four numerical variables**.
  (code: df.skew())



Insights from the above Results:

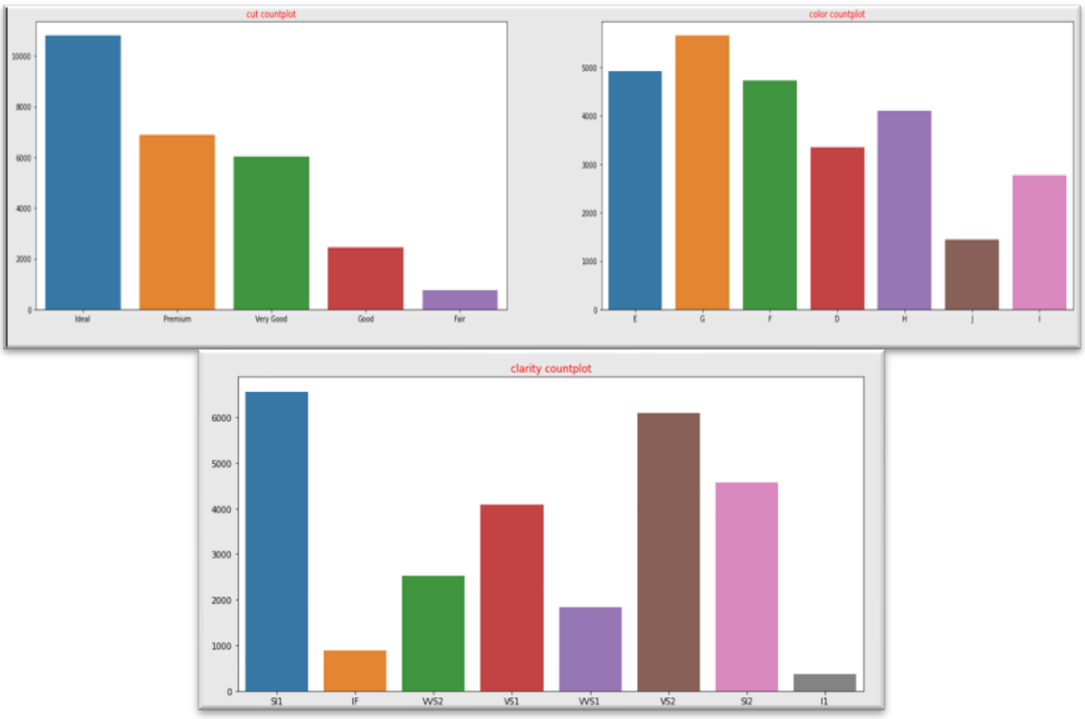
- Carat, table, x, y, z and price variables are not normally distributed where depth variable has similar to normal distribution trend.
- Box plot implies that all the seven variables have outliers. Let's not remove the outliers at this point in time.
- Carat variable is right skewed. It has many outliers present.
- Depth variable is near to normal distribution.

Univariate Analysis (Categorical Variables)

Check the data types of all the variables and get the **unique counts of all the 'Object' type variables. (code: value_counts()).**

```
cut
Ideal        10805
Premium       6886
Very Good     6027
Good          2435
Fair           780
Name: cut, dtype: int64


color
G    5653
E    4916
F    4723
H    4095
D    3341
I    2765
J    1440
Name: color, dtype: int64


clarity
SI1     6565
VS2     6093
SI2     4564
VS1     4087
VVS2    2530
VVS1    1839
IF       891
I1       364
Name: clarity, dtype: int64
```

- Generating **count plot for three categorical variables (code: sns.countplot())**



Insights from the above Results:

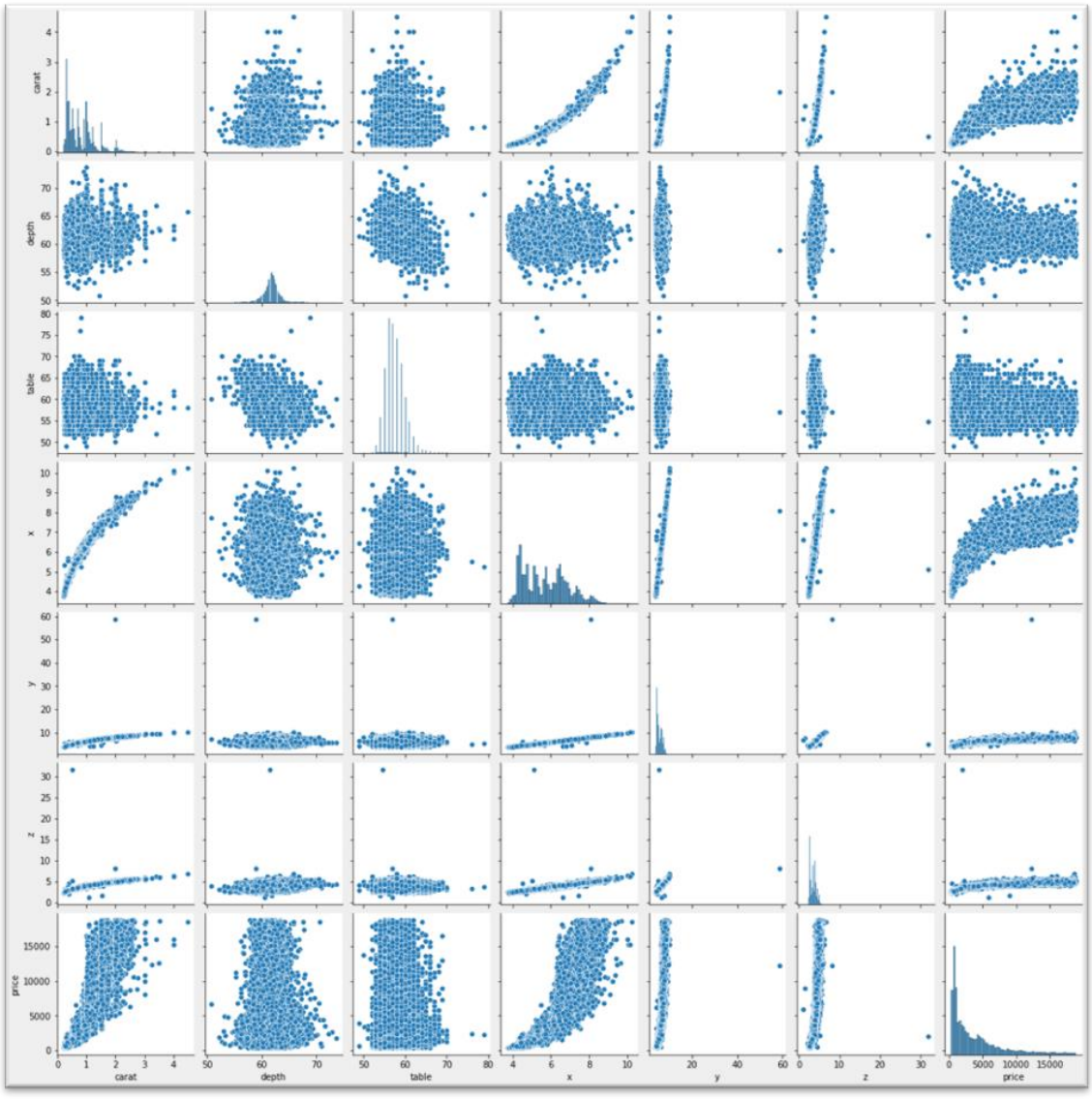
- **Cut type: Ideal has the highest count** followed by premium, very good, good and fair.
- **Color: G has the highest count** followed by E, F, H, D, I and J.
- **Clarity: SI1 has higher contribution** followed by VS2, SI2, VS1, VVS2, VVS1, IF and I1.
- **Color variable has 7 variants** and with G and J being the most and least no of observations
- **Cut variable has 5 variants** and Ideal type is highly present in the dataset
- **Clarity variable has 8 variants** with S1 being the most frequent in the dataset

Bivariate Analysis (Numerical variables Vs Numerical Variables)

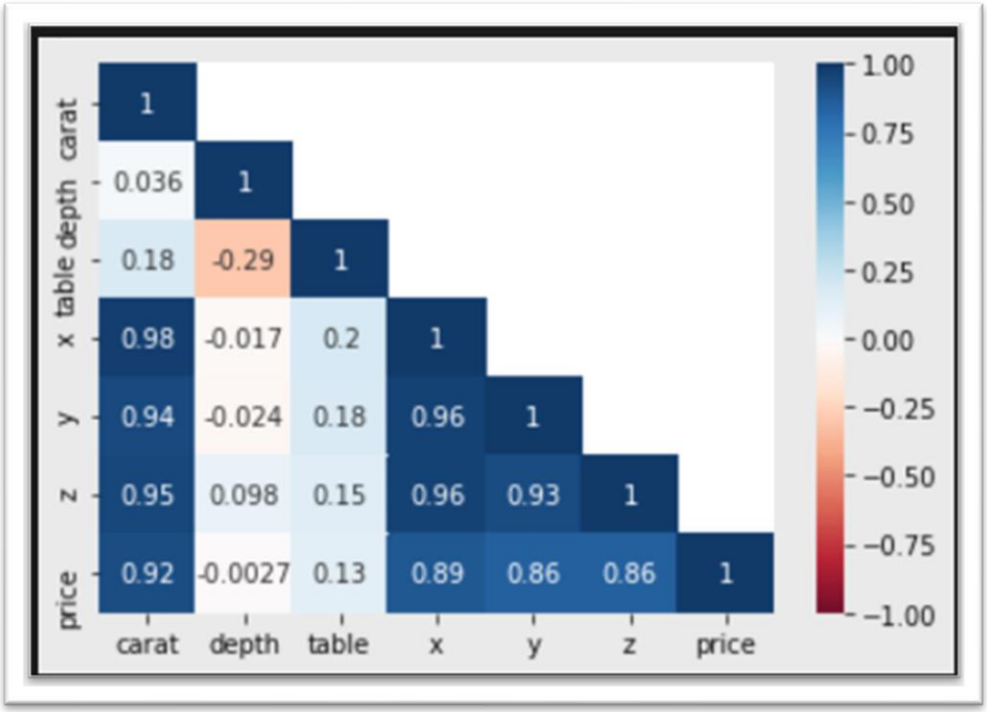
Steps for analysis:

- Generating **scatter plot.(code : sns.pairplot())** & Generating heat map.**(code: sns.heatmap())**

Scatter Plot:



Heatmap:



Insights from the above Results:

We have used scatter plot and heatmap and observed the below insights:

- Above **pair plot shows carat, x, y and z are positively correlated with target variable Price**.
- **Depth and table don't have strong correlation** with any dependent or with target variable Price.
- Presence of **multi-collinearity among independent variables - x, y, z and carat**.
- In scatter plot and heat map both shows that:
  - **(x, y, z) & carat**
  - **(x, y, z) & price and**
  - **carat & price variables**

  have the quite a **good positive correlation (more than 0.85)** *(i.e., if one increases other also increases)*
- Other variables are **not showing that strong relationships**.

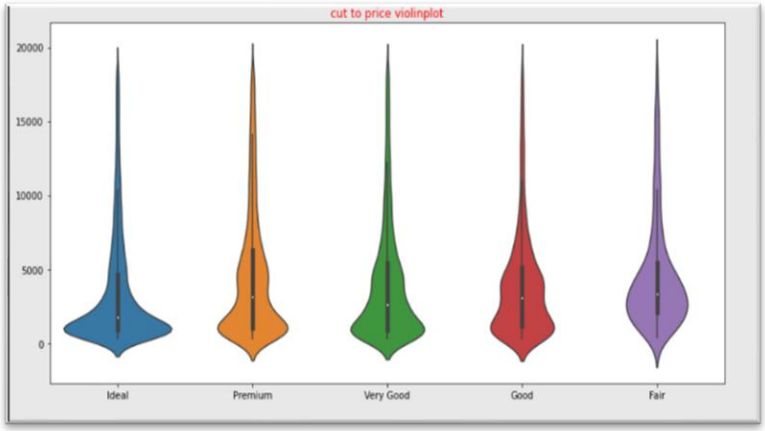
Bivariate Analysis (Categorical variables Vs Numerical Variables)

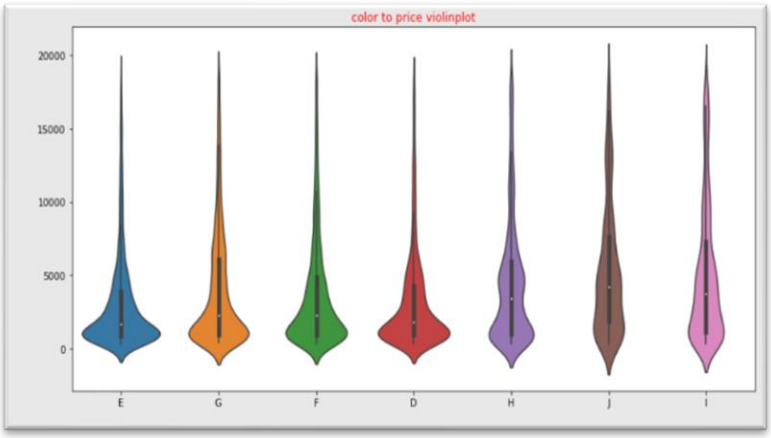
For Categorical vs Numerical variables **Violin plot, Box plot, strip plot and lmplot are the right options**.

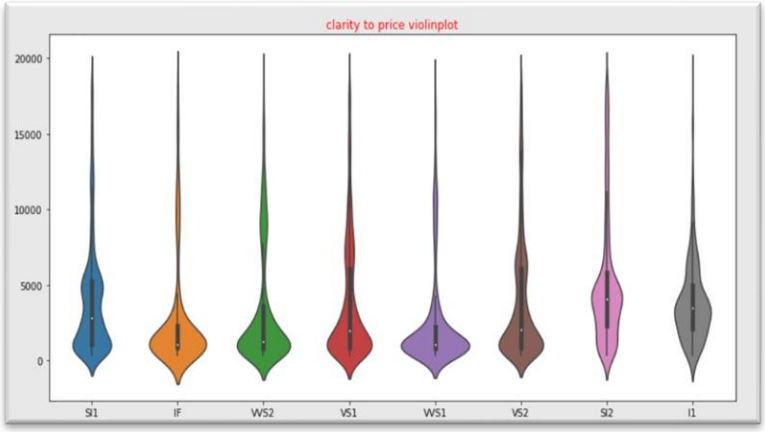
Steps for analysis:

- Generating violin plot. (code: sns.violinplot())

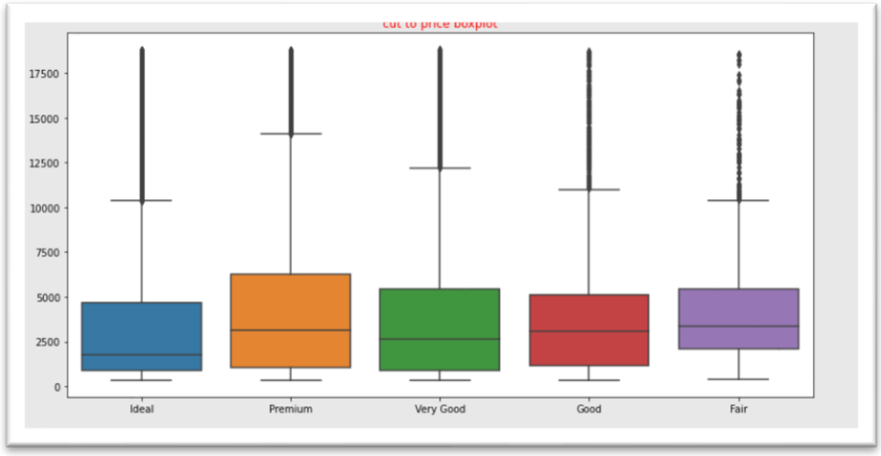
Results on all variables to “price”:

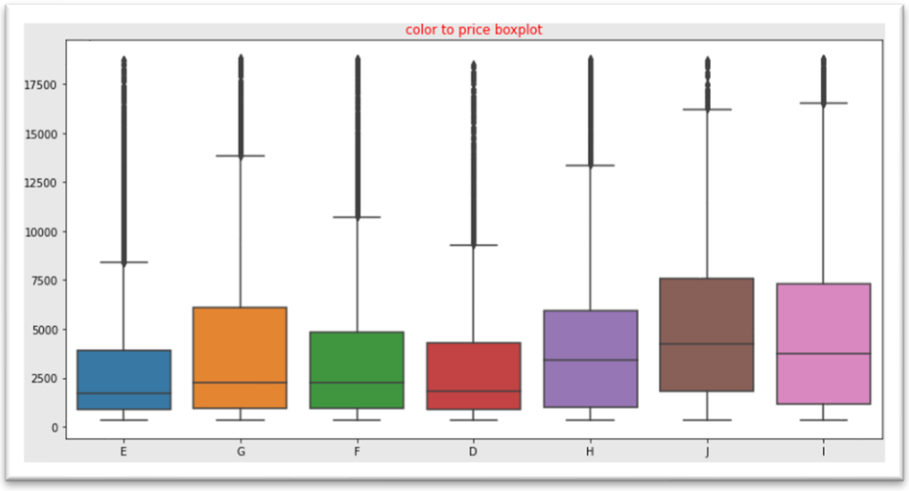


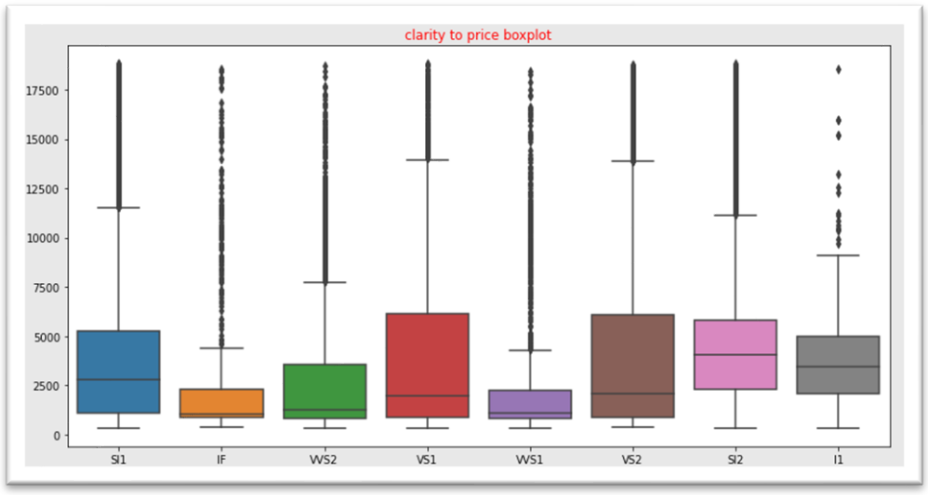




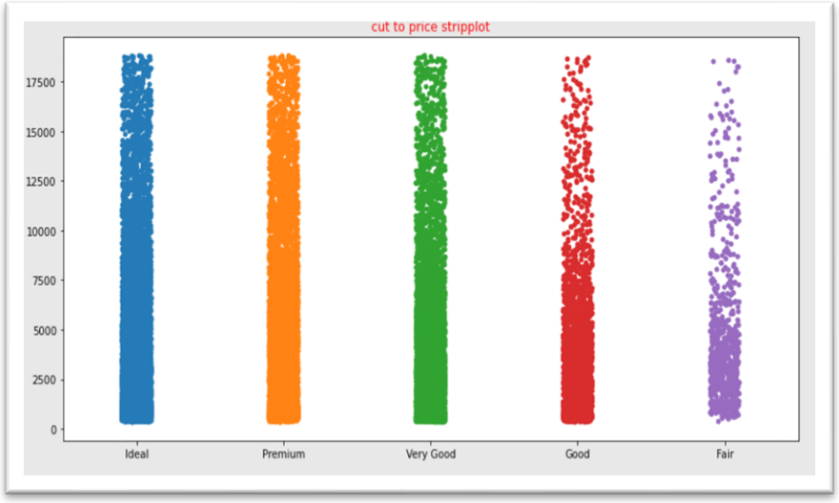
Box Plot:

