greatlearning
Power Ahead

Predictive Modelling

20th Feb 2022
Data Science and Business Analytics
(PGP-DSBA)
Online September_A 2021-22

Made By:
Anmol Tripathi

Table of Contents:

Problem 1:

S No:	Description:	Page No:
I.	Executive Summary	9
II.	Background of the problem	9
III.	Data Dictionary	9 - 10
IV.	Data Description	10
V.	Sample dataset	11
1.1	Read the data and do exploratory data analysis. Describe the data briefly. (Check the null values, Data types, shape, EDA, duplicate values). Perform Univariate and Bivariate Analysis.	12 - 25
1.2	Impute null values if present, also check for the values which are equal to zero. Do they have any meaning or do we need to change them or drop them? Check for the possibility of combining the sub levels of a ordinal variables and take actions accordingly. Explain why you are combining these sub levels with appropriate reasoning.	26 - 28
1.3	Encode the data (having string values) for Modelling. Split the data into train and test (70:30). Apply Linear regression using scikit learn. Perform checks for significant variables using appropriate method from statsmodel. Create multiple models and check the performance of Predictions on Train and Test sets using Rsquare, RMSE & Adj Rsquare. Compare these models and select the best one with appropriate reasoning.	29 - 38
1.4	Inference: Basis on these predictions, what are the business insights and recommendations.	39 - 40

Problem 2:

S No:	Description:	Page No:
I.	Executive Summary	41
II.	Background of the problem	41
III.	Data Dictionary	41
IV.	Data Description	41 to 42
V.	Sample dataset	42
2.1	Data Ingestion: Read the dataset. Do the descriptive statistics and do null value condition check, write an inference on it? Perform Univariate and Bivariate Analysis. Do exploratory data analysis.	43 to 58
2.2	Do not scale the data. Encode the data (having string values) for Modelling. Data Split: Split the data into train and test (70:30). Apply Logistic Regression and LDA (linear discriminant analysis).	59 to 74
2.3	Performance Metrics: Check the performance of Predictions on Train and Test sets using Accuracy, Confusion Matrix, Plot ROC curve and get ROC_AUC score for each model Final Model: Compare Both the models and write inference which model is best/optimized.	75 to 86
2.4	Inference: Basis on these predictions, what are the insights and recommendations.	87

List of Tables:

S No:	Description:	Page No:
Table 1.1	Shape and size	10
Table 1.2	No null Values	10
Table 1.3	Count of Duplicate Rows	10
Table 1.4	Count of Duplicate Rows	10
Table 1.5	Data type	11
Table 1.6	Data type	11
Table 1.7	Data Set	12
Table 1.8	Columns	12
Table 1.9	Type of Data Set	12
Table 1.10	Shape	13
Table 1.11	Data Type	13
Table 1.12	Dropped the Serial Number Column	13
Table 1.13	summary statistics of the dataset	14
Table 1.14	summary statistics of the object variable	14
Table 1.15	Checking for the values which are equal to zero	14
Table 1.16	Shape	14
Table 1.17	Dropping dimensionless diamonds	14
Table 1.18	Number of duplicate rows	15
Table 1.19	Before and After	15
Table 1.20	Number of duplicate rows	15
Table 1.21	Checking Missing value	15
Table 1.22	Measure of Skewness	21
Table 1.23	How each feature affects the price of diamonds	22
Table 1.24	Null Values	26
Table 1.25	Median Scores	26
Table 1.26	After Imputing Missing values	27
Table 1.27	Shape	27
Table 1.28	Getting unique counts of all Objects	29
Table 1.29	Data Type	30
Table 1.30	After Conversion	30
Table 1.31	Data set	30
Table 1.32	Data set	31