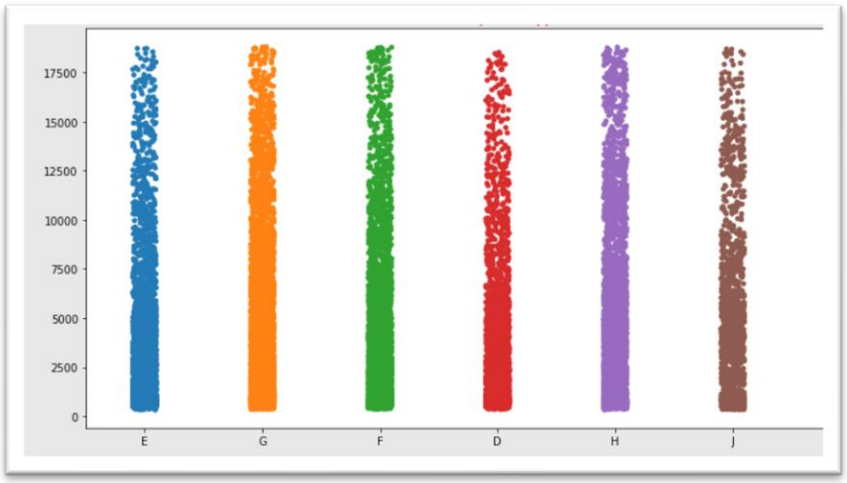

17500
15000
12500
10000
7500
5000
2500
0
Ideal
Premium
Very Good
Good
Fair

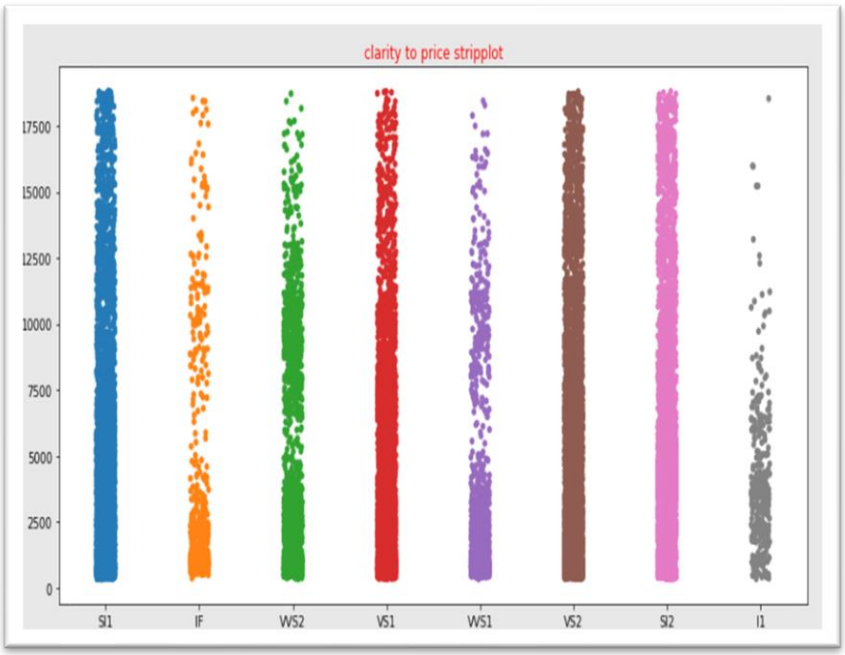

color to price boxplot
17500
15000
12500
10000
7500
5000
2500
0
E
G
F
D
H
J
I


clarity to price boxplot
17500
15000
12500
10000
7500
5000
2500
0
SI1
IF
VVS2
VS1
VVS1
VS2
SI2
I1

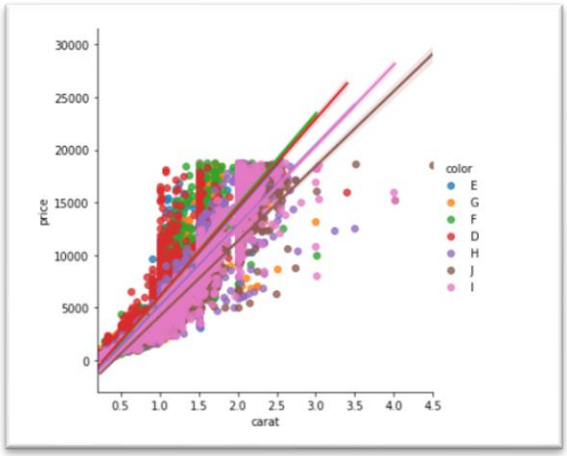
❖ **Strip plot for all categorical & numerical variables as "carat" (code: sns.stripplot()).**

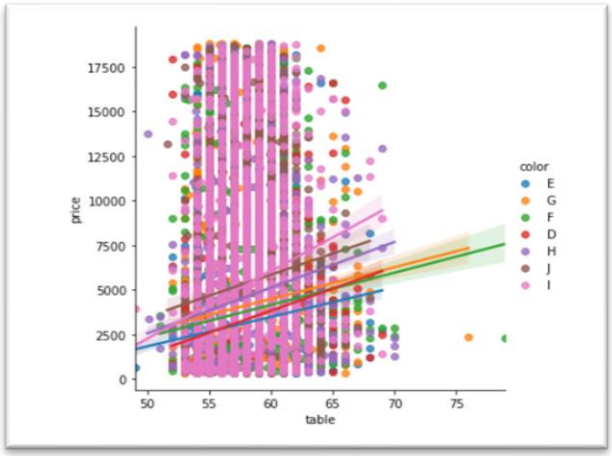


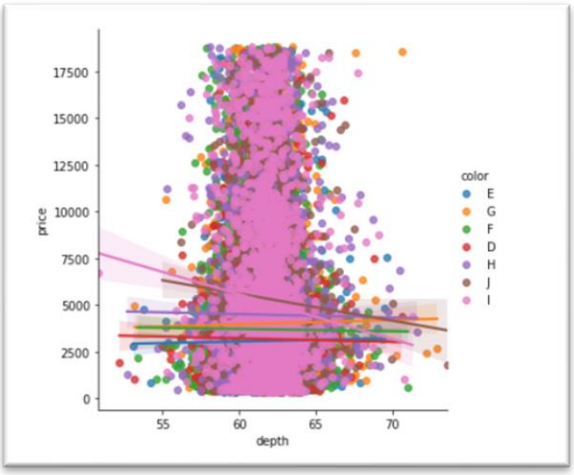


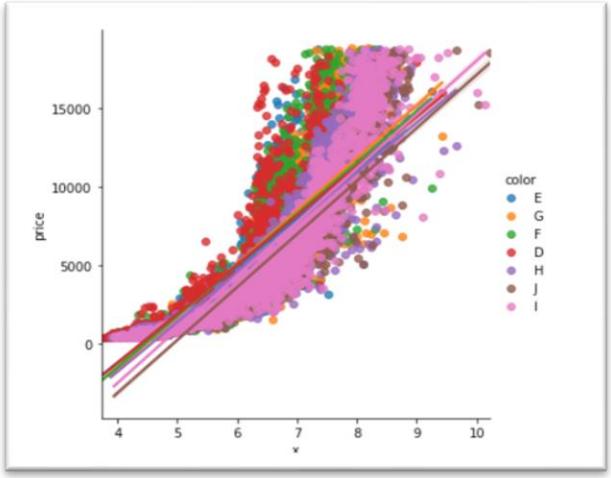


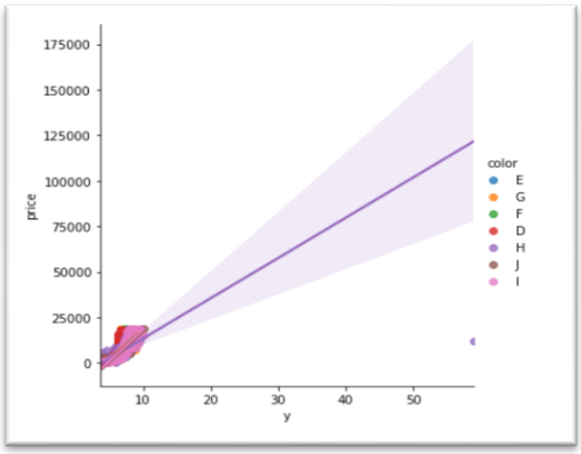
❖ **Lmplot: (code: sns.lmplot()).**

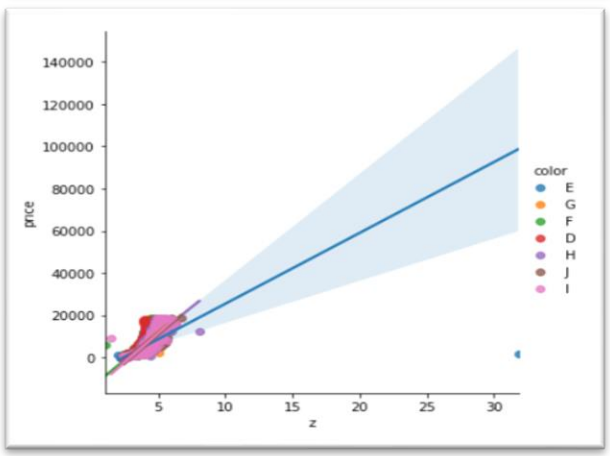












Insights from the above Results:

In Bivariate analysis (Categorical vs Numerical variables), we have used **violin plot, box plot, strip plot & lmplot** and compared **all categorical variables with price** and found:

- **Median price across cut type except IDEAL** cut type are almost same. *Further, price across the cut type is almost same*.
- **Median price across color is almost same**. *Further, price across the color is almost same*.
- **Median price across clarity is almost same**. *Further, price across the clarity is almost same except IJ1*.
- **Color variable has 7 variants and with G and J** being the most and least no of observations
- **Cut variable has 5 variants and Ideal type is highly** present in the dataset
- **Clarity variable has 8 variants with S1 being the most frequent in the dataset**

1.2 Use Full Data to develop a model to identify significant predictors. Check whether the proposed model is free of multicollinearity. Apply variable selection method as required. Show all intermediate models leading to the final model. Justify your choice of the final model. Which are the significant predictors?

Build various iterations of the Linear Regression model using appropriate variable selection techniques for the full data.

Step1:

- Treat the object variables appropriately by either creating dummy variables (One-Hot Encoding) or coding it up in an ordinal manner.
- Linear regression model does not take categorical values so that we have encoded categorical values to integer for better results.
- Categorical variables Cut and Clarity are encoded using ordinal manner. This is selected since the variables are ordinal and in case of linear regression this encoding helps to find the exact weightage each class possess.

Ordinal manner:

We are coding up the **'cut' variable** in an ordinal manner:

Cut	Ranking
Fair	1
Good	2
Very Good	3
Premium	4
Ideal	5

We are coding up the **'clarity' variable** in an ordinal manner:

Clarity	Ranking
I3	1
I2	2
I1	3
SI2	4
SI1	5
VS2	6
VS1	7
VVS2	8
VVS1	9
IF	10
FL	11

Converting the **cut & clarity variable to numeric using astype(int)**.

Convert the Color variable to numeric using **one hot encoder using get_dummies() and drop the first column**.

we used **.info function** to check the data types after conversion:

```
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   carat     26933 non-null  float64
 1   cut       26933 non-null  int32
 2   clarity   26933 non-null  int32
 3   depth     26933 non-null  float64
 4   table     26933 non-null  float64
 5   x         26933 non-null  float64
 6   y         26933 non-null  float64
 7   z         26933 non-null  float64
 8   price     26933 non-null  int64
 9   color_E   26933 non-null  uint8
 10  color_F   26933 non-null  uint8
 11  color_G   26933 non-null  uint8
 12  color_H   26933 non-null  uint8
 13  color_I   26933 non-null  uint8
 14  color_J   26933 non-null  uint8
dtypes: float64(6), int32(2), int64(1), uint8(6)
memory usage: 3.0 MB
```

<u>Step 2:</u>

Data Scaling & Model building:

We scaled the data using minmax scaler using the function **MinMaxScaler()**

First 2 records from the scaled Data Set:

	carat	cut	clarity	depth	table	x	y	z	price	color_E	color_F	color_G	color_H	color_I	color_J
0	0.023256	5	5	0.495614	0.3	0.083077	0.010509	0.051741	0.009355	1	0	0	0	0	0
1	0.030233	4	10	0.438596	0.3	0.106154	0.013589	0.053043	0.035583	0	0	1	0	0	0

Identify dependent variables and independent variables and put it into Y and X dataset respectively.

Independent Variables:

X Dataset for the first 5 records

	carat	cut	clarity	depth	table	x	y	z	color_E	color_F	color_G	color_H	color_I	color_J
0	0.023256	5	5	0.495614	0.300000	0.083077	0.010509	0.051741	1	0	0	0	0	0
1	0.030233	4	10	0.438596	0.300000	0.106154	0.013589	0.053043	0	0	1	0	0	0
2	0.162791	3	8	0.500000	0.366667	0.355385	0.043667	0.088187	1	0	0	0	0	0
3	0.051163	5	7	0.473684	0.233333	0.167692	0.019750	0.061503	0	1	0	0	0	0
4	0.025581	5	9	0.421053	0.333333	0.095385	0.013046	0.051416	0	1	0	0	0	0

Dependent Variable:

Y – Dataset for the first 5 records.

```
0    0.009355
1    0.035583
2    0.322464
3    0.040883
4    0.024497
```

MODEL BUILDING:

MODEL 1

Now, Let's build the model using **statsmodels**, for the detailed statistics using function **model. Summary().**

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.911
Model:                            OLS   Adj. R-squared:                  0.911
Method:                 Least Squares   F-statistic:                 1.969e+04
Date:                Sun, 11 Apr 2021   Prob (F-statistic):               0.00
Time:                        18:39:52   Log-Likelihood:                 35453.
No. Observations:               26933   AIC:                        -7.088e+04
Df Residuals:                   26918   BIC:                        -7.075e+04
Df Model:                          14
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         -0.1334      0.007    -19.862      0.000      -0.147      -0.120
carat          2.6082      0.018    145.832      0.000       2.573       2.643
cut            0.0061      0.000     14.010      0.000       0.005       0.007
clarity        0.0262      0.000     97.539      0.000       0.026       0.027
depth         -0.1093      0.008    -13.381      0.000      -0.125      -0.093
table         -0.0478      0.007     -7.112      0.000      -0.061      -0.035
x             -0.3642      0.016    -22.957      0.000      -0.395      -0.333
y              0.0866      0.070      1.240      0.215      -0.050       0.223
z             -0.0825      0.068     -1.214      0.225      -0.216       0.051
color_E       -0.0122      0.001     -8.398      0.000      -0.015      -0.009
color_F       -0.0172      0.001    -11.673      0.000      -0.020      -0.014
color_G       -0.0288      0.001    -20.001      0.000      -0.032      -0.026
color_H       -0.0564      0.002    -36.682      0.000      -0.059      -0.053
color_I       -0.0818      0.002    -47.729      0.000      -0.085      -0.078
color_J       -0.1261      0.002    -59.941      0.000      -0.130      -0.122
==============================================================================
Omnibus:                     6341.021   Durbin-Watson:                   2.012
Prob(Omnibus):                  0.000   Jarque-Bera (JB):           343389.495
Skew:                          -0.186   Prob(JB):                         0.00
Kurtosis:                      20.489   Cond. No.                     1.39e+03
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.39e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

Check for multicollinearity in the predictor variables using Variance Inflation Factor (VIF).

Calculate the Variance factor using the function **checkVIF().**

	Features	VIF
0	const	288.72
6	x	48.33
1	carat	25.20
8	z	16.08
7	y	13.87
11	color_G	2.20
9	color_E	2.03
10	color_F	2.02
12	color_H	1.95
13	color_I	1.73
4	depth	1.60
5	table	1.60
2	cut	1.50
14	color_J	1.43
3	clarity	1.25

Model 1 Insight:

The **p-value of the 'z' seems to be higher than the significance value of 0.05** and it is insignificant compare to the other independent variables.

MODEL 2

We have re-run the model **after dropping the variable 'z' and found the below summary**.

Model 2 Evaluation Summary using the **function model.summary().**

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.911
Model:                            OLS   Adj. R-squared:                  0.911
Method:                 Least Squares   F-statistic:                 2.121e+04
Date:                Sun, 11 Apr 2021   Prob (F-statistic):               0.00
Time:                        18:39:53   Log-Likelihood:                 35452.
No. Observations:               26933   AIC:                        -7.088e+04
Df Residuals:                   26919   BIC:                        -7.076e+04
Df Model:                          13
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         -0.1352      0.007    -20.598      0.000      -0.148      -0.122
carat          2.6079      0.018    145.829      0.000       2.573       2.643
cut            0.0061      0.000     14.036      0.000       0.005       0.007
clarity        0.0262      0.000     97.531      0.000       0.026       0.027
depth         -0.1128      0.008    -14.746      0.000      -0.128      -0.098
table         -0.0476      0.007     -7.081      0.000      -0.061      -0.034
x             -0.3735      0.014    -26.848      0.000      -0.401      -0.346
y              0.0759      0.069      1.095      0.273      -0.060       0.212
color_E       -0.0123      0.001     -8.410      0.000      -0.015      -0.009
color_F       -0.0172      0.001    -11.673      0.000      -0.020      -0.014
color_G       -0.0288      0.001    -20.001      0.000      -0.032      -0.026
color_H       -0.0564      0.002    -36.680      0.000      -0.059      -0.053
color_I       -0.0818      0.002    -47.726      0.000      -0.085      -0.078
color_J       -0.1261      0.002    -59.942      0.000      -0.130      -0.122
==============================================================================
Omnibus:                     6340.806   Durbin-Watson:                   2.012
Prob(Omnibus):                  0.000   Jarque-Bera (JB):           343418.523
Skew:                          -0.186   Prob(JB):                         0.00
Kurtosis:                      20.489   Cond. No.                     1.32e+03
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.32e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

Check for multicollinearity in the predictor variables using Variance Inflation Factor (VIF).

	Features	VIF
0	const	275.51
6	x	37.16
1	carat	25.19
7	y	13.64
10	color_G	2.20
8	color_E	2.03
9	color_F	2.02
11	color_H	1.95
12	color_I	1.73
5	table	1.60
2	cut	1.50
13	color_J	1.43
4	depth	1.40
3	clarity	1.25

Model 2 Insight:

p-value of **'y' seems to be higher than the significance value of 0.05 and it is insignificant variable in presence of other independent variables**.

MODEL 3

We have re-run the model **after dropping the variable 'y' and found the below summary.**

Model 3 Evaluation Summary using the function model.summary()

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.911
Model:                            OLS   Adj. R-squared:                  0.911
Method:                 Least Squares   F-statistic:                 2.297e+04
Date:                Sun, 11 Apr 2021   Prob (F-statistic):               0.00
Time:                        18:39:53   Log-Likelihood:                 35452.
No. Observations:               26933   AIC:                        -7.088e+04
Df Residuals:                   26920   BIC:                        -7.077e+04
Df Model:                          12
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         -0.1348      0.007    -20.569      0.000      -0.148      -0.122
carat          2.6084      0.018    145.912      0.000       2.573       2.643
cut            0.0061      0.000     14.012      0.000       0.005       0.007
clarity        0.0262      0.000     97.545      0.000       0.026       0.027
depth         -0.1131      0.008    -14.811      0.000      -0.128      -0.098
table         -0.0479      0.007     -7.134      0.000      -0.061      -0.035
x             -0.3649      0.011    -31.738      0.000      -0.387      -0.342
color_E       -0.0123      0.001     -8.410      0.000      -0.015      -0.009
color_F       -0.0172      0.001    -11.674      0.000      -0.020      -0.014
color_G       -0.0288      0.001    -20.004      0.000      -0.032      -0.026
color_H       -0.0564      0.002    -36.677      0.000      -0.059      -0.053
color_I       -0.0818      0.002    -47.729      0.000      -0.085      -0.078
color_J       -0.1261      0.002    -59.944      0.000      -0.130      -0.122
==============================================================================
Omnibus:                     6344.783   Durbin-Watson:                   2.012
Prob(Omnibus):                  0.000   Jarque-Bera (JB):           343933.505
Skew:                          -0.187   Prob(JB):                         0.00
Kurtosis:                      20.503   Cond. No.                         405.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

Check for multicollinearity in the predictor variables using Variance Inflation Factor (VIF)

(code: checkVIF()).

	Features	VIF
0	const	274.77
6	x	25.38
1	carat	25.17
9	color_G	2.20
7	color_E	2.03
8	color_F	2.02
10	color_H	1.95
11	color_I	1.73
5	table	1.60
2	cut	1.50
12	color_J	1.43
4	depth	1.40
3	clarity	1.25

Model 3 Insight:

From the above output the **'x' and carat variables have high VIF value. (shows that x has high multicollinearity.)**

MODEL 4

We have re-run the model after **dropping the variable 'x', first, and found the below summary**.

Model 4 Evaluation Summary using the function model.summary().

OLS Regression Results

Dep. Variable:	price	R-squared:	0.908
Model:	OLS	Adj. R-squared:	0.908
Method:	Least Squares	F-statistic:	2.407e+04
Date:	Sun, 11 Apr 2021	Prob (F-statistic):	0.00
Time:	18:39:53	Log-Likelihood:	34957.
No. Observations:	26933	AIC:	-6.989e+04
Df Residuals:	26921	BIC:	-6.979e+04
Df Model:	11		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-0.2074	0.006	-33.142	0.000	-0.220	-0.195
carat	2.0560	0.004	495.646	0.000	2.048	2.064
cut	0.0063	0.000	14.271	0.000	0.005	0.007
clarity	0.0276	0.000	102.542	0.000	0.027	0.028
depth	-0.0558	0.008	-7.387	0.000	-0.071	-0.041
table	-0.0466	0.007	-6.812	0.000	-0.060	-0.033
color_E	-0.0123	0.001	-8.279	0.000	-0.015	-0.009
color_F	-0.0189	0.002	-12.567	0.000	-0.022	-0.016
color_G	-0.0305	0.001	-20.811	0.000	-0.033	-0.028
color_H	-0.0564	0.002	-36.020	0.000	-0.059	-0.053
color_I	-0.0805	0.002	-46.171	0.000	-0.084	-0.077
color_J	-0.1237	0.002	-57.790	0.000	-0.128	-0.120

Omnibus:	5325.396	Durbin-Watson:	2.008
Prob(Omnibus):	0.000	Jarque-Bera (JB):	105948.900
Skew:	0.419	Prob(JB):	0.00
Kurtosis:	12.680	Cond. No.	196.

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

Check for multicollinearity in the predictor variables using Variance Inflation Factor (VIF).

(code: checkVIF()).

	Features	VIF
0	const	241.35
8	color_G	2.20
6	color_E	2.03
7	color_F	2.01
9	color_H	1.95
10	color_I	1.73
5	table	1.60
2	cut	1.50
11	color_J	1.43
4	depth	1.32
1	carat	1.31
3	clarity	1.22

Step 3:

Residual, Prediction & Evaluation of Model:

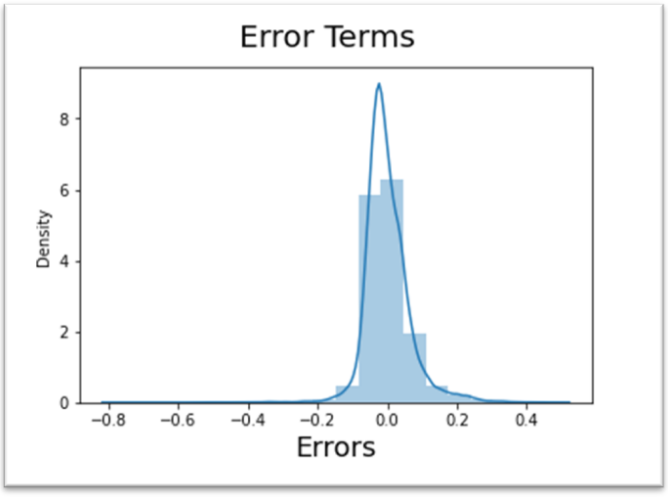
- Let us predict the Y variable based on the predictor variables.

  (Code: sm.OLS(Y,X).fit() , & lm.predict(X_new))

```
Y_pred

0       -0.043627
1        0.087579
2        0.310183
3        0.066770
4        0.067749
            ...
26962    0.318621
26963    0.070000
26964    0.079078
26965    0.009550
26966    0.293066
Length: 26933, dtype: float64
```

- Plot the histogram of the **error terms (code: sns.distplot())**.

Insights: Error Terms seem to be approximately normally distributed, so the assumption on the linear modelling seems to be fulfilled.



Step 4:

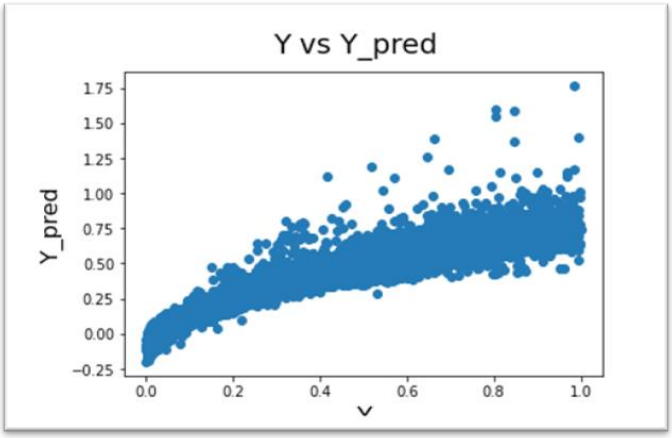
Model Metrics

Import libraries such as r2_score from Sklearn and calculate accuracy

Accuracy: `0.9077137570747361`

- Plotting Y and Y_pred to understand the spread. (code: plt.scatter()).

Graph shows scatter plot between actual price vs predicted for test dataset and it is linear and diagonal showing that the forecast accuracy is good.



Calculate MSE: Import the libraires from Sklearn (mean_square_error & mean_absolute_error).

MSE: (code: mean_squared_error ()) `0.004366730355127467`

Calculate RMSE: The Root Mean Square Error which is one of the metrics how good the Linear Regression model is.

RMSE: (code: sqrt(MSE_fulldata)) `0.06608124056891991`

Accuracy: (code: r2_score (Y, Y_pred)) `0.9077137570747361`

- Comparing all the metrics such as Accuracy, MSE and RMSE.

	Full Data
Accuracy	0.908
MSE	0.004
RMSE	0.066

Final Insights for Question 2:

- carat, cut, clarity, color, depth and table are the significant variables to predict the price variable.
- R-squared and Adjusted R-squared **(extent of fit) - 0.908 and 0.908 - ~91% variance explained**.
- F-stats and Prob(F-stats) (overall model fit) **- 2.407e+04 and 0.00 - Model is significant and explained ~91% variance.**
- p-values - p-values for all the coefficients seem to be **less than the significance level of 0.05**. - meaning that all the predictors are **statistically significant**.
- Model Metrics:
    - **MSE: 0.004**
    - **RMSE: 0.066**

1.3 Split the data into training (70%) and test (30%). Build the various iterations of the Linear Regression models on the training data and use those models to predict on the test data using appropriate model evaluation metrics.

1.3.1 Alternatively, if prediction accuracy of the price is the only objective, then you may want to divide the data into a training and a test set, chosen randomly, and use the training set to develop a model and test set to validate your model.

1.3.2 Use the models developed in Part (2) to compare accuracy in training and test sets. Compare the final model of Part (2) and the proposed one in Part (3). Which model provides the most accurate prediction? If the model found in Part (2) is different from the proposed model in Part (3), give an explanation.

1.3.1 Alternatively, if prediction accuracy of the price is the only objective, then you may want to divide the data into a training and a test set, chosen randomly, and use the training set to develop a model and test set to validate your model.

Train-Test Split

Splitting data into training and test set (test_size=.30, random_state=1)

- This is done so that 70% of the dataset is used for building the model and remaining 30% used to test the model. We used the train_test_split function.
- Check the dimensions of the Train and Test Data using shape ().

```
df_train: (18853, 15)
df_test: (8080, 15)
Total Count: 26933
```

Df_train has 18853 rows and 15 columns, Df_test has 8080 rows and 15 columns.

Total no of records is 26933.

Data Scaling

- Import the MinMaxScaler libraries from Sklearn.
- Apply the data scaling using minmax scaler function for the variables **("carat", depth"," table, 'x', 'y', 'z' and price)** (code: MinMaxScaler () and scaler.fit_transform())

Check the first two records using the function head(2) after applying the scaling.

	carat	cut	clarity	depth	table	x	y	z	price	color_E	color_F	color_G	color_H	color_I	color_J
3211	0.440	5	4	0.443	0.267	0.703	0.081	0.128	0.886	0	0	0	1	0	0
1962	0.091	2	5	0.584	0.233	0.245	0.030	0.076	0.074	0	0	0	0	0	0

Dividing the data into dependent (Y) and independent variables (X).

```
#Dividing data into X and y variables
Y_train = df_train.pop('price')
X_train = df_train
```

MODEL BUILDING:

MODEL 1

Check the Model 1 summary using function model. Summary ()

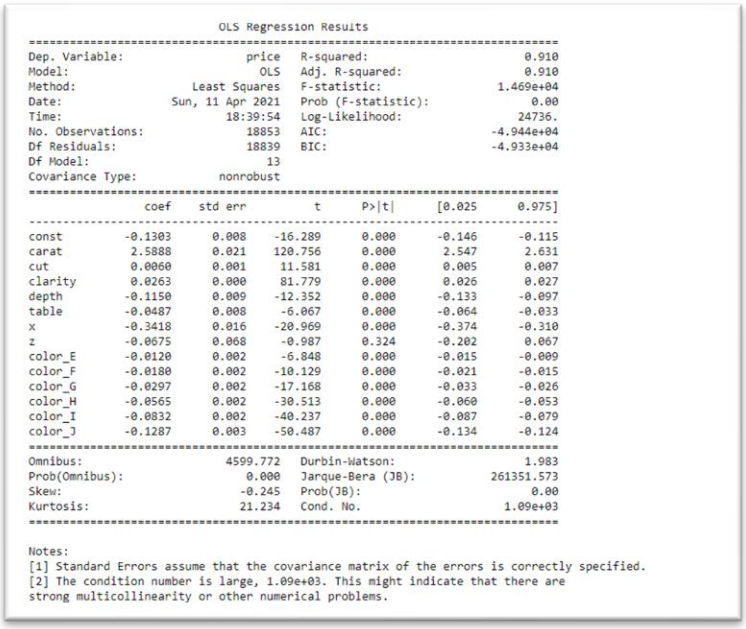
```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.910
Model:                            OLS   Adj. R-squared:                  0.910
Method:                 Least Squares   F-statistic:                 1.364e+04
Date:                Sun, 11 Apr 2021   Prob (F-statistic):               0.00
Time:                        18:39:54   Log-Likelihood:                 24736.
No. Observations:               18853   AIC:                        -4.944e+04
Df Residuals:                   18838   BIC:                        -4.932e+04
Df Model:                          14
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         -0.1304      0.008    -16.296      0.000      -0.146      -0.115
carat          2.5885      0.021    120.698      0.000       2.546       2.631
cut            0.0060      0.001     11.588      0.000       0.005       0.007
clarity        0.0263      0.000     81.774      0.000       0.026       0.027
depth         -0.1146      0.009    -12.276      0.000      -0.133      -0.096
table         -0.0485      0.008     -6.047      0.000      -0.064      -0.033
x             -0.3450      0.018    -19.514      0.000      -0.380      -0.310
y              0.0328      0.071      0.465      0.642      -0.105       0.171
z             -0.0714      0.069     -1.036      0.300      -0.206       0.064
color_E       -0.0120      0.002     -6.848      0.000      -0.015      -0.009
color_F       -0.0180      0.002    -10.128      0.000      -0.021      -0.015
color_G       -0.0297      0.002    -17.167      0.000      -0.033      -0.026
color_H       -0.0565      0.002    -30.514      0.000      -0.060      -0.053
color_I       -0.0832      0.002    -40.235      0.000      -0.087      -0.079
color_J       -0.1287      0.003    -50.485      0.000      -0.134      -0.124
==============================================================================
Omnibus:                     4598.015   Durbin-Watson:                   1.983
Prob(Omnibus):                  0.000   Jarque-Bera (JB):           261137.776
Skew:                          -0.245   Prob(JB):                         0.00
Kurtosis:                      21.226   Cond. No.                     1.17e+03
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.17e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

Model 1 Insights:

P-Value of '`y`' (**0.642**) seems to be higher than the significance value of 0.05, and it is insignificant compare to other independent variables.

MODEL 2

Build the new Model 2 by dropping the 'y' variable using the function drop ()).



OLS Regression Results

Dep. Variable:	price	R-squared:	0.910
Model:	OLS	Adj. R-squared:	0.910
Method:	Least Squares	F-statistic:	1.469e+04
Date:	Sun, 11 Apr 2021	Prob (F-statistic):	0.00
Time:	18:39:54	Log-Likelihood:	24736.
No. Observations:	18853	AIC:	-4.944e+04
Df Residuals:	18839	BIC:	-4.933e+04
Df Model:	13		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-0.1303	0.008	-16.289	0.000	-0.146	-0.115
carat	2.5888	0.021	120.756	0.000	2.547	2.631
cut	0.0060	0.001	11.581	0.000	0.005	0.007
clarity	0.0263	0.000	81.779	0.000	0.026	0.027
depth	-0.1150	0.009	-12.352	0.000	-0.133	-0.097
table	-0.0487	0.008	-6.067	0.000	-0.064	-0.033
x	-0.3418	0.016	-20.969	0.000	-0.374	-0.310
z	-0.0675	0.068	-0.987	0.324	-0.202	0.067
color_E	-0.0120	0.002	-6.848	0.000	-0.015	-0.009
color_F	-0.0180	0.002	-10.129	0.000	-0.021	-0.015
color_G	-0.0297	0.002	-17.168	0.000	-0.033	-0.026
color_H	-0.0565	0.002	-30.513	0.000	-0.060	-0.053
color_I	-0.0832	0.002	-40.237	0.000	-0.087	-0.079
color_J	-0.1287	0.003	-50.487	0.000	-0.134	-0.124

Omnibus:	4599.772	Durbin-Watson:	1.983
Prob(Omnibus):	0.000	Jarque-Bera (JB):	261351.573
Skew:	-0.245	Prob(JB):	0.00
Kurtosis:	21.234	Cond. No.	1.09e+03

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.09e+03. This might indicate that there are
strong multicollinearity or other numerical problems.

Model 2 Insights:

From the above result we can observe that the p-vale of '`z`' seems to be higher than the significance value of 0.05 and it is insignificant in presence of other variables.

MODEL 3

Build the new Model 3 by dropping the 'z' variable using the function drop ()).

OLS Regression Results

Dep. Variable:	price	R-squared:	0.910
Model:	OLS	Adj. R-squared:	0.910
Method:	Least Squares	F-statistic:	1.592e+04
Date:	Sun, 11 Apr 2021	Prob (F-statistic):	0.00
Time:	18:39:54	Log-Likelihood:	24735.
No. Observations:	18853	AIC:	-4.944e+04
Df Residuals:	18840	BIC:	-4.934e+04
Df Model:	12		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-0.1318	0.008	-16.783	0.000	-0.147	-0.116
carat	2.5885	0.021	120.754	0.000	2.546	2.630
cut	0.0060	0.001	11.607	0.000	0.005	0.007
clarity	0.0263	0.000	81.773	0.000	0.026	0.027
depth	-0.1177	0.009	-13.223	0.000	-0.135	-0.100
table	-0.0484	0.008	-6.038	0.000	-0.064	-0.033
x	-0.3504	0.014	-25.427	0.000	-0.377	-0.323
color_E	-0.0120	0.002	-6.860	0.000	-0.015	-0.009
color_F	-0.0180	0.002	-10.128	0.000	-0.021	-0.015
color_G	-0.0297	0.002	-17.167	0.000	-0.033	-0.026
color_H	-0.0565	0.002	-30.512	0.000	-0.060	-0.053
color_I	-0.0832	0.002	-40.236	0.000	-0.087	-0.079
color_J	-0.1287	0.003	-50.489	0.000	-0.134	-0.124

Omnibus:	4599.042	Durbin-Watson:	1.983
Prob(Omnibus):	0.000	Jarque-Bera (JB):	261309.243
Skew:	-0.245	Prob(JB):	0.00
Kurtosis:	21.232	Cond. No.	404.

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

Calculating the VIF for model 3 using the function **checkVIF ():**

	Features	VIF
0	const	273.880
6	x	25.190
1	carat	25.020
9	color_G	2.230
7	color_E	2.040
8	color_F	2.020
10	color_H	1.960
11	color_I	1.720
5	table	1.570
2	cut	1.490
12	color_J	1.430
4	depth	1.400
3	clarity	1.250

Model 3 Insights:

Dropping the `'x'` because of high VIF value of 25.190 (shows that x has high multicollinearity) using the drop () function.

MODEL 4

Build the new Model by dropping the 'x' variable using the function drop ()).

OLS Regression Results

Dep. Variable:	price	R-squared:	0.907
Model:	OLS	Adj. R-squared:	0.907
Method:	Least Squares	F-statistic:	1.673e+04
Date:	Sun, 11 Apr 2021	Prob (F-statistic):	0.00
Time:	18:39:55	Log-Likelihood:	24417.
No. Observations:	18853	AIC:	-4.881e+04
Df Residuals:	18841	BIC:	-4.872e+04
Df Model:	11		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-0.2026	0.007	-27.109	0.000	-0.217	-0.188
carat	2.0578	0.005	413.880	0.000	2.048	2.068
cut	0.0063	0.001	11.978	0.000	0.005	0.007
clarity	0.0276	0.000	85.782	0.000	0.027	0.028
depth	-0.0636	0.009	-7.240	0.000	-0.081	-0.046
table	-0.0454	0.008	-5.564	0.000	-0.061	-0.029
color_E	-0.0121	0.002	-6.801	0.000	-0.016	-0.009
color_F	-0.0194	0.002	-10.739	0.000	-0.023	-0.016
color_G	-0.0314	0.002	-17.850	0.000	-0.035	-0.028
color_H	-0.0562	0.002	-29.864	0.000	-0.060	-0.053
color_I	-0.0821	0.002	-39.026	0.000	-0.086	-0.078
color_J	-0.1266	0.003	-48.850	0.000	-0.132	-0.121

Omnibus:	3714.069	Durbin-Watson:	1.987
Prob(Omnibus):	0.000	Jarque-Bera (JB):	82496.317
Skew:	0.364	Prob(JB):	0.00
Kurtosis:	13.222	Cond. No.	193.

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

Calculating the VIF for Model 4 using the function checkVIF ():

All the below VIF values are **less than 5 and they are ideal.**

	Features	VIF
0	const	239.560
8	color_G	2.220
6	color_E	2.040
7	color_F	2.020
9	color_H	1.960
10	color_I	1.720
5	table	1.570
2	cut	1.490
11	color_J	1.430
4	depth	1.320
1	carat	1.300
3	clarity	1.210

Step3:

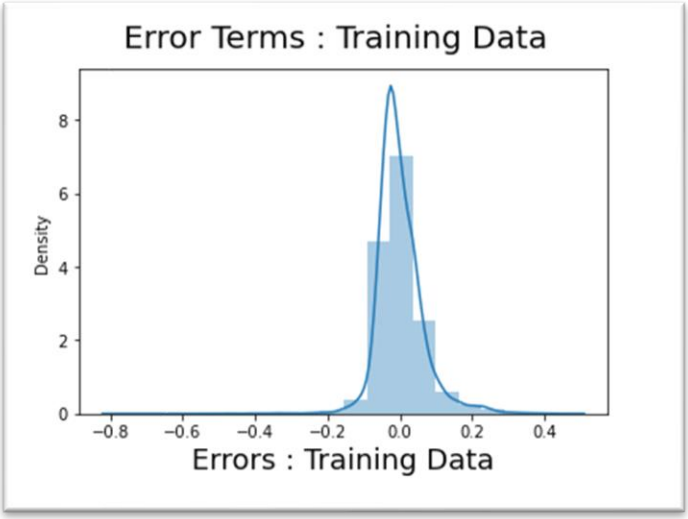
Residual, Prediction & Evaluation of Model:

- Prediction on Training Data (Code: sm.OLS(). fit()).