Table 1.33	intercept for the model	32
Table 1.34	intercept for the model	35
Table 1.35	Regression Model Score	35
Table 1.36	Modified Data Set	35
Table 1.37	Fitting the data set	36
Table 1.38	Inferential statistics	36
Table 2.1	Shape	41
Table 2.2	Null Values	42
Table 2.3	Count of Duplicate Rows	42
Table 2.4	Data Information	42
Table 2.5	Data set	42
Table 2.6	Data set	43
Table 2.7	Data set	43
Table 2.8	Data set	43
Table 2.9	Null Values	43
Table 2.10	Shape	44
Table 2.11	Descriptive Analysis	44
Table 2.12	Missing values	44
Table 2.13	Checking for Duplicates	44
Table 2.14	Unique values in the categorical data	45
Table 2.15	Percentage of target	45
Table 2.16	Percentage of target	45
Table 2.17	Modified data set	59
Table 2.18	Test / Train Split	59
Table 2.19	Train Value Count	59
Table 2.20	Test Value Count	60
Table 2.21	Getting the Predicted Classes and Probs	60
Table 2.22	Confusion Matrix for the training data	61
Table 2.23	Classification Report of the Train Data	62
Table 2.24	Classification Report of the Test Data	63
Table 2.25	Classification Report of the Training Data	64
Table 2.26	Confusion Matrix Array	64
Table 2.27	Classification Report of the Testing data	64
Table 2.28	Confusion Matrix	65
Table 2.29	Classification Report of the Default / Custom cut off Test data	74
Table 2.30	Confusion Matrix for the training data	76

Table 2.31	score card	77
Table 2.32	score card	77
Table 2.33	Classification Report of the Training Data	78
Table 2.34	Confusion Matrix	79
Table 2.35	Classification Report of Test data	79
Table 2.36	Confusion Matrix	79
Table 2.37	Classification Report of the Default / Custom cut off Test data	85
Table 2.38	Score Card Comparison	86

List of Graph:

S No:	Description:	Page No:
Graph 1.1	Box Plot	16
Graph 1.2	Box Plot	16
Graph 1.3	Box Plot	16
Graph 1.4	Box Plot	17
Graph 1.5	Box Plot	17
Graph 1.6	Box Plot	17
Graph 1.7	Box Plot	18
Graph 1.8	Box Plot	18
Graph 1.9	Box Plot	18
Graph 1.10	Box Plot	19
Graph 1.11	Box Plot	19
Graph 1.12	Box Plot	19
Graph 1.13	Univariate Analysis	20
Graph 1.14	Bivariate Analysis	21
Graph 1.15	Heat Map	22
Graph 1.16	Count Plot	23
Graph 1.17	Box Plot	23
Graph 1.18	Count Plot	23
Graph 1.19	Box Plot	24
Graph 1.20	Count Plot	24
Graph 1.21	Box Plot	24
Graph 1.22	Scatter Plot	33
Graph 1.23	Scatter Plot	37
Graph 2.1	Dist-Plot || Histogram	46

Graph 2.2	Dist-Plot || Histogram	46
Graph 2.3	Dist-Plot || Histogram	47
Graph 2.4	Dist-Plot || Histogram	47
Graph 2.5	Dist-Plot || Histogram	48
Graph 2.6	Count Plot	48
Graph 2.7	Count Plot	49
Graph 2.8	Swarm Plot	49
Graph 2.9	Swarm Plot	49
Graph 2.10	Swarm Plot	50
Graph 2.11	Swarm Plot	50
Graph 2.12	Swarm Plot	50
Graph 2.13	Swarm Plot	51
Graph 2.14	Swarm Plot	51
Graph 2.15	Swarm Plot	51
Graph 2.16	Swarm Plot	52
Graph 2.17	BIVARITE ANALYSIS	52
Graph 2.18	Box Plot	53
Graph 2.19	Box Plot	53
Graph 2.20	Box Plot	53
Graph 2.21	Box Plot	54
Graph 2.22	Box Plot	54
Graph 2.23	Heat map	55
Graph 2.24	Box Plot	55
Graph 2.25	Box Plot	56
Graph 2.26	Box Plot	56
Graph 2.27	Box Plot	56
Graph 2.28	Box Plot	57
Graph 2.29	Box Plot	57
Graph 2.30	Box Plot	57
Graph 2.31	Box Plot	58
Graph 2.32	Box Plot	58
Graph 2.33	Box Plot	58
Graph 2.34	ROC Curve	61
Graph 2.35	ROC Curve	61
Graph 2.36	Confusion Matrix	62
Graph 2.37	Confusion Matrix	62