```
Y_train_pred

3211    0.748
1962    0.087
19698   0.140
2719    0.600
15179   0.095
         ...
12126   0.037
24451   0.017
14157   0.214
23345   0.255
5641    0.034
Length: 18853, dtype: float64
```

- Plot the histogram of the error terms using the function sns.distplot().

Insights:
Error terms seem to be approximately normally distributed, so the assumption on the linear modelling seems to be fulfilled.



Step4: Model Metrics:

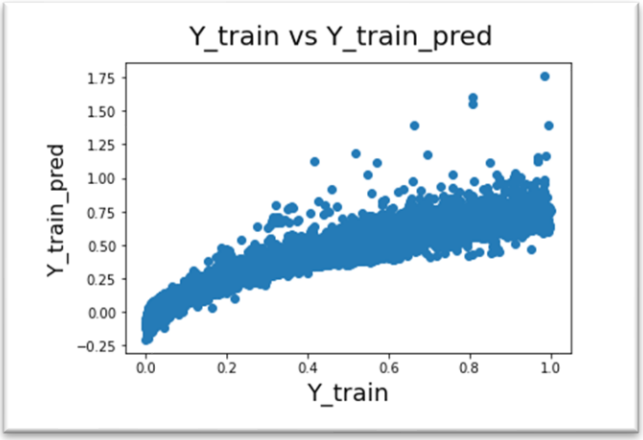
Import r2_score from Sklearn, Calculate the accuracy for the training dataset using the function **r2_score ().**

Accuracy:

```
0.907146163829093
```

Plotting Y and Y_pred to understand the spread using the function **plt.scatter().**

Graph shows scatter plot between actual price vs predicted for test dataset and it is linear and diagonal showing that the forecast accuracy is good.



Calculate MSE on Train Data using the function **mean_squared_error()**

MSE:

```
0.004391201184753229
```

Calculate RMSE on Train Data **(code: np.sqrt(MSE))**

RMSE:

```
0.06626613905120193
```

Calculate MAE on Train Data using the function **mean_absolute_error ().**

MAE:

```
0.045992631095252166
```

Comparing the Accuracy, MSE and RMSE on Full Data and Training Data.

	Full Data	Training Data
Accuracy	0.9077	0.9071
MSE	0.0044	0.0044
RMSE	0.0661	0.0663

Test Data: Data Scaling and Prediction

- Import the MinMaxScaler libraries from Sklearn.
- Apply the data scaling using minmax scaler function for the variables (“carat”, depth”,” table, ’x’, ’y’, ’z’ and price) for Test dataset.
  (code: MinMaxScaler () and scaler.fit_transform())
- Check the first two records for the Test dataset using the function head(2) after applying the scaling.

	carat	cut	clarity	depth	table	x	y	z	price	color_E	color_F	color_G	color_H	color_I	color_J
5478	0.0765	4	3	0.4673	0.4211	0.2019	0.2029	0.3368	0.0366	0	0	1	0	0	0
5347	0.0237	4	6	0.4953	0.4211	0.0743	0.0779	0.2406	0.0307	0	0	0	0	0	0

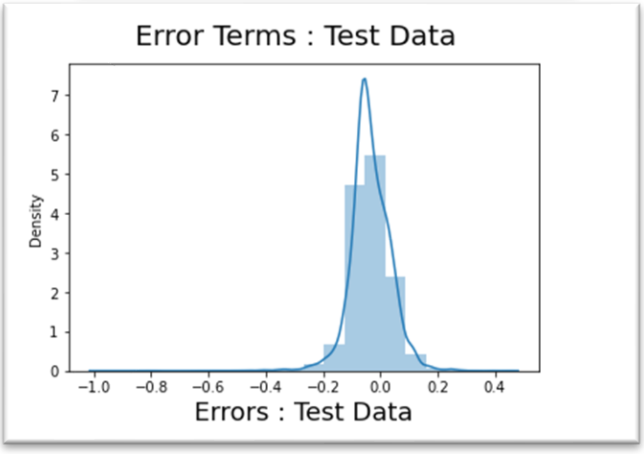
- Split the data into dependent (Y_test) and independent variables (X_test)

Let’s use the model to make the predictions for Test Data using the function lm.predict().

```
5478    -0.0171
5347    -0.0132
16648    0.1067
6630     0.1975
11947    0.0276
          ...
3811     0.0069
26828    0.0871
668      0.6687
5295     0.3267
19282    0.0730
Length: 8080, dtype: float64
```

Plot the histogram of the error terms for Test Data using the function sns.distplot()

```
Text(0.5, 0, 'Errors : Test Data')
```



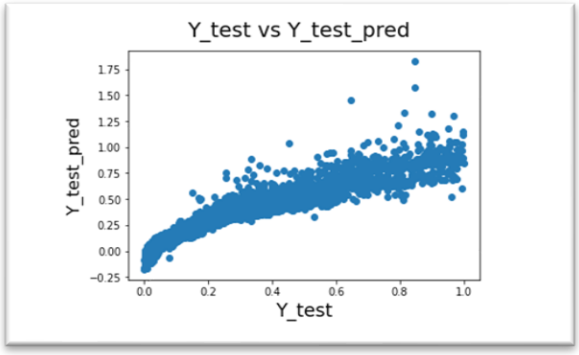
Error terms seem to be approximately normally distributed, so the assumption on the linear modelling seems to be fulfilled.

Calculate the Model Metrics for Test Dataset:

EVALUATION OF THE MODEL:

Plotting Y and Y_pred to understand the spread using the scatter plot **plt.scatter().**

Graph shows scatter plot between actual price vs predicted for test dataset and it is linear and diagonal showing that the forecast accuracy is good.



Calculate MSE on Test Data using the function mean_squared_error()

MSE: `0.006422022419657556`

Calculate RMSE on Test Data using the function np.sqrt(mse)

RMSE: `0.08013752192111731`

Calculate MAE on Test Data using the function mean_absolute_error().

MAE: `0.06123308527211573`

Insights:

- carat, cut, clarity, color, depth and table are the significant variable to predict price variable.
- *R-squared for training data* - **0.907** - **~91%** variance explained and *R-squared for test data* - **0.865** - **~87%** variance explained.
- *p-values* - p-values for all the coefficients seem to be **less than the significance level of 0.05.** - meaning that all the predictors are statistically significant.
- ***Model Metrics: Training Data***
  - MSE: 0.0044
  - RMSE: 0.0663
- ***Model Metrics: Test Data***
  - MSE: 0.0064
  - RMSE: 0.0801

1.3.2 Use the models developed in Part (2) to compare accuracy in training and test sets. Compare the final model of Part (2) and the proposed one in Part (3). Which model provides the most accurate prediction? If the model found in Part (2) is different from the proposed model in Part (3), give an explanation.

Pros and Cons of using full data and Train&Test data:

- Train-test data is appropriate when you have a very large dataset. Full data set is appropriate when the datasets is small
- Train-test split data is used when we need a good estimate of model performance. However, in full data set the model is directly put into production so we cannot predict the performance.
- If train-test split data is done on a file ordered by one of the variables, there are chances the train data would have only certain class of variables which would lead to over fitting.
  For eg: if a file is ordered by employees of certain age, or level of income.
- Train & Test dataset are easy to implement and interpret. Less time consuming in execution but if the dataset is small, keeping a portion for testing would be decrease the accuracy of the predictive model. If the split is not random, the output of the evaluation matrices is inaccurate.

Comparing the Full Data, Training Data and Test Data:

The performance of predictions on testing and training as below

	Full Data	Training Data	Test Data
Accuracy	0.9077	0.9071	0.8652
MSE	0.0044	0.0044	0.0064
RMSE	0.0661	0.0663	0.0801

Business Interpretations & Recommendations:

- Based on the above table we can conclude that Full Data set have higher accuracy and lower RMSE compared to Train and Test Dataset hence a better approach in this case, as compared to train/test data split approach
- Depth and Table have negative effect on the price.
- Carat has highest beta coefficient value, making it biggest multiplier to the price of zirconia diamond.
- All types of cuts have a positive beta coefficient with Cut Ideal with the highest beta value and indicating the price to be the highest if the cut is “Ideal”.
- All the colors have a negative effect on price. Which indicates that any presence of color will reduce the price.

The ideal, premium, very good cut types are the one which are bringing profits so that we could use marketing for these to bring in more profits.

- The clarity of the diamond is the next important attributes the more the clear is the stone the profits are more.
- So, the company has to take the carat, clarity, color and cut to consideration while fixing the price of the Zirconia gem.
- Carat plays the major role in the prediction of the price and hence the company need to rely on the carat variable for deciding the price on the zirconia gems.

Problem2:

You are hired by a sports analysis agency to understand the selection process of high school football players into college with a full or partial scholarship.

You are provided details of **6215 high school graduates** who have been inducted into 4-year degree colleges with **either full or partial scholarships**. You have to help the agency in **predicting whether a high school graduate will win a full scholarship on the basis of the information given in the data set**.

Also, find out the **important factors** which are instrumental in winning a full scholarship in colleges.
The data dictionary is given below:

#	Variables	Description
1	**Scholarship**	Won a college scholarship: Full / Partial
2	**Academic Score**	High school academic performance of a candidate
3	**Score on Plays Made**	A composite score based on the achievements on the field
4	**Missed Play Score**	A composite score based on the failures on the field
5	**Injury Propensity**	This has 3 ordinal levels: High, Moderate, Normal and Low. It has been calculated based on what proportion of time a candidate had an injury problem
6	**School Type**	3 types of schools based on their location
7	**School Score**	A composite score based on the overall achievement of the candidates' school, based on the school's academic, sports and community service performance
8	**Overall Score**	A composite score based on a candidate's family financial state, school performance, psychosocial attitude etc
9	**Region**	Region of the country where the school is located

Remarks: All the questions of problem2 which explained here, are also performed in python as well. Please refer python notebook "PM_GA_Problem2_Logistic Regression and LDA".

Import the necessary libraries and load the dataset:

Steps and Results:

- Import libraries:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

import warnings
warnings.filterwarnings("ignore")
```

❖ Load the data set 'Football+Scholarship.csv' and display first five rows using .head function:

	Academic_Score	Score_on_Plays_Made	Missed_Play_Score	Injury_Propensity	School_Type	School_Score	Overall_Score	Region	Scholarship
0	7.000	0.270	0.360	High	D	0.450	8.800	Eastern	Partial
1	6.300	0.300	0.340	Low	C	0.490	9.500	Eastern	Partial
2	8.100	0.280	0.400	Moderate	C	0.440	10.100	Eastern	Partial
3	7.200	0.230	0.320	Moderate	C	0.400	9.900	Eastern	Partial
4	7.200	0.230	0.320	Moderate	C	0.400	9.900	Eastern	Partial

❖ We made a copy of the original df using .copy function and checked first 2 records using .head function

	Academic_Score	Score_on_Plays_Made	Missed_Play_Score	Injury_Propensity	School_Type	School_Score	Overall_Score	Region	Scholarship
0	7.000	0.270	0.360	High	D	0.450	8.800	Eastern	Partial
1	6.300	0.300	0.340	Low	C	0.490	9.500	Eastern	Partial

2.1 EDA (Exploratory Data Analysis)

Objective of EDA:

The very first step of any data analysis assignment is to do the exploratory data analysis (EDA). Once you have understood the nature of all the variables, especially identified the response and the predictors, apply appropriate methods to determine whether

- ❖ There is any **duplicate observation or missing data and whether the variables have a symmetric or skewed distribution.**
- ❖ The data may contain **various types of attributes and so numerical and/or visual data summarization techniques** need to be appropriately decided.
- ❖ Both **univariate and bivariate analyses and pre-processing of data** are important.
- ❖ Check for **outliers and comment on removing or keeping them while model building**.
- ❖ This is a classification problem, **the dependence of the response on the predictors needs to be investigated.**

A: We checked the number of Rows and Columns in the dataset using .shape function:

Result:

```
The number of columns (variables) in the dataset is 9
The number of rows (observations per variable) in the dataset is 6215
```

B: Then we checked the data type of variables in the data set using .info function

Result:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6215 entries, 0 to 6214
Data columns (total 9 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   Academic_Score       6215 non-null   float64
 1   Score_on_Plays_Made  6215 non-null   float64
 2   Missed_Play_Score    6215 non-null   float64
 3   Injury_Propensity    6215 non-null   object
 4   School_Type          6215 non-null   object
 5   School_Score         6215 non-null   float64
 6   Overall_Score        6215 non-null   float64
 7   Region               6215 non-null   object
 8   Scholarship          6215 non-null   object
dtypes: float64(5), object(4)
memory usage: 437.1+ KB
```

Insights:
This shows that there are 5 numerical and 4 object variables.

C: We checked the basic measures of descriptive statistics using .describe (include='all').T function

Result:

	count	unique	top	freq	mean	std	min	25%	50%	75%	max
Academic_Score	6,215.0	NaN	NaN	NaN	7.219	1.292	3.8	6.4	7.0	7.7	15.9
Score_on_Plays_Made	6,215.0	NaN	NaN	NaN	0.337	0.16	0.08	0.23	0.29	0.4	1.33
Missed_Play_Score	6,215.0	NaN	NaN	NaN	0.32	0.145	0.0	0.25	0.31	0.39	1.66
Injury_Propensity	6215	4	Low	2650	NaN	NaN	NaN	NaN	NaN	NaN	NaN
School_Type	6215	3	C	3384	NaN	NaN	NaN	NaN	NaN	NaN	NaN
School_Score	6,215.0	NaN	NaN	NaN	0.531	0.147	0.22	0.43	0.51	0.6	1.98
Overall_Score	6,215.0	NaN	NaN	NaN	10.457	1.173	8.0	9.5	10.2	11.3	14.9
Region	6215	3	Eastern	2835	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Scholarship	6215	2	Partial	4028	NaN	NaN	NaN	NaN	NaN	NaN	NaN

And mode using .mode function:

Result:

	Academic_Score	Score_on_Plays_Made	Missed_Play_Score	Injury_Propensity	School_Type	School_Score	Overall_Score	Region	Scholarship
0	6.800	0.280	0.300	Low	C	0.500	9.500	Eastern	Partial

Insights:

1. Looking at the 5 values (min, 25%, 50%, 75% and max), it appears that there are outliers in the numerical variables.
2. The gap between mean and median for all the five numerical variables are minimal**.**
3. Injury Propensity has *4* unique values followed by School Type with *3,* Region with *3* and Scholarship with *2* unique values respectively.

D: Checked for duplicate records in the dataset using .duplicated() function:

Result:

```
There are 947 records in the dataset which are duplicates
```

Insights: As the dataset relates to student records and we don't have unique identifier (like student id), these 947 seems to be potential duplicate and hence, NOT dropped from the dataset.

E: Check missing values in the dataset using isnull().sum() function

Result:

```
Academic_Score          0
Score_on_Plays_Made     0
Missed_Play_Score       0
Injury_Propensity       0
School_Type             0
School_Score            0
Overall_Score           0
Region                  0
Scholarship             0
dtype: int64
```

Insights:
There is NO missing value in the dataset.

F: Check skewness of the dataset using .skew function

Result:

```
Academic_Score          1.751
Score_on_Plays_Made     1.406
Missed_Play_Score       0.492
School_Score            1.733
Overall_Score           0.601
dtype: float64
```

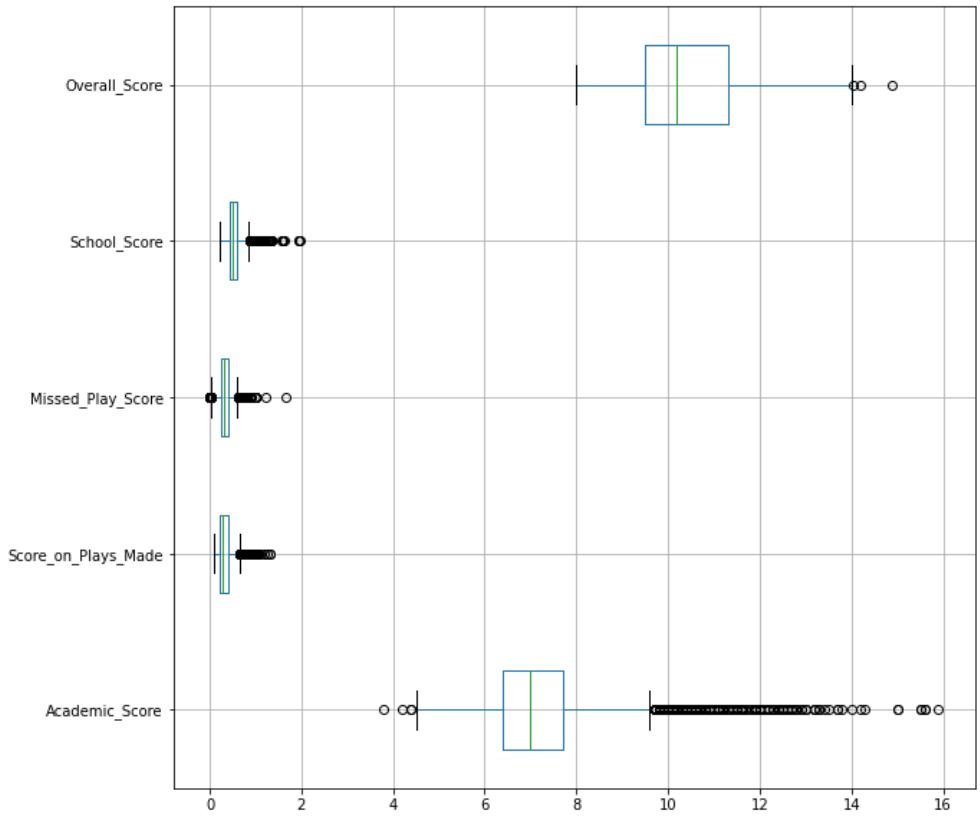
Insights:

As a general rule, if the skewness value is less than −1 or greater than +1, then the distribution is considered as highly skewed. If the skewness value is between −1 and −½ or between +½ and +1, then the distribution is considered as moderately skewed**.**

In our dataset, it appears that:

1. Academic Score, Score On Plays Made and School Score are highly skewed.
2. Overall Score appears to be Moderately Skewed and
3. Missed Play Score appears to be a borderline case of Moderately skewed distribution.

G: Above insights / skewness is easily visible using boxplot, as below:



Insights: The boxplots also show that all above are right-skewed.

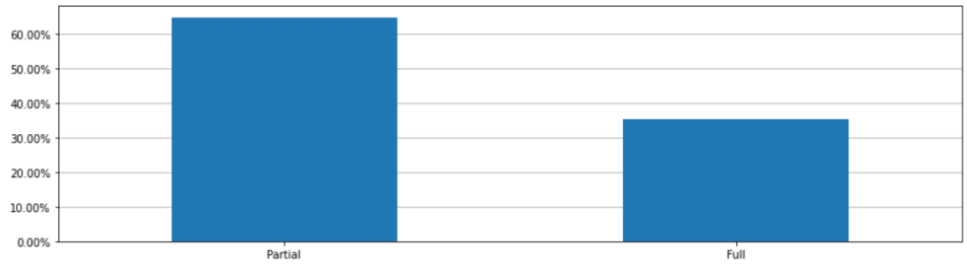
H: Target Variable Analysis – We then looked at the distribution of the Target variable using value_counts function.

Count of unique values in the Target variable ("Scholarship") along with the ratio.

```
Partial    4028
Full       2187
Name: Scholarship, dtype: int64
```

```
Partial   0.648
Full      0.352
Name: Scholarship, dtype: float64
```

We also looked at the visual presentation using bar chart.

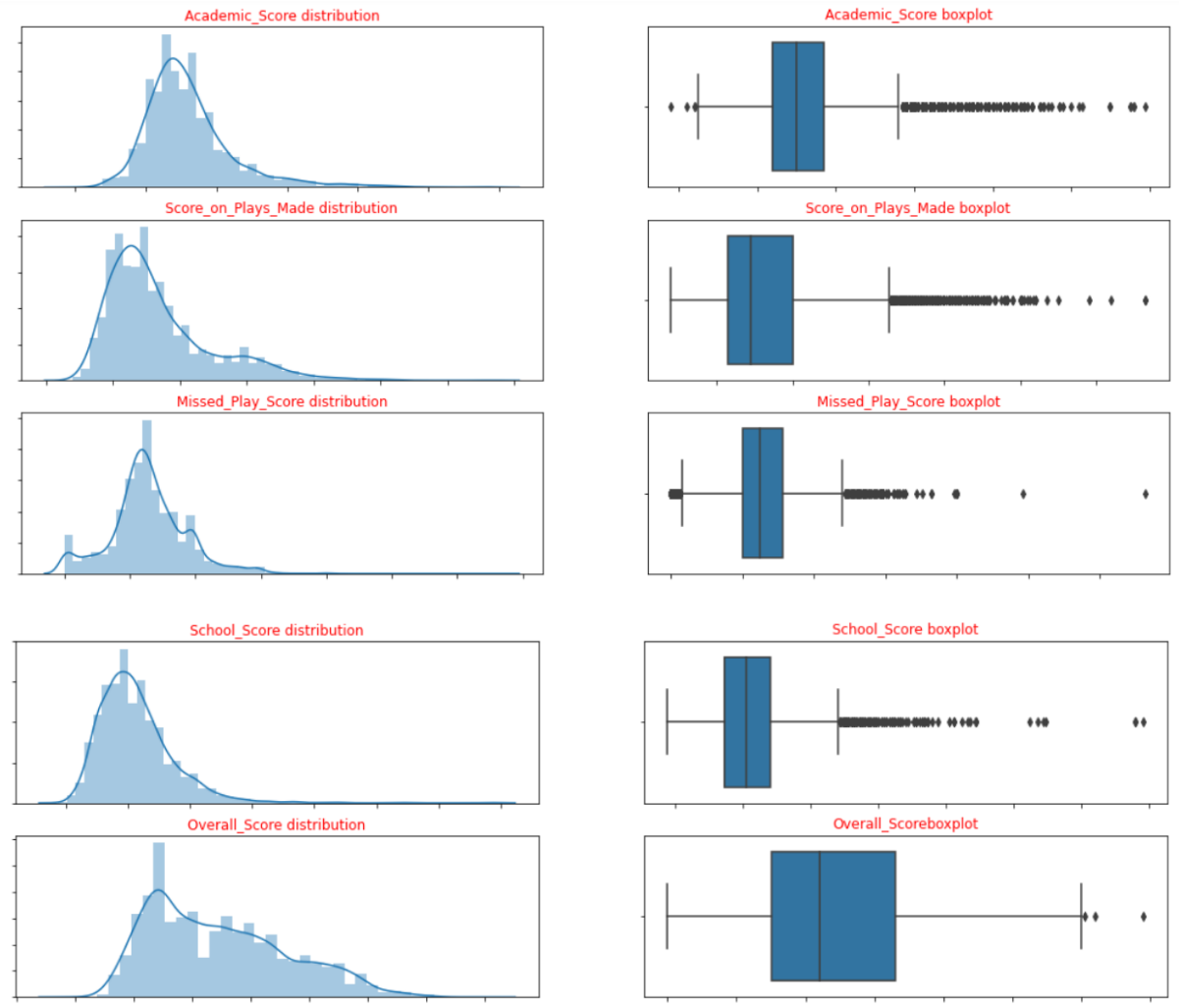


Insights:

- Out of total. there are **4028(~64%) candidates** who have won **Partial Scholarship** and 2187(~35%) candidates have won full scholarship. This clearly indicates that there is a class imbalance.

I: Univariate Analysis (for five Numerical Variables)

We using distplot, boxplot and subplot functions for all 5 numerical variables.



Insights:

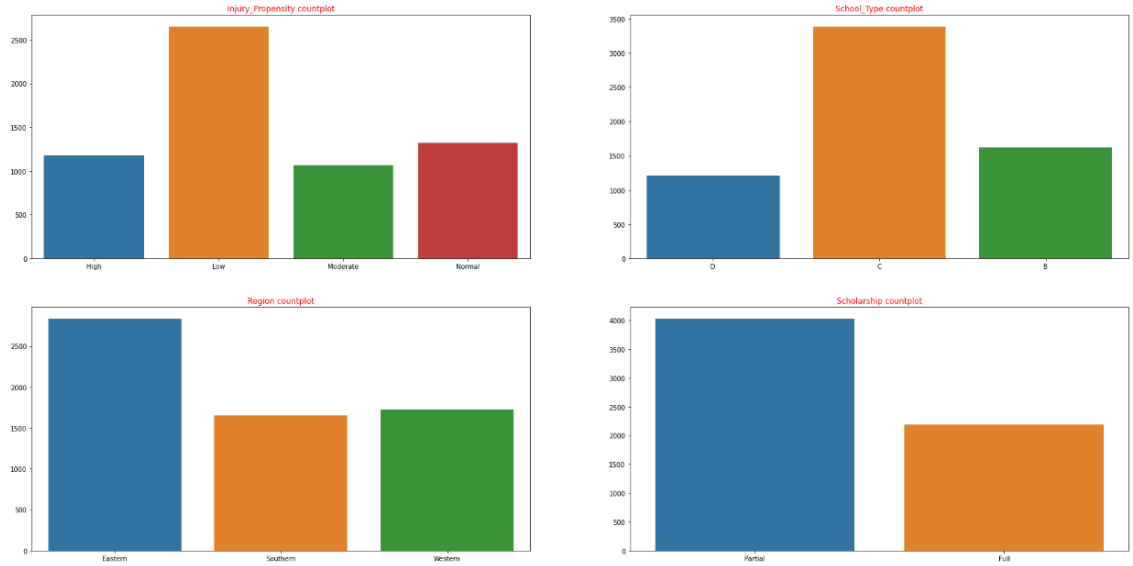
- For **all the variables, data is not normally distributed**
- All **five variables** have **outliers** in the dataset. The outliers are not treated at this point of time.

J: Univariate Analysis (Categorical Variables)

Analysing the Unique counts of all the 'Object' type variables

```
Injury_Propensity
Low         2650
Normal      1319
High        1181
Moderate    1065
Name: Injury_Propensity, dtype: int64


School_Type
C    3384
B    1620
D    1211
Name: School_Type, dtype: int64


Region
Eastern     2835
Western     1724
Southern    1656
Name: Region, dtype: int64


Scholarship
Partial    4028
Full       2187
Name: Scholarship, dtype: int64
```

K: The for four categorical variables is visualised as bar chart – using the count plot function



Insights:

In Univariate analysis (categorial variables), we have used **count plot** and found:

- **Injury Propensity:** Highest number of students had **Low** injury propensity followed by *normal, high and moderate*.
- **School Type: C** has the highest count followed by *B and D*.
- **Region: Eastern** has higher contribution followed by *Western and Southern*.
- **Scholarship:** Majority of the students are with **Partial** scholarship and remaining with *full* scholarship.

L: Bivariate Analysis (Numerical Variables vs Numerical Variables)

Check Measures of Dispersion

Co-variance

	Academic_Score	Score_on_Plays_Made	Missed_Play_Score	School_Score	Overall_Score
Academic_Score	1.670	0.047	0.061	0.058	-0.117
Score_on_Plays_Made	0.047	0.026	-0.009	0.006	-0.008
Missed_Play_Score	0.061	-0.009	0.021	0.001	-0.002
School_Score	0.058	0.006	0.001	0.022	0.001
Overall_Score	-0.117	-0.008	-0.002	0.001	1.375

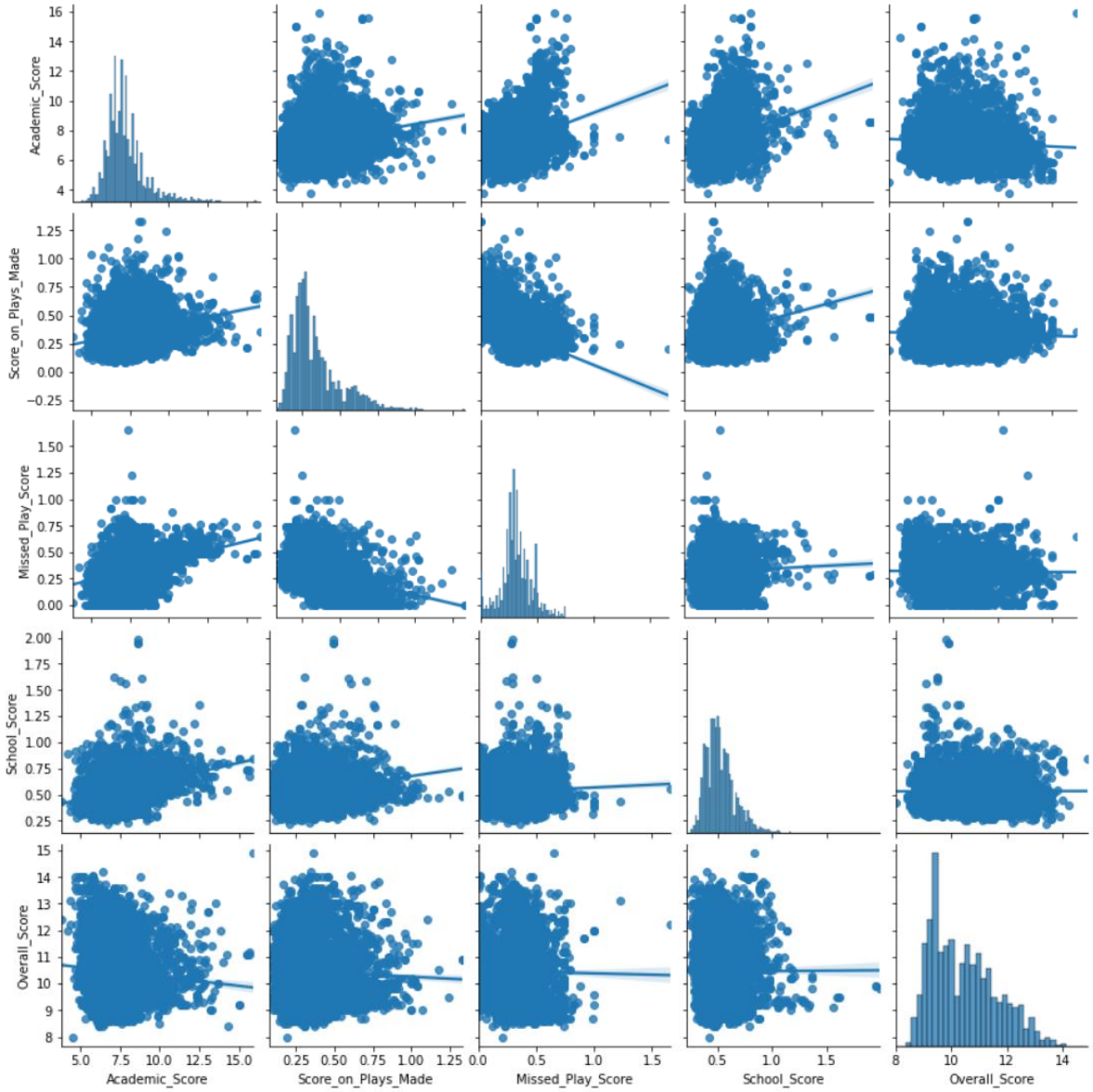
Insights:

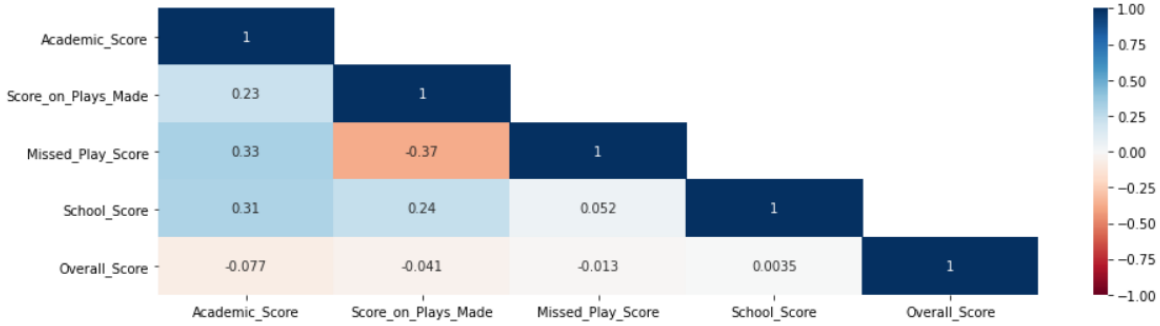
- Most of the variables are positively correlated. However, none of the variables have high variance value.
- This strength of the relationship will be established in the correlation analysis of variables below.

Correlation

	Academic_Score	Score_on_Plays_Made	Missed_Play_Score	School_Score	Overall_Score
Academic_Score	1.000000	0.225572	0.325317	0.307041	-0.077024
Score_on_Plays_Made	0.225572	1.000000	-0.367801	0.237480	-0.040800
Missed_Play_Score	0.325317	-0.367801	1.000000	0.051730	-0.013180
School_Score	0.307041	0.237480	0.051730	1.000000	0.003476
Overall_Score	-0.077024	-0.040800	-0.013180	0.003476	1.000000

Visualisation of correlation matrix using Scatter plot & Heatmap





Ranked the variables based on the strength of correlation – HIGH to LOW