Graph 2.38	Confusion Matrix	63
Graph 2.39	ROC Curve	68
Graph 2.40	ROC Curve	68
Graph 2.41	Confusion Matrix	69
Graph 2.42	Confusion Matrix	69
Graph 2.43	Confusion Matrix	70
Graph 2.44	Confusion Matrix	70
Graph 2.45	Confusion Matrix	71
Graph 2.46	Confusion Matrix	71
Graph 2.47	Confusion Matrix	72
Graph 2.48	Confusion Matrix	72
Graph 2.49	Confusion Matrix	73
Graph 2.50	Confusion Matrix	73
Graph 2.51	ROC Curve	75
Graph 2.52	ROC Curve	76
Graph 2.53	Confusion Matrix	76
Graph 2.54	Confusion Matrix	77
Graph 2.55	Confusion Matrix	78
Graph 2.56	ROC Curve	79
Graph 2.57	ROC Curve	80
Graph 2.58	Confusion Matrix	80
Graph 2.59	Confusion Matrix	81
Graph 2.60	Confusion Matrix	81
Graph 2.61	Confusion Matrix	82
Graph 2.62	Confusion Matrix	82
Graph 2.63	Confusion Matrix	83
Graph 2.64	Confusion Matrix	83
Graph 2.65	Confusion Matrix	84
Graph 2.66	Confusion Matrix	84
Graph 2.67	Confusion Matrix	85

Problem 1:

Executive Summary:

The given data set has number columns and rows in them. In this data set we are given dataset containing the prices and other attributes of almost 27,000 cubic zirconia (which is an inexpensive diamond alternative with many of the same qualities as a diamond). The company is earning different profits on different prize slots. Here we also have to help the company in predicting the price for the stone on the bases of the details given in the dataset so it can distinguish between higher profitable stones and lower profitable stones so as to have better profit share.

Background of the problem:

You are hired by a company Gem Stones co ltd, which is a cubic zirconia manufacturer. You are provided with the dataset containing the prices and other attributes of almost 27,000 cubic zirconia (which is an inexpensive diamond alternative with many of the same qualities as a diamond). The company is earning different profits on different prize slots. You have to help the company in predicting the price for the stone on the bases of the details given in the dataset so it can distinguish between higher profitable stones and lower profitable stones so as to have better profit share. Also, provide them with the best 5 attributes that are most important.

Data Dictionary:

Variable Name	Description
Carat	Carat weight of the cubic zirconia.
Cut	Describe the cut quality of the cubic zirconia. Quality is increasing order Fair, Good, Very Good, Premium, Ideal.
Color	Colour of the cubic zirconia.With D being the worst and J the best.
Clarity	Clarity refers to the absence of the Inclusions and Blemishes. (In order from Worst to Best in terms of avg price) IF, VVS1, VVS2, VS1, VS2, Sl1, Sl2, l1
Depth	The Height of cubic zirconia, 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	the 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.