```
Missed_Play_Score    Score_on_Plays_Made    0.37
Score_on_Plays_Made  Missed_Play_Score      0.37
Missed_Play_Score    Academic_Score         0.33
Academic_Score       Missed_Play_Score      0.33
                     School_Score           0.31
School_Score         Academic_Score         0.31
                     Score_on_Plays_Made    0.24
Score_on_Plays_Made  School_Score           0.24
                     Academic_Score         0.23
Academic_Score       Score_on_Plays_Made    0.23
                     Overall_Score          0.08
Overall_Score        Academic_Score         0.08
Missed_Play_Score    School_Score           0.05
School_Score         Missed_Play_Score      0.05
Score_on_Plays_Made  Overall_Score          0.04
dtype: float64
```

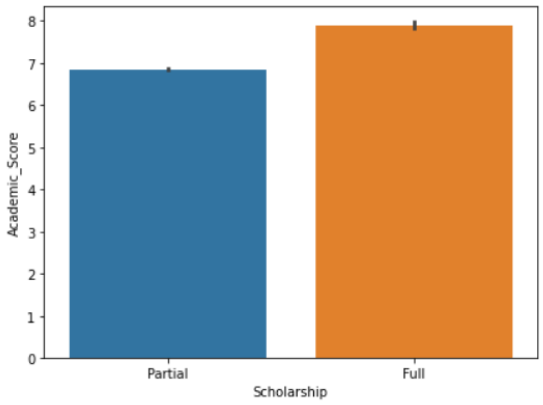
Insights:

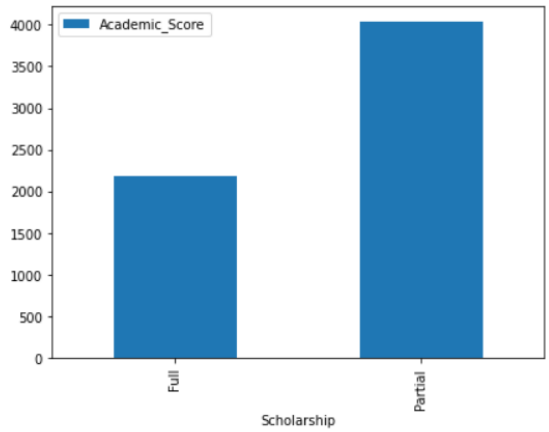
In Bivariate analysis (Numerical variables), we have used **scatter plot & heatmap** and found:

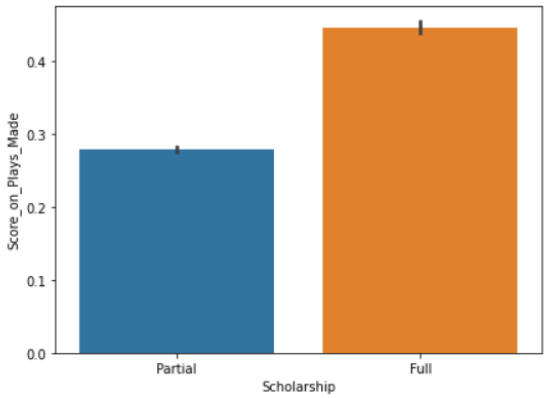
- In scatter plot and heat map both shows that **none of the variables shows the strong relationship**.

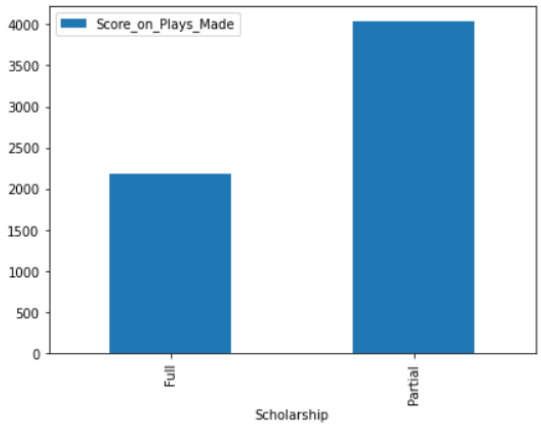
M: Bivariate Analysis (Numerical Variables vs Target Variables) using barplot and pairplot

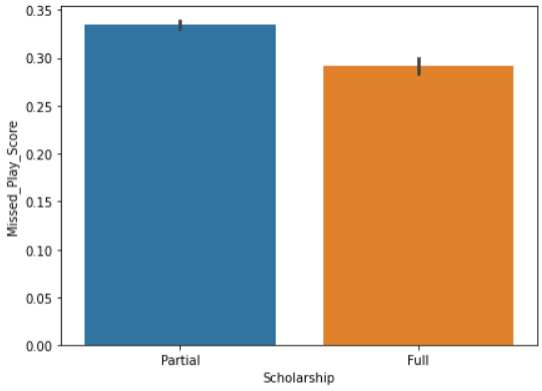
Av Academic Score	No. of Students

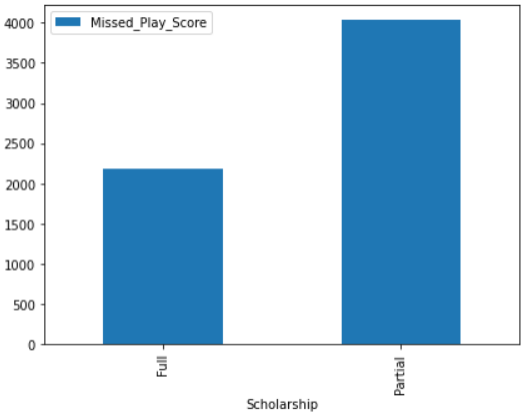


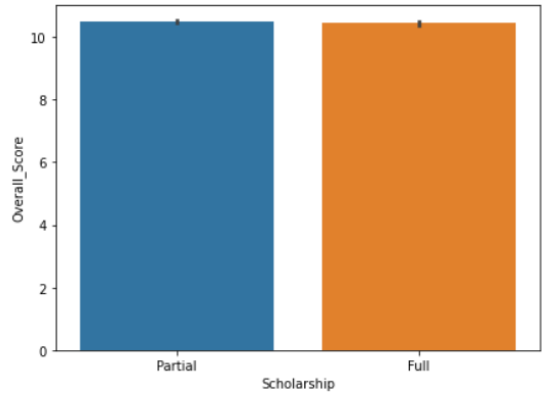


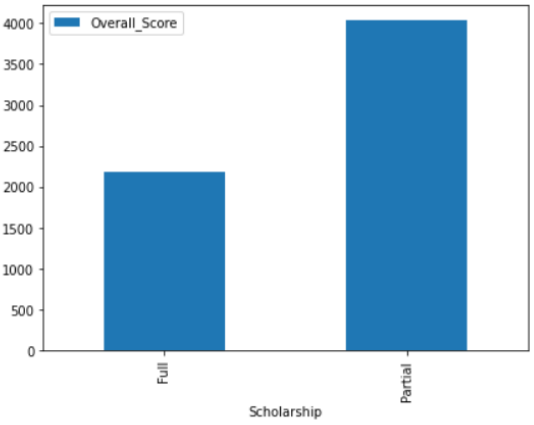


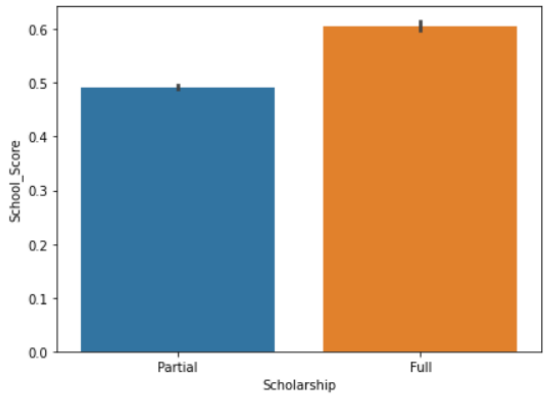


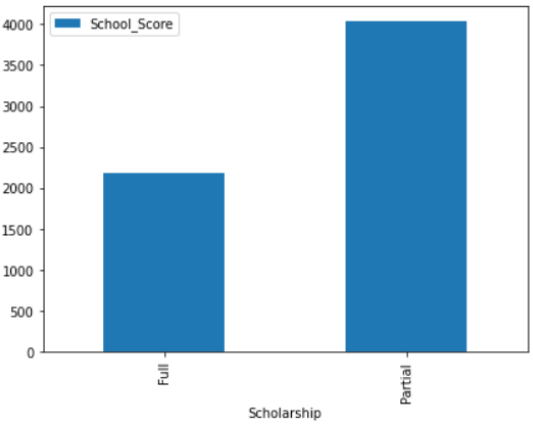


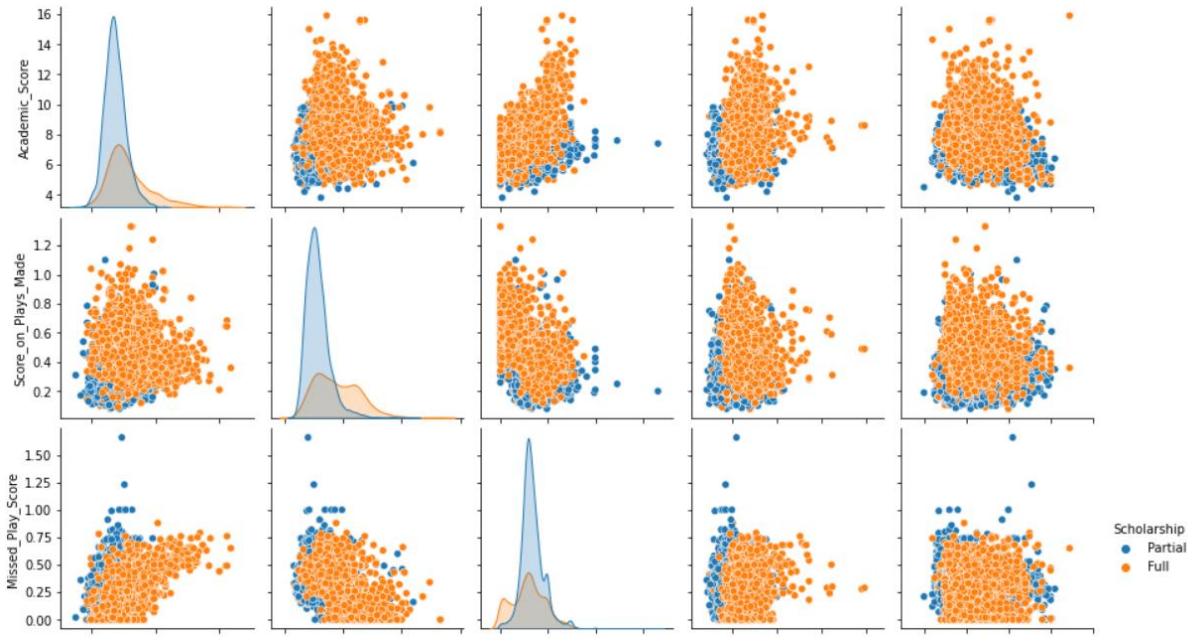


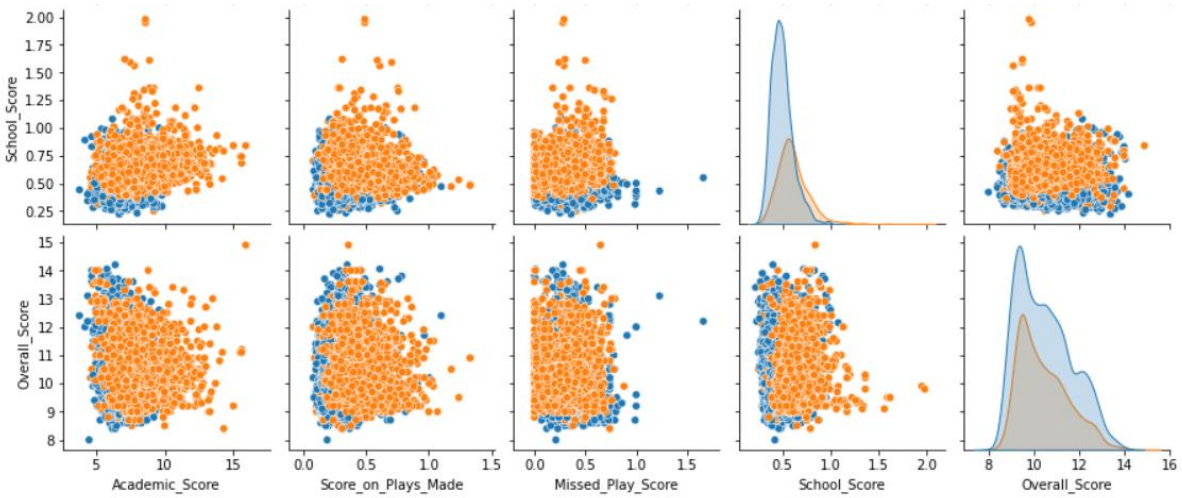

Overall_Score
Partial
Full
Scholarship

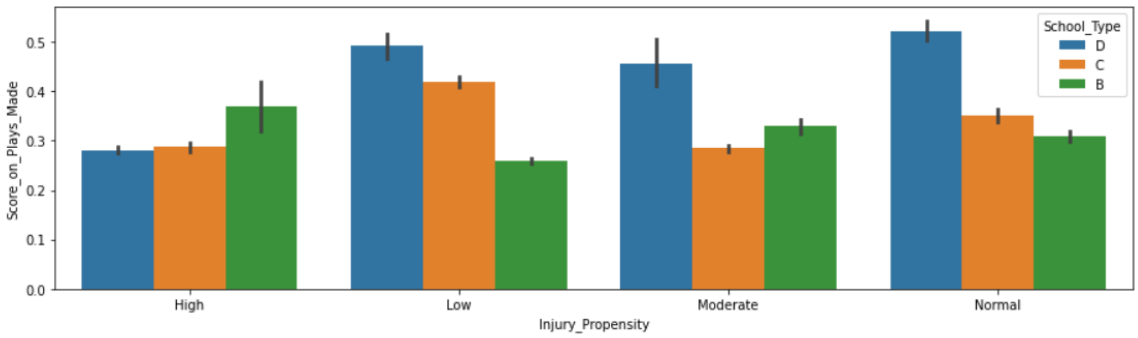

Overall_Score
Full
Partial
Scholarship


School_Score
Partial
Full
Scholarship


School_Score
Full
Partial
Scholarship


Academic_Score
Score_on_Plays_Made
Missed_Play_Score
Scholarship
Partial
Full

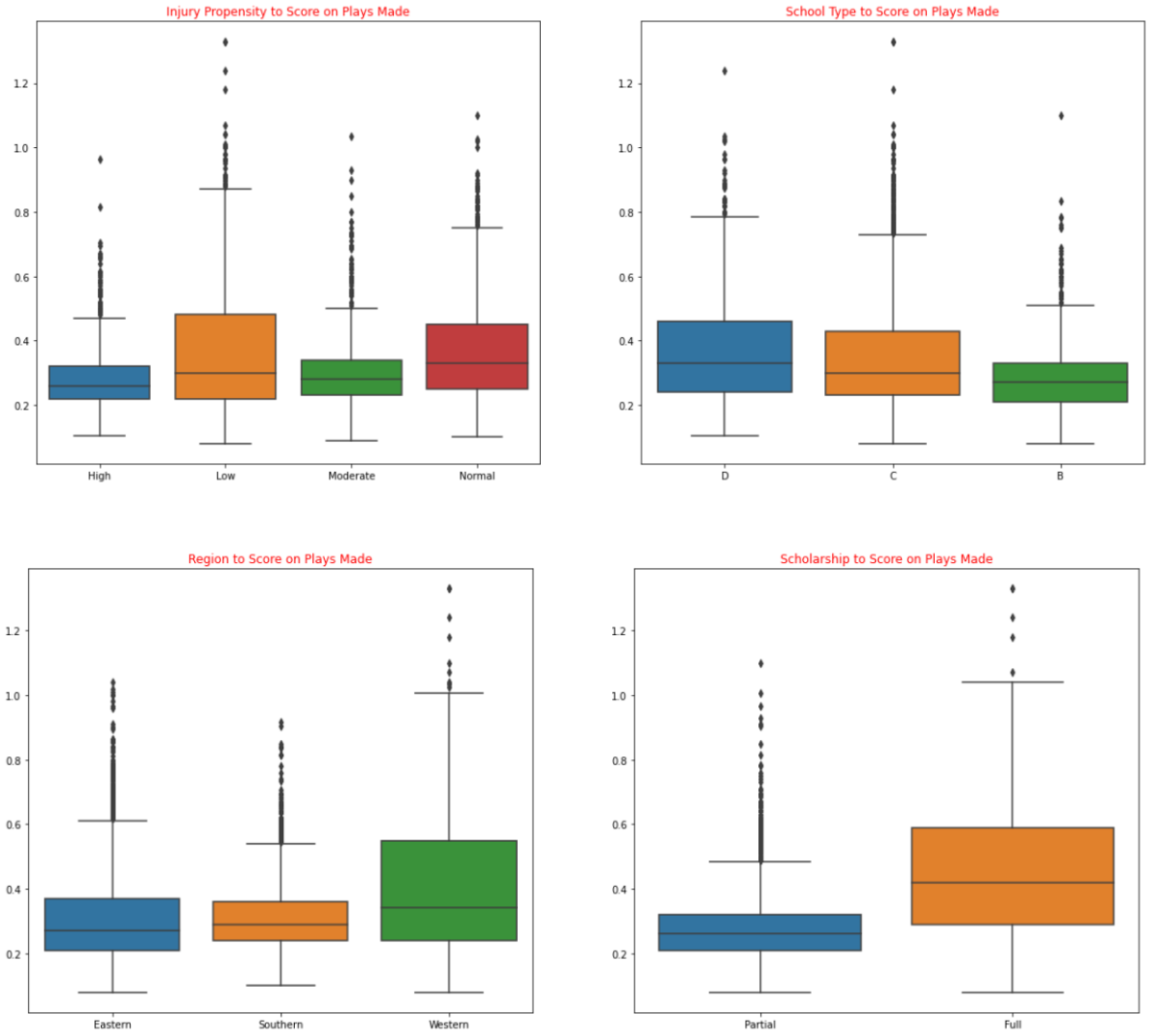




Insights:

- Students scoring high in Academic Score, Score on Plays made, School Score, Overall Score are the winners of Full Scholarship.
- Missed in Play made has an effect to missing the Full scholarship.
- The bar plot (count) explains students scoring high is less in number and so the count of partial scholarship is higher.
- Our analysis also shows that Injury Propensity has a relation with Score on Plays Made. The Scores go HIGH when the Injury Propensity is LOW and NORMAL. However, the scores move LOW when the Injury Propensity is HIGH.
- We can also see that even with high and moderate injury propensity, the average score on plays made by students of school B is highest among all 3 schools.

N: Bivariate Analysis (Categorical variables vs Scores on Play Made) using boxplot and subplot:

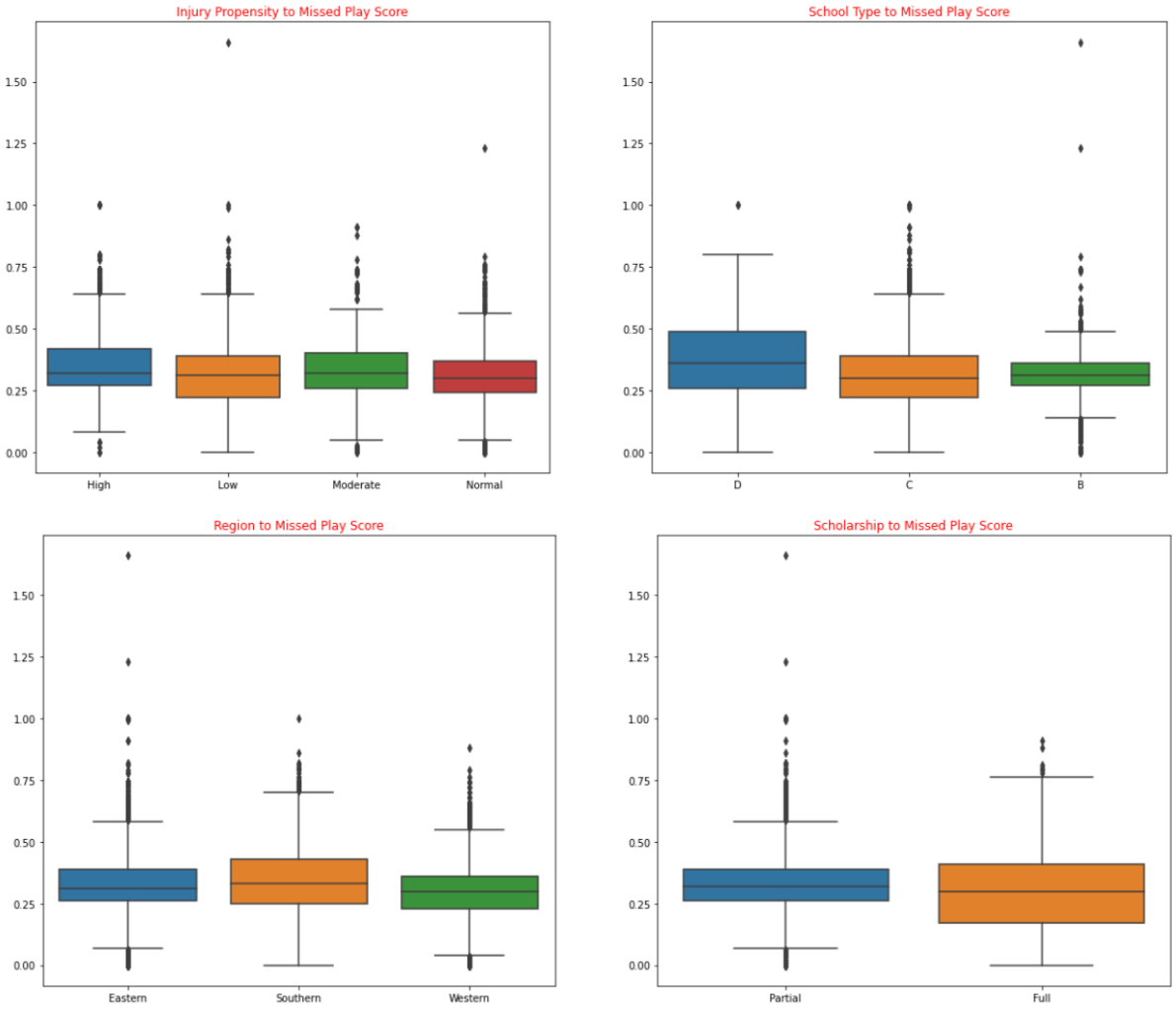


Insights:

In Bivariate analysis (Categorical vs Numerical variables), we have compared **all categorical variables with Score on Plays Made**. Find below the detailed analysis:

- **Median of Score on Play Made across all the categories of Injury - Different propensity** leads to different average scores. Further, *it is noted that **LOW** and **Normal** injury propensity has the highest median Score and IQR on Play Made.*
- **Median of Score on Play Made across all the school types are different.** School D has highest median and IQR of scores on play made whereas school C has the highest Score on Play Made compared to the other school.
- **Median of Score on Play Made is different for all the Regions**. The Score on Play Made, median score and IQR are found to be the highest for Western Region which is followed by the Eastern and Southern Region.
- **Median of Score on Play Made is different for both Full and Partial Scholarship won students**. Score on Play Made, median score and IQR is highest for students who have won **Full Scholarship**.

O: Bivariate Analysis (Categorical variables vs Missed Play Score) using boxplot and subplot

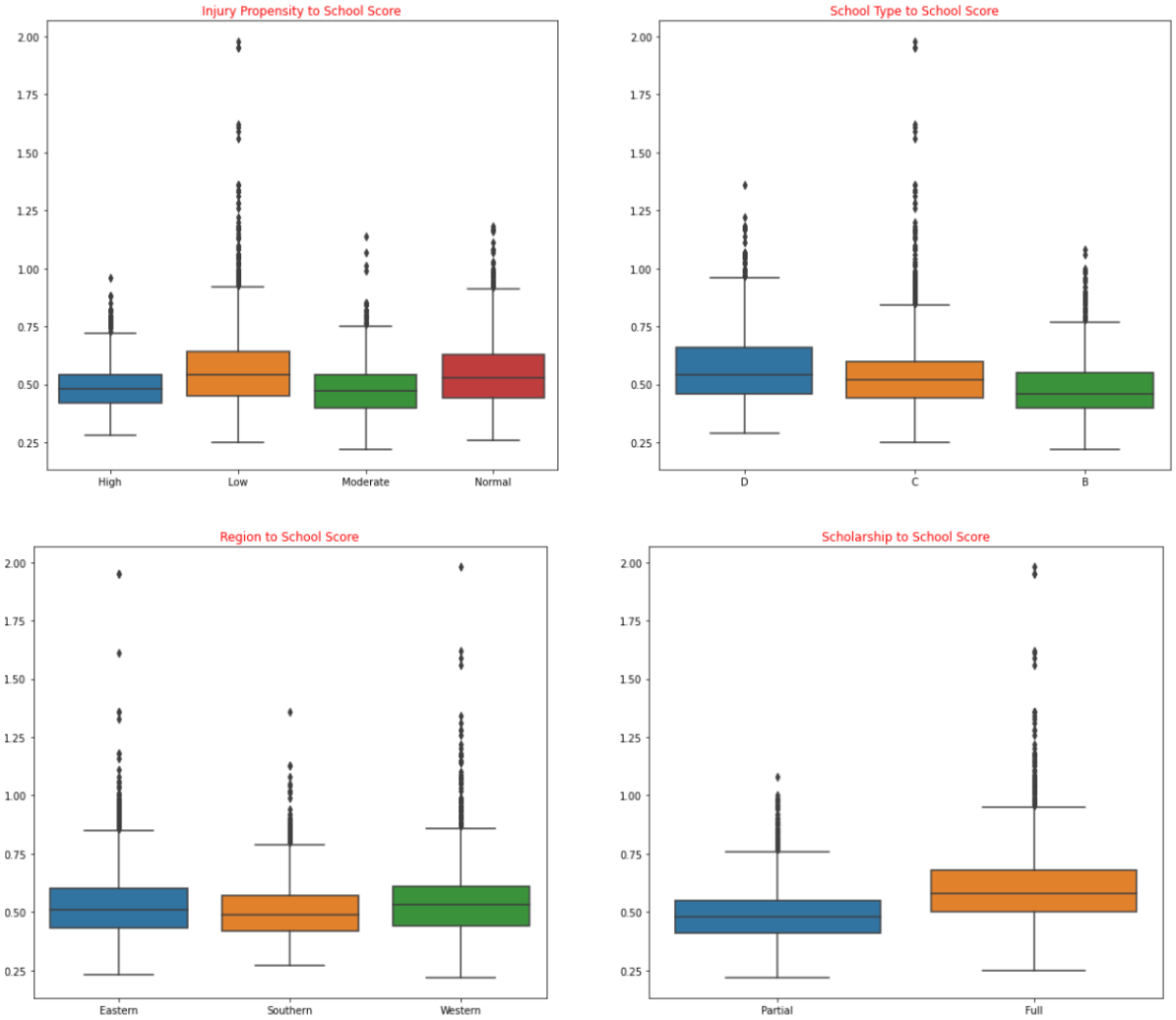


Insights:

In Bivariate analysis (Categorical vs Numerical variables), we have compared **all categorical variables with Missed Play Score**. Find below the detailed analysis:

- **Median of Missed Play Score across all the categories of Injury Propensity** is the **same**. Further, *it is also noted that **LOW** category has similar scores.*
- **Median of Missed Play Score is similar for school type C and B but different for school type D. School type 'B'** has the highest Missed Play Score compared to the other school types.
- **Median of Missed Play Score has minimal difference across all the Regions**. The Missed Play Score is found to be the highest for Eastern Region which is followed by the Southern and Western Region.
- **Median of Missed Play Score is different for both Full and Partial Scholarship won students** missed Play Score is highest for students who have won **Partial Scholarship**.

P: Bivariate Analysis (Categorical variables vs School Score)

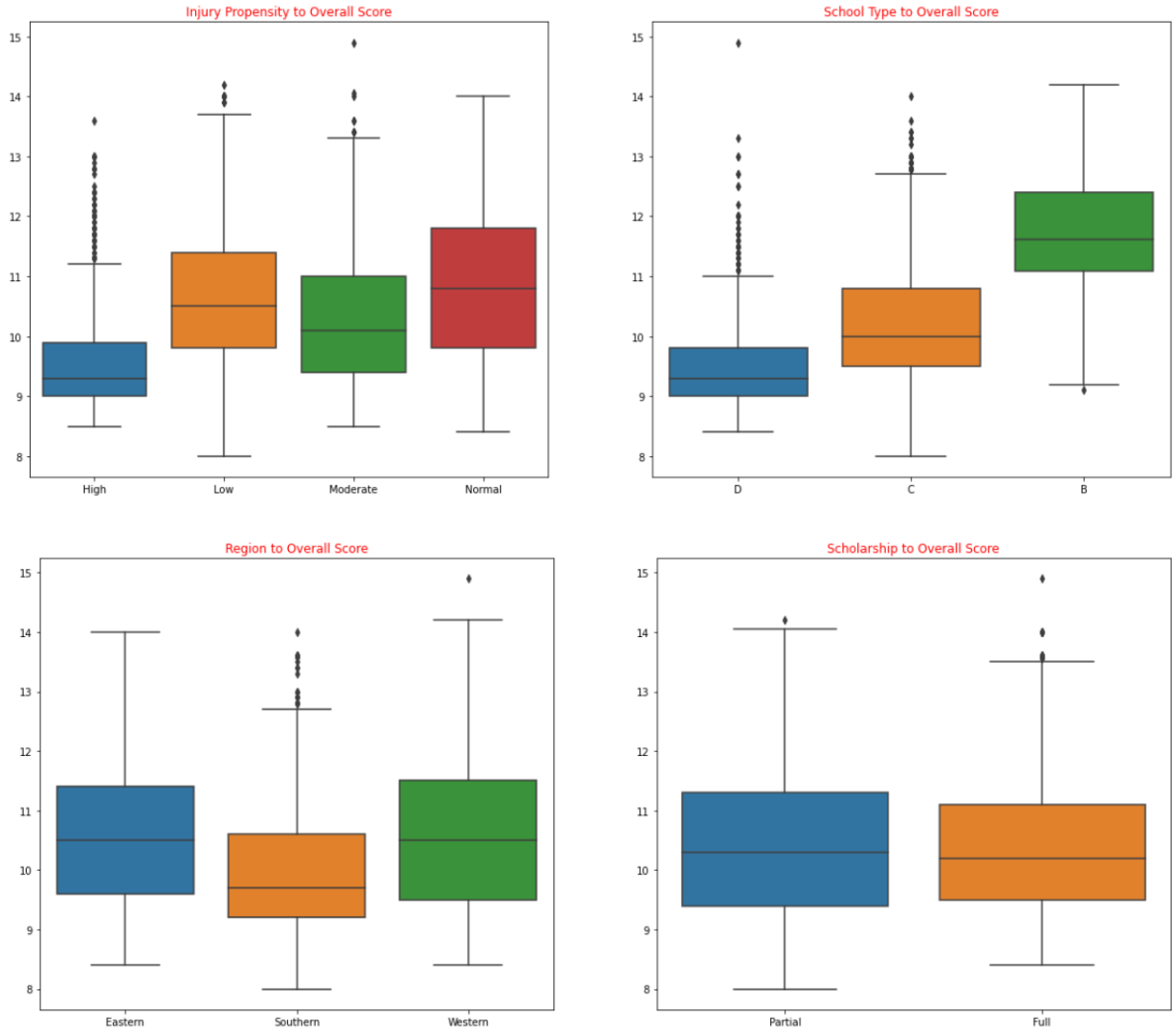


Insights:

In Bivariate analysis (Categorical vs Numerical variables), we have compared **all categorical variables with School Score**. Find below the detailed analysis:

- **Median of School Score across all the categories of Injury Propensity** is **different**. Further, *it is also noted that **LOW** category has the highest School Scores.*
- **Median of School Score is different across all school types. School type 'C'** has the highest School Score compared to the other school types.
- **Median of School Score has minimal difference** across all the Regions. The School Score is found to be the highest for Western Region which is followed by the Eastern and Southern Region.
- **Median of School Score is different for both Full and Partial Scholarship won students.** School Score is highest for students who have won **Full Scholarship**.

Q: Bivariate Analysis (Categorical variables vs Overall Score)



Insights:

In Bivariate analysis (Categorical vs Numerical variables), we have compared **all categorical variables with Overall Score**. Find below the detailed analysis:

- **Median of Overall Score across all the categories of Injury Propensity** is **different**. Further, *it is also noted that **Moderate** category has the highest Overall Scores.*
- **Median of Overall Score is different across all school types. School type 'D'** has the highest Overall Score compared to the other school types.
- **Median of Overall Score is different across all the Regions**. The Overall Score is found to be the highest for Western Region.
- **Median of Overall Score is different for both Full and Partial Scholarship won students**. Overall Score is highest for students who have won **Full Scholarship**.

2.2 Build various iterations of the Logistic Regression model using appropriate variable selection techniques for the full data. Compare values of model selection criteria for proposed models. Compare as many criteria as you feel are suitable

Objective:

- Use **Full Data** to develop a **logistic regression model** to **identify significant predictors**.
- Check whether the proposed model is **free of multicollinearity**. Apply variable selection method as required. Show **all intermediate models leading to the final model**.
- Justify your choice of the final model. Which are **the significant predictors?**
- For easier interpretation of the models, later on, it may be better to code **Full = 1 and Partial = 0**. You may assume the opposite, but then you have to be very careful about the interpretation of the logistic model coefficients later.

Step 1: Treat the object variables appropriately by either creating dummy variables (One-Hot Encoding) or coding it up in an ordinal manner.

	Academic_Score	Score_on_Plays_Made	Missed_Play_Score	Injury_Propensity	School_Type	School_Score	Overall_Score	Region	Scholarship
0	7.0	0.27	0.36	High	D	0.45	8.8	Eastern	Partial
1	6.3	0.30	0.34	Low	C	0.49	9.5	Eastern	Partial

- Analysing the data, we code 'Injury Propensity' variable in an ordinal manner.
- We also code the "Scholarship" variable based on the class of interest. Hence, consider full scholarship as "1" and partial scholarship as "0"

Step 2: Understand the datatype of variables and convert to categorical/numeric wherever necessary.

Datatype of variables BEFORE conversion:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6215 entries, 0 to 6214
Data columns (total 9 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   Academic_Score       6215 non-null   float64
 1   Score_on_Plays_Made  6215 non-null   float64
 2   Missed_Play_Score    6215 non-null   float64
 3   Injury_Propensity    6215 non-null   object
 4   School_Type          6215 non-null   object
 5   School_Score         6215 non-null   float64
 6   Overall_Score        6215 non-null   float64
 7   Region               6215 non-null   object
 8   Scholarship          6215 non-null   object
dtypes: float64(5), object(4)
memory usage: 437.1+ KB
```