Data Description:

The shape of the data set seems to be with 26967 rows and 11 columns.

```
df.shape
(26967, 11)
```

Table 1.1 Shape and size

- No null values in the dataset.
- The Data set is having float and object Data types.
- There are duplicate values present in the data set. We have removed them from the data set for the further processing.
- One column is having null values which are treated further. Also this is being stated further in the report.

```
Unnamed: 0      int64
carat         float64
cut            object
color          object
clarity        object
depth         float64
table         float64
x             float64
y             float64
z             float64
price           int64
dtype: object
```

Table 1.2 No null Values

```
Number of duplicate rows = 33
(26958, 10)
```

Table 1.3 Count of Duplicate Rows

After doing changes, we iterate the duplicate items:

```
Number of duplicate rows = 0
```

Table 1.4 Count of Duplicate Rows

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 26967 entries, 0 to 26966
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
```

Table 1.5 Data type

Sample Data Set:

	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
5	6	1.02	Ideal	D	VS2	61.5	56.0	6.46	6.49	3.99	9502
6	7	1.01	Good	H	SI1	63.7	60.0	6.35	6.30	4.03	4836
7	8	0.50	Premium	E	SI1	61.5	62.0	5.09	5.06	3.12	1415
8	9	1.21	Good	H	SI1	63.8	64.0	6.72	6.63	4.26	5407
9	10	0.35	Ideal	F	VS2	60.5	57.0	4.52	4.60	2.76	706

Table 1.6 Data type

The above table represents the actual representation of the data set.

1.1 Read the data and do exploratory data analysis. Describe the data briefly. (Check the null values, Data types, shape, EDA, duplicate values). Perform Univariate and Bivariate Analysis.

General Data Set:

	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
5	6	1.02	Ideal	D	VS2	61.5	56.0	6.46	6.49	3.99	9502
6	7	1.01	Good	H	SI1	63.7	60.0	6.35	6.30	4.03	4836
7	8	0.50	Premium	E	SI1	61.5	62.0	5.09	5.06	3.12	1415
8	9	1.21	Good	H	SI1	63.8	64.0	6.72	6.63	4.26	5407
9	10	0.35	Ideal	F	VS2	60.5	57.0	4.52	4.60	2.76	706

Table 1.7 Data Set

```
Index(['Unnamed: 0', 'carat', 'cut', 'color', 'clarity', 'depth', 'table', 'x',
       'y', 'z', 'price'],
      dtype='object')
```

Table 1.8 Columns

Checking the type of the dataset

```
Unnamed: 0      int64
carat         float64
cut            object
color          object
clarity        object
depth         float64
table         float64
x             float64
y             float64
z             float64
price           int64
dtype: object
```

Table 1.9 Type of Data Set

Checking the shape of the dataset:

```
df.shape
```

```
(26967, 11)
```

Table 1.10 Shape

Getting the info data types column wise:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 26967 entries, 0 to 26966
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
```

Table 1.11 Data Type

- The data set contains 26967 row, 11 columns.
- In the given data set, there are 2 Integer type features,6 Float type features.3 Object type features. Where 'price' is the target variable and all other are predector variable.
- The first column is an index ("Unnamed: 0") as this only serial no, we can remove it.
- Except depth, in all the column non null count is 26967.

Dropped the Serial Number Column:

```
Index(['carat', 'cut', 'color', 'clarity', 'depth', 'table', 'x', 'y', 'z',
       'price'],
      dtype='object')
```

Table 1.12 Dropped the Serial Number Column

Getting the summary statistics of the dataset:

	count	mean	std	min	25%	50%	75%	max
carat	26967.0	0.798375	0.477745	0.2	0.40	0.70	1.05	4.50
depth	26270.0	61.745147	1.412860	50.8	61.00	61.80	62.50	73.60
table	26967.0	57.456080	2.232068	49.0	56.00	57.00	59.00	79.00
x	26967.0	5.729854	1.128516	0.0	4.71	5.69	6.55	10.23
y	26967.0	5.733569	1.166058	0.0	4.71	5.71	6.54	58.90
z	26967.0	3.538057	0.720624	0.0	2.90	3.52	4.04	31.80
price	26967.0	3939.518115	4024.864666	326.0	945.00	2375.00	5360.00	18818.00

Table 1.13 summary statistics of the dataset

Getting the summary statistics of the object variable:

	cut	color	clarity
count	26967	26967	26967
unique	5	7	8
top	Ideal	G	SI1
freq	10816	5661	6571