Datatype of variables AFTER conversion:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6215 entries, 0 to 6214
Data columns (total 9 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   Academic_Score       6215 non-null   float64
 1   Score_on_Plays_Made  6215 non-null   float64
 2   Missed_Play_Score    6215 non-null   float64
 3   Injury_Propensity    6215 non-null   int64
 4   School_Type          6215 non-null   object
 5   School_Score         6215 non-null   float64
 6   Overall_Score        6215 non-null   float64
 7   Region               6215 non-null   object
 8   Scholarship          6215 non-null   int64
dtypes: float64(5), int64(2), object(2)
memory usage: 437.1+ KB
```

After conversion, we look at first 2 records:

```
df.head(2)
```

	Academic_Score	Score_on_Plays_Made	Missed_Play_Score	Injury_Propensity	School_Type	School_Score	Overall_Score	Region	Scholarship
0	7.000	0.270	0.360	4	D	0.450	8.800	Eastern	0
1	6.300	0.300	0.340	1	C	0.490	9.500	Eastern	0

Step 3: Convert categorical/object variable to dummy variables for building the model.

```
# Converting the other 'object' type variables as dummy variables

df_dummy = pd.get_dummies(df,drop_first=True)
df_dummy.head(2)
```

	Academic_Score	Score_on_Plays_Made	Missed_Play_Score	Injury_Propensity	School_Score	Overall_Score	Scholarship	School_Type_C	School_Type_D	F
0	7.000	0.270	0.360	4	0.450	8.800	0	0	1	
1	6.300	0.300	0.340	1	0.490	9.500	0	1	0	

We look at the df.info again after adding dummy variables

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6215 entries, 0 to 6214
Data columns (total 11 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   Academic_Score       6215 non-null   float64
 1   Score_on_Plays_Made  6215 non-null   float64
 2   Missed_Play_Score    6215 non-null   float64
 3   Injury_Propensity    6215 non-null   int64
 4   School_Score         6215 non-null   float64
 5   Overall_Score        6215 non-null   float64
 6   Scholarship          6215 non-null   int64
 7   School_Type_C        6215 non-null   uint8
 8   School_Type_D        6215 non-null   uint8
 9   Region_Southern      6215 non-null   uint8
 10  Region_Western       6215 non-null   uint8
dtypes: float64(5), int64(2), uint8(4)
memory usage: 364.3 KB
```

Below are the total number of variables after creating dummy variables.

```
Index(['Academic_Score', 'Score_on_Plays_Made', 'Missed_Play_Score',
       'Injury_Propensity', 'School_Score', 'Overall_Score', 'Scholarship',
       'School_Type_C', 'School_Type_D', 'Region_Southern', 'Region_Western'],
      dtype='object')
```

Step 4: Building Model

MODEL 1

A: Importing stats library

```
import statsmodels.formula.api as sm
```

B: Setting dataset for building model and fitting model1

```
Logistic_1 = sm.logit(formula=model_1,data=df_dummy).fit()
```

```
Optimization terminated successfully.
         Current function value: 0.412189
         Iterations 7
```

C: Model Evaluation Summary

```
Logistic_1.summary()
```

Logit Regression Results

Dep. Variable:	Scholarship	**No. Observations:**	6215
Model:	Logit	**Df Residuals:**	6204
Method:	MLE	**Df Model:**	10
Date:	Tue, 13 Apr 2021	**Pseudo R-squ.:**	0.3645
Time:	22:53:23	**Log-Likelihood:**	-2561.8
converged:	True	**LL-Null:**	-4031.1
Covariance Type:	nonrobust	**LLR p-value:**	0.000

	coef	std err	z	P>|z|	[0.025	0.975]
Intercept	-8.9368	0.536	-16.677	0.000	-9.987	-7.886
Academic_Score	0.4449	0.039	11.268	0.000	0.368	0.522
Score_on_Plays_Made	5.2091	0.296	17.573	0.000	4.628	5.790
Missed_Play_Score	-1.5331	0.315	-4.872	0.000	-2.150	-0.916
Injury_Propensity	-0.6080	0.042	-14.525	0.000	-0.690	-0.526
School_Score	2.9063	0.281	10.333	0.000	2.355	3.458
Overall_Score	0.2283	0.043	5.312	0.000	0.144	0.313
School_Type_C	1.3720	0.121	11.343	0.000	1.135	1.609
School_Type_D	2.2691	0.199	11.401	0.000	1.879	2.659
Region_Southern	-0.4410	0.088	-5.009	0.000	-0.614	-0.268
Region_Western	0.0400	0.088	0.456	0.648	-0.132	0.212

D: Check for multicollinearity in the predictor variables using Variance Inflation Factor (VIF).

```
Academic_Score  VIF =  1.79
Score_on_Plays_Made  VIF =  1.55
Missed_Play_Score  VIF =  1.51
Injury_Propensity  VIF =  1.83
School_Score  VIF =  1.26
Overall_Score  VIF =  2.02
School_Type_C  VIF =  2.98
School_Type_D  VIF =  4.52
Region_Southern  VIF =  1.23
Region_Western  VIF =  1.25
```

Insight (MODEL 1):

1. p-value of Region Western seems to be higher than the significance value of 0.05, hence **dropping the same** as it is insignificant in presence of other variables.
2. *Mc.fadden **R-squared value** for full data is* - **0.3645 i.e. 36 %**

MODEL 2

A: Setting dataset for building model and fitting model_2

```
Logistic_2 = sm.logit(formula=model_2,data=df_dummy).fit()
```

```
Optimization terminated successfully.
         Current function value: 0.412205
         Iterations 7
```

B: Model Evaluation Summary

```
Logistic_2.summary()
```

Logit Regression Results

Dep. Variable:	Scholarship	**No. Observations:**	6215
Model:	Logit	**Df Residuals:**	6205
Method:	MLE	**Df Model:**	9
Date:	Tue, 13 Apr 2021	**Pseudo R-squ.:**	0.3645
Time:	22:53:23	**Log-Likelihood:**	-2561.9
converged:	True	**LL-Null:**	-4031.1
Covariance Type:	nonrobust	**LLR p-value:**	0.000

	coef	std err	z	P>|z|	[0.025	0.975]
Intercept	-8.9238	0.535	-16.674	0.000	-9.973	-7.875
Academic_Score	0.4454	0.039	11.281	0.000	0.368	0.523
Score_on_Plays_Made	5.2320	0.292	17.909	0.000	4.659	5.805
Missed_Play_Score	-1.5369	0.314	-4.888	0.000	-2.153	-0.921
Injury_Propensity	-0.6094	0.042	-14.597	0.000	-0.691	-0.528
School_Score	2.9067	0.281	10.334	0.000	2.355	3.458
Overall_Score	0.2281	0.043	5.306	0.000	0.144	0.312
School_Type_C	1.3675	0.121	11.343	0.000	1.131	1.604
School_Type_D	2.2674	0.199	11.397	0.000	1.877	2.657
Region_Southern	-0.4552	0.082	-5.528	0.000	-0.617	-0.294

C: Check for multicollinearity in the predictor variables using Variance Inflation Factor (VIF).

```
Academic_Score  VIF =  1.79
Score_on_Plays_Made  VIF =  1.5
Missed_Play_Score  VIF =  1.51
Injury_Propensity  VIF =  1.83
School_Score  VIF =  1.26
Overall_Score  VIF =  2.02
School_Type_C  VIF =  2.96
School_Type_D  VIF =  4.52
Region_Southern  VIF =  1.09
```

Insights (MODEL 2):

1. VIF value 4.52 for *School_Type_D* indicates having HIGH multicollinearity and therefore dropping School_Type_D.
2. *McFadden* ***R-squared value*** *for full data is* - **0.3645 i.e., 36 %**

MODEL 3

A: Setting dataset for building model and fitting model_3

```
Optimization terminated successfully.
         Current function value: 0.423189
         Iterations 7
```

B: Model Evaluation Summary

Logit Regression Results

Dep. Variable:	Scholarship	**No. Observations:**	6215
Model:	Logit	**Df Residuals:**	6206
Method:	MLE	**Df Model:**	8
Date:	Tue, 13 Apr 2021	**Pseudo R-squ.:**	0.3475
Time:	22:53:24	**Log-Likelihood:**	-2630.1
converged:	True	**LL-Null:**	-4031.1
Covariance Type:	nonrobust	**LLR p-value:**	0.000

	coef	std err	z	P>|z|	[0.025	0.975]
Intercept	-7.2733	0.499	-14.573	0.000	-8.252	-6.295
Academic_Score	0.6173	0.036	17.120	0.000	0.547	0.688
Score_on_Plays_Made	5.9004	0.287	20.589	0.000	5.339	6.462
Missed_Play_Score	-1.4594	0.312	-4.678	0.000	-2.071	-0.848
Injury_Propensity	-0.3577	0.034	-10.517	0.000	-0.424	-0.291
School_Score	3.8427	0.269	14.287	0.000	3.316	4.370
Overall_Score	-0.0744	0.033	-2.229	0.026	-0.140	-0.009
School_Type_C	0.3282	0.075	4.365	0.000	0.181	0.476
Region_Southern	-0.4672	0.081	-5.751	0.000	-0.626	-0.308

C: Check for multicollinearity in the predictor variables using Variance Inflation Factor (VIF).

```
Academic_Score  VIF =  1.43
Score_on_Plays_Made  VIF =  1.43
Missed_Play_Score  VIF =  1.51
Injury_Propensity  VIF =  1.27
School_Score  VIF =  1.18
Overall_Score  VIF =  1.28
School_Type_C  VIF =  1.11
Region_Southern  VIF =  1.09
```

Insight (MODEL 3):

1. Academic Score, Score on Plays Made, Missed Play Score, Injury Propensity, School Score, Overall Score, School Type and Region Southern are the significant variable to predict Scholarship variable.
2. *McFadden **R-squared value** for full data is* - **0.3475 i.e., 34 %**
3. ***p-values*** for all the coefficients seem to be **less than the significance level of 0.05.** - meaning that all the predictors are statistically significant.

Step 5: Accuracy Score and Predict the class

A: Import libraries

```
from sklearn.linear_model import LogisticRegression
```

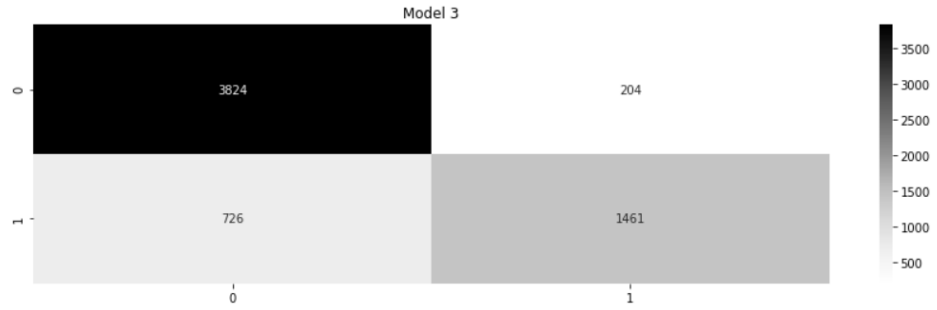
B: Model Score, Class predict and probability

```
Accuracy Score of Model 3: 0.8503620273531778
predict class of Model 3: [0 0 0 ... 1 1 1]
predict class probabilities of Model 3: [[0.9038492  0.0961508 ]
 [0.72390277 0.27609723]
 [0.73309812 0.26690188]
 ...
 [0.17865599 0.82134401]
 [0.18708237 0.81291763]
 [0.12574284 0.87425716]]
```

C : Interpret Logistic Model

	odds_ratio	Probability	variable
1	365.183	[0.9972691213999313]	Score_on_Plays_Made
4	46.649	[0.9790131906009052]	School_Score
0	1.854	[0.6496138816963952]	Academic_Score
6	1.388	[0.5813126913746783]	School_Type_C
5	0.928	[0.4813990539954299]	Overall_Score
3	0.699	[0.41152313286725567]	Injury_Propensity
7	0.627	[0.38526915080726054]	Region_Southern
2	0.232	[0.1885625286228554]	Missed_Play_Score

D: Confusion Matrix



E: Classification Report (Accuracy with the cut off probability 0.50)

Model 3:

	precision	recall	f1-score	support
0	0.84	0.95	0.89	4028
1	0.88	0.67	0.76	2187
accuracy			0.85	6215
macro avg	0.86	0.81	0.83	6215
weighted avg	0.85	0.85	0.84	6215

F : Maximising the accuracy through different cut off-probability

0.1

Accuracy Score 0.493
F1 Score 0.556

0.2

Accuracy Score 0.681
F1 Score 0.637

0.3

Accuracy Score 0.781
F1 Score 0.704

0.4

Accuracy Score 0.831
F1 Score 0.747

0.5

Accuracy Score 0.85
F1 Score 0.759

0.6

Accuracy Score 0.851
F1 Score 0.747

0.7

Accuracy Score 0.839
F1 Score 0.712

0.8

Accuracy Score 0.812
F1 Score 0.64

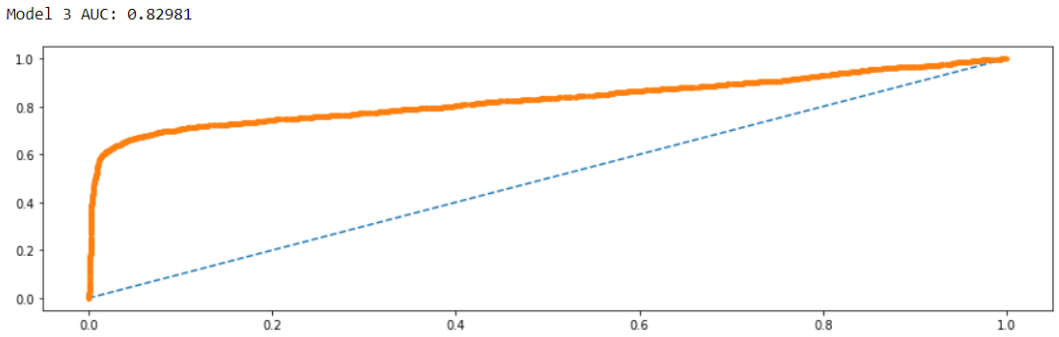
0.9

Accuracy Score 0.748
F1 Score 0.445

G: Saving the classification report in dataset (for consolidation)

```
m3_cm=classification_report(df_dummy['Scholarship'],model_3_class,output_dict=True)

#transpose the dataset
m3_cm_temp=pd.DataFrame(m3_cm).transpose()

#fetch the key metrics from the table
m3_f1=round(m3_cm_temp.loc["1"][2],2)
m3_recall=round(m3_cm_temp.loc["1"][1],2)
m3_precision=round(m3_cm_temp.loc["1"][0],2)
```

H: Model 3 AUC – ROC curve



	Full Data
AUC	0.8298
Accuracy	0.8504
Recall	0.6700
Precision	0.8800
F1 Score	0.7600

Final Insights for Question 2:

- ***MODEL 3*** is taken as the **final model** for the full dataset as the existing independent variables are significant and no multicollinearity exists that would affect the performance of the model. The ideal *McFadden* R-squared score is 40 %; therefore, to increase the R-squared value we would have to work on adding additional Independent variables or increase the number of observations.
- We see that **0.5 and 0.6** gives **better accuracy** than the rest of the cut-off values. But **0.5 cut-off gives us the best 'f1-score'**. So, we conclude that the **cut-off as 0.5 (which is by default)** to get the optimum 'f1' score.
- Evaluation Metrics:

  - AUC Score: **82.98%**
  - Accuracy: **85.04%**
  - Recall: **67%**
  - Precision : **88%**
  - F1-score: **76%**

2.3 Split the data into training (70%) and test (30%). Build the various iterations of the Linear Regression models on the training data and use those models to predict on the test data using appropriate model evaluation metrics.

Objective:

- If prediction accuracy of the **full scholarship is the only objective**, then you may want to divide the data into a **training and a test set, chosen randomly, and use the training set to develop a model** and **test set to validate your model**.
- Use the **models developed in Part (II)** to compare accuracy in **training and test sets**.
- Compare the **final model of Part (II) and the proposed one in Part (III)**.
- Which model provides the most accurate prediction? If the model found in **Part (II) is different from the proposed model in Part (III), give an explanation**.

A: Import libraries appropriately

```
from sklearn.model_selection import train_test_split
```

B: Train &Test split

```
df_train: (4350, 11)
df_test: (1865, 11)
Total Count: 6215
```

C: Model Building

```
from sklearn.linear_model import LogisticRegression
LR = LogisticRegression(solver='newton-cg',penalty='none')
```

D: Building model on training data and checking the accuracy score on the training and test data

Model 1 – Accuracy Score

```
Accuracy Score of Model 1 train dataset: 0.8549425287356321

Accuracy Score of Model 1 test dataset: 0.8680965147453084
```

Model 2

```
Accuracy Score of Model 2 train dataset: 0.8549425287356321

Accuracy Score of Model 2 test dataset: 0.8680965147453084
```

Model 3

```
Accuracy Score of Model 3 train dataset: 0.8473563218390805
Accuracy Score of Model 3 test dataset: 0.8536193029490616
```

E: Predicting the classes and the probabilities on the Test Data

Model 1

```
predict class of Model 1 test dataset: [0 0 0 ... 1 0 1]
predict class probabilities of Model 1 test dataset: [[0.85628504 0.14371496]
 [0.9030674  0.0969326 ]
 [0.9793763  0.0206237 ]
 ...
 [0.10167988 0.89832012]
 [0.93095574 0.06904426]
 [0.09569543 0.90430457]]
```

Model 2

```
predict class of Model 2 test dataset: [0 0 0 ... 1 0 1]
predict class probabilities of Model 2 test dataset: [[0.85625257 0.14374743]
 [0.90314156 0.09685844]
 [0.97937533 0.02062467]
 ...
 [0.10175265 0.89824735]
 [0.93084797 0.06915203]
 [0.09560349 0.90439651]]
```

Model 3

```
predict class of Model 3 train dataset: [0 0 1 ... 0 0 0]
predict class probabilities of Model 3 traint dataset: [[0.71861068 0.28138932]
 [0.83898184 0.16101816]
 [0.16585226 0.83414774]
 ...
 [0.88525878 0.11474122]
 [0.9272579  0.0727421 ]
 [0.97241002 0.02758998]]

predict class of Model 3 test dataset: [0 0 0 ... 1 0 1]
predict class probabilities of Model 3 test dataset: [[0.87503534 0.12496466]
 [0.86830358 0.13169642]
 [0.96260455 0.03739545]
 ...
 [0.0740383  0.9259617 ]
 [0.89761643 0.10238357]
 [0.14988223 0.85011777]]
```

F: Accuracy summarised

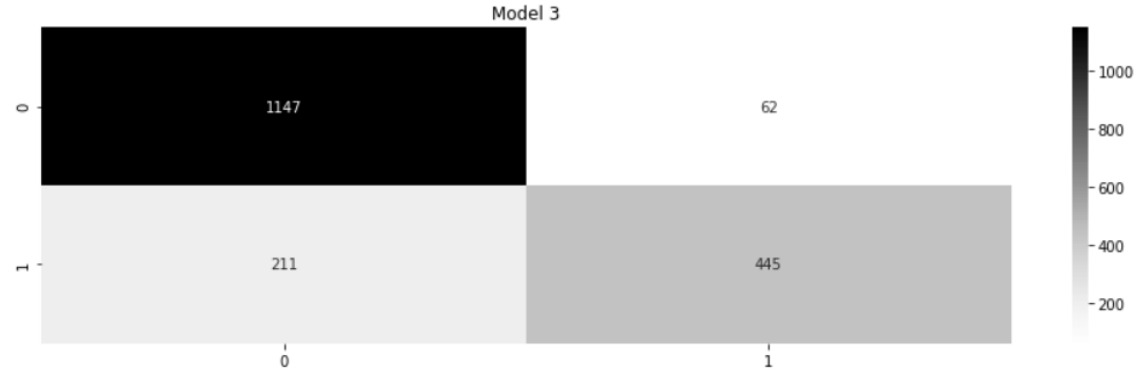
	Accuracy
Model 1_Train	0.855
Model 1_Test	0.868
Model 2_Train	0.855
Model 2_Test	0.868
Model 3_Train	0.847
Model 3_Test	0.854

Insights: Accuracy score difference of training and test set is minimal (i.e., 0.7) for MODEL 3 and therefore MODEL 3 is the proposed model.

G: Interpret Logistic Model_3 and confusion matrix on test data

	odds_ratio	Probability	variable
1	306.256	[0.9967453845803259]	Score_on_Plays_Made
4	43.212	[0.9773818255238618]	School_Score
0	1.913	[0.6567241969800917]	Academic_Score
6	1.417	[0.5862720846754061]	School_Type_C
5	0.930	[0.4818945501437915]	Overall_Score
3	0.725	[0.42021085183354595]	Injury_Propensity
7	0.604	[0.37657373544625317]	Region_Southern
2	0.185	[0.15578208942750638]	Missed_Play_Score

H: Confusion Matrix



I: Classification Report on test data

```
Model 3
              precision    recall  f1-score   support

           0       0.84      0.95      0.89      1209
           1       0.88      0.68      0.77       656

    accuracy                           0.85      1865
   macro avg       0.86      0.81      0.83      1865
weighted avg       0.86      0.85      0.85      1865
```

J: Saving classification report in dataset (for consolidation)

```
#save the classification report in dataset
m3_train_cm=classification_report(df_train['Scholarship'],model_3_train_class,output_dict=True)

#transpose the dataset
m3_cm_train_temp=pd.DataFrame(m3_train_cm).transpose()

#fetch the key metrics from the table
m3_train_f1=round(m3_cm_train_temp.loc["1"][2],2)
m3_train_recall=round(m3_cm_train_temp.loc["1"][1],2)
m3_train_precision=round(m3_cm_train_temp.loc["1"][0],2)
```

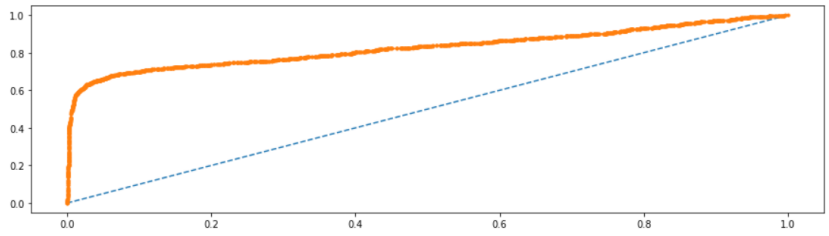
```
#save the classification report in dataset
m3_test_cm=classification_report(df_test['Scholarship'],model_3_test_class,output_dict=True)

#transpose the dataset
m3_cm_test_temp=pd.DataFrame(m3_test_cm).transpose()

#fetch the key metrics from the table
m3_test_f1=round(m3_cm_test_temp.loc["1"][2],2)
m3_test_recall=round(m3_cm_test_temp.loc["1"][1],2)
m3_test_precision=round(m3_cm_test_temp.loc["1"][0],2)
```

K: Maximising the accuracy through different cut off-probability

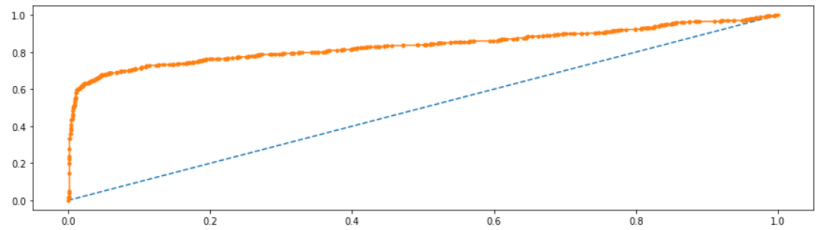
```
0.1

Accuracy Score 0.481
F1 Score 0.55

0.2

Accuracy Score 0.677
F1 Score 0.634

0.3

Accuracy Score 0.777
F1 Score 0.699

0.4

Accuracy Score 0.827
F1 Score 0.741

0.5

Accuracy Score 0.847
F1 Score 0.753

0.6

Accuracy Score 0.849
F1 Score 0.742

0.7

Accuracy Score 0.837
F1 Score 0.706

0.8

Accuracy Score 0.81
F1 Score 0.635

0.9

Accuracy Score 0.747
F1 Score 0.443
```

L: AUC – ROC curve on the train and test data

Model 3 Train AUC: 0.82700



Model 3 Test AUC: 0.83566



M: Comparison table for the scores of full data set(Q2) and train & test data set(Q3)

	Full Data_final model	Train Data_final model	Test Data_final model
AUC	0.8298	0.8270	0.8357
Accuracy	0.8504	0.8474	0.8536
Recall	0.6700	0.6600	0.6800
Precision	0.8800	0.8700	0.8800
F1 Score	0.7600	0.7500	0.7700

Final Insights for Question 3:

1. We have used **all three models which were developed during question2** and based on that mentioned below were **the result for model accuracy**:
    - Model1 Accuracy: (Train: **85.5%**, Test: **86.8%**)
    - Model2 Accuracy: (Train: **85.5%**, Test: **86.8%**)
    - Model3 Accuracy: (Train: **84.7%**, Test: **85.4%**)

    We are choosing **Model3 as final version** because:

        - Model is significant (p-value less than 0.05 for all variables)
        - Model is free of multicollinearity
        - Accuracy variation between train and test is very minimal
2. Compared Q2 final model and Q3 final model and mentioned below is the inferences:
    - Q3 model has higher AUC score, Accuracy, Recall than Q2 AUC
3. Further, we have checked the custom cut-off to see if we could increase the accuracy and f1-score and found, **0.5 and 0.6** gives **better accuracy** than the rest of the custom cut-off values. But **0.5 cut-off gives us the best 'f1-score'**. Here, we will take the **cut-off as 0.5 (which is by default)** to get the optimum 'f1' score.

Hence, we suggest to go with Question3 train and test split model.

2.4 Use the same training-test data split in Part (3) to develop a suitable linear discriminant model. Use the same to predict discriminant scores for the test data. Compare the final output from the logistic regression model and LDA.

A: Checking the dimensions of the training and test data

```
df_train: (4350, 11)
df_test: (1865, 11)
Total Count: 6215
```

```
0   0.6480
1   0.3520
Name: Scholarship, dtype: float64
```

B: Overall significance of the model using MANOVA

Import libraries and define the hypothesis

```
from statsmodels.multivariate.manova import MANOVA
```

HYPOTHESIS:

H0 (NULL): No independent variables is significant discriminator of predictor variable (scholarship)

Ha (ALTERNATE): Atleast one independent variables is significant discriminator of predictor variable (scholarship)

Build MANOVA model

```
                  Multivariate linear model
================================================================

----------------------------------------------------------------
       Intercept         Value    Num DF   Den DF   F Value   Pr > F
----------------------------------------------------------------
          Wilks' lambda   0.0072 10.0000 4339.0000 59993.6653 0.0000
         Pillai's trace   0.9928 10.0000 4339.0000 59993.6653 0.0000
 Hotelling-Lawley trace 138.2661 10.0000 4339.0000 59993.6653 0.0000
    Roy's greatest root 138.2661 10.0000 4339.0000 59993.6653 0.0000
----------------------------------------------------------------

----------------------------------------------------------------
          Scholarship       Value   Num DF   Den DF  F Value  Pr > F
----------------------------------------------------------------
             Wilks' lambda 0.5887 10.0000 4339.0000 303.0965 0.0000
            Pillai's trace 0.4113 10.0000 4339.0000 303.0965 0.0000
    Hotelling-Lawley trace 0.6985 10.0000 4339.0000 303.0965 0.0000
       Roy's greatest root 0.6985 10.0000 4339.0000 303.0965 0.0000
================================================================
```

C: Linear Discriminant Analysis(LDA) using scikit learn

- Define train and test dataset

```
X_train = df_train[['Academic_Score','Score_on_Plays_Made','Missed_Play_Score','Injury_Propensity','School_Score','Overall_Score
Y_train = df_train['Scholarship']
```

```
X_test = df_test[['Academic_Score','Score_on_Plays_Made','Missed_Play_Score','Injury_Propensity','School_Score','Overall_Score',
Y_test = df_test['Scholarship']
```

- Build LDA model

```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as LDA

lda = LDA(n_components=1)
LDA_model = lda.fit(X_train, Y_train)
```

- Find out the group means

	Academic_Score	Score_on_Plays_Made	Missed_Play_Score	Injury_Propensity	School_Score	Overall_Score	School_Type_C	School_Type_D	Region_Southe
0	6.8600	0.2800	0.3400	2.3600	0.4900	10.4800	0.5000	0.1600	0.31
1	7.9200	0.4400	0.3000	1.6900	0.6000	10.4300	0.6100	0.2700	0.19

- Then checking the prior probabilities

```
#Prior probabilities
LDA_model.priors_
```

```
array([0.64804598, 0.35195402])
```

- Model Predictions

Predict Class (Train) and Predict Class (Test)

```
#Predict Class (Train)
pred_train=LDA_model.predict(X_train)
pred_train[:10]
```

```
array([0, 0, 1, 1, 0, 0, 0, 0, 0, 1], dtype=int64)
```

```
#Predict Class (Test)
pred_test=LDA_model.predict(X_test)
pred_test[:10]
```

```
array([0, 0, 0, 0, 0, 0, 0, 1, 0, 0], dtype=int64)
```

❖ **Predict Probability (Train) and Predict Probability (Test)**

Train

```
array([[0.66, 0.34],
       [0.89, 0.11],
       [0.12, 0.88],
       ...,
       [0.95, 0.05],
       [0.97, 0.03],
       [0.99, 0.01]])
```

Test

```
array([[0.9 , 0.1 ],
       [0.95, 0.05],
       [0.99, 0.01],
       ...,
       [0.09, 0.91],
       [0.97, 0.03],
       [0.08, 0.92]])
```

❖ Between-groups variance and within-groups variance for a variable

```
calcSeparations1(X_train, Y_train)
```

	XVar	Vw	Vb	wilks
0	Academic_Score	1.4500	1,126.1300	0.0010
1	Score_on_Plays_Made	0.0200	26.8800	0.0010
4	School_Score	0.0200	12.4300	0.0020
3	Injury_Propensity	1.2500	457.2100	0.0030
9	Region_Western	0.1900	20.9000	0.0090
8	Region_Southern	0.1900	14.3600	0.0130
2	Missed_Play_Score	0.0200	1.4500	0.0140
7	School_Type_D	0.1600	11.4600	0.0140
6	School_Type_C	0.2500	12.3600	0.0200
5	Overall_Score	1.3800	2.4200	0.3630

❖ Model Performance Evaluation – LDA

Accuracy Score

Train

```
0.8581609195402299
```

Test

```
0.868632707774799
```

❖ **Confusion Matrix**

```
array([[1163,   46],
       [ 199,  457]], dtype=int64)
```

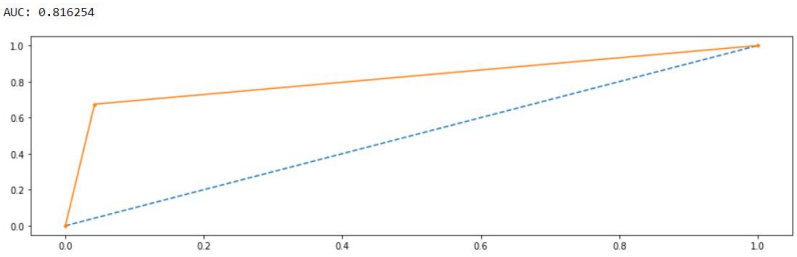
❖ **Classification Report: Train**

	precision	recall	f1-score	support
0	0.84	0.96	0.90	2819
1	0.90	0.67	0.77	1531
accuracy			0.86	4350
macro avg	0.87	0.82	0.83	4350
weighted avg	0.86	0.86	0.85	4350

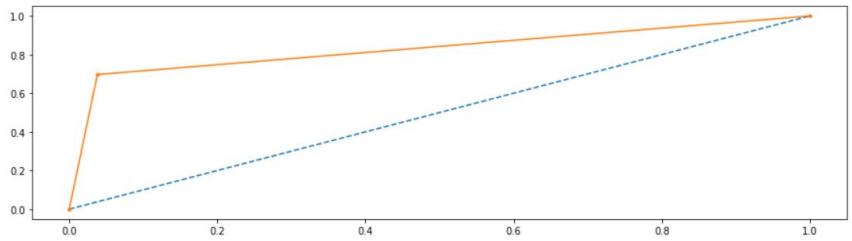
❖ **Classification Report: Test**

	precision	recall	f1-score	support
0	0.85	0.96	0.90	1209
1	0.91	0.70	0.79	656
accuracy			0.87	1865
macro avg	0.88	0.83	0.85	1865
weighted avg	0.87	0.87	0.86	1865

❖ **AUC-ROC score for Train data**



❖ **AUC-ROC score for Test data**:



Insights for Q4:

1. We have used **train and test split data and developed LDA model** and mentioned below is the summary around the same:
    - Overall significance of the model using MANOVA is significant which means *at least one independent variable is significant discriminator of predictor variable* ***(scholarship)***
    - We performed **group means, prior model probability, predicted classes & probabilities** for **train and test data of independent variables**.
    - We also calculated **Discriminant Scores**
    - We also checked **wilk's lambda score** *(score of extent of "separation")* for each independent variable
2. LDA model performance:
    - AUC: (Train: **81.6%**, Test: **82.9%**)
    - Accuracy: (Train: **85.8%**, Test: **86.9%**)
    - Recall: (Train: **67.0%**, Test: **70.0%**)
    - Precision: (Train: **90.0%**, Test: **91.0%**)
    - F1-socre: (Train: **77.0%**, Test: **79.0%**)

Comparison of both approaches - Logistic Regression and Linear Discriminant Analysis

	Train Data_Logistic model	Test Data_Logistic model	Train Data_LDA model	Test Data_LDA model
AUC	0.827	0.836	0.816	0.829
Accuracy	0.847	0.854	0.858	0.869
Recall	0.660	0.680	0.670	0.700
Precision	0.870	0.880	0.900	0.910
F1 Score	0.750	0.770	0.770	0.790

```
Logistic Model:          LDA Model:
True Negative: 1147      True Negative: 1163
False Positives: 62      False Positives: 46
False Negatives: 211     False Negatives: 199
True Positives: 445      True Positives: 457
```

Findings to select which model is more suitable:

1. LDA model has less false negatives (i.e., **199**) (which means model has predicted "partial scholarship" but actual was "full scholarship") than Logistic model (i.e., **211**). In other words, LDA model has **6% less Type-I error** than Logistic model), hence better for the high school from the financial analysis perspective.

2. LDA model has higher accuracy, F1-score than logistic model.

The findings above lead us to the conclusion **LDA model is more suitable than Logistic model.**

Business Interpretation

- The important factors that is instrumental in winning FULL scholarship are Academic Score, School Score, Overall Score, School Score and School on Plays Made score. This was analysed in the EDA and the pvalue computed in building the model.
- The model accuracy on the training as well as the test set is ~86%, which is roughly the same proportion as the *class 1 observations* in the dataset. This model is affected by a class imbalance problem. Since we only have 6215 observations, if re-build the same LDA model with *more number of data points, an even better model could be built*.