Table 1.14 summary statistics of the object variable

Checking for the values which are equal to zero:

```
Number of rows with x == 0: 3
Number of rows with y == 0: 3
Number of rows with z == 0: 9
Number of rows with depth == 0: 0
```

Table 1.15 Checking for the values which are equal to zero

```
lr_df.shape
```

```
(26967, 10)
```

Table 1.16 Shape

Dropping dimensionless diamonds:

```
lr_df = lr_df.drop(lr_df[lr_df["x"]==0].index)
lr_df = lr_df.drop(lr_df[lr_df["y"]==0].index)
lr_df = lr_df.drop(lr_df[lr_df["z"]==0].index)
lr_df.shape
```

```
(26958, 10)
```

Table 1.17 Dropping dimensionless diamonds

On the given data set the mean and median values does not have much difference. We can observe Min value of "x", "y", & "z" are zero this indicates that they are faulty values. As we know dimensionless or 2-dimensional diamonds are not possible. So, we need to filter out those as it clearly faulty data entries. There are three object data type 'cut', 'color' and 'clarity'.

Performing EDA:

We will follow the below mentioned steps to perform EDA
Step 1: Checking & Removing duplicates.
Step 2: Checking and treating Missing value.
Step 3: Outlier Treatment.
Step 4: Univariate Analysis.
Step 5: Bivariate Analysis.

EDA-Step-1: Checking for duplicate records in the data

We have duplicates present in our data set which has been treated as mention below. For checking the command line, do check out the jupyter notebook.

```
Number of duplicate rows = 33
(26958, 10)
```

Table 1.18 Number of duplicate rows

```
Before (26958, 10)
After (26925, 10)
```

Table 1.19 Before and After

```
Number of duplicate rows = 0
```

Table 1.20 Number of duplicate rows

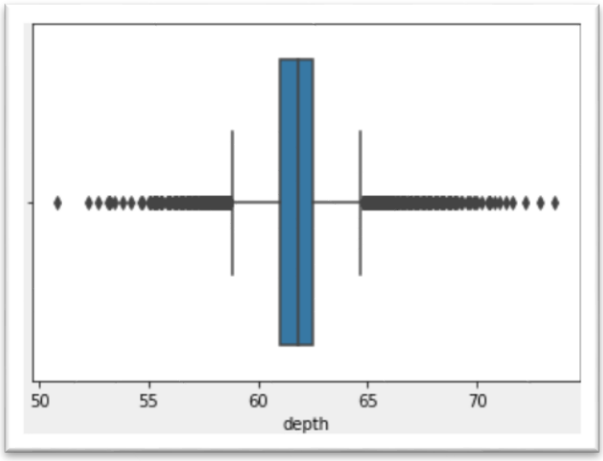
EDA-Step 2: Checking Missing value:

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

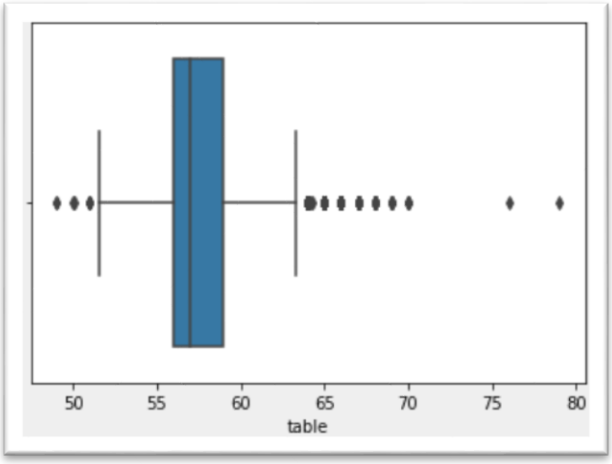
Table 1.21 Checking Missing value

We can observe there are 697 missing values in the depth column. Missing value treatment will be done in section 1.2.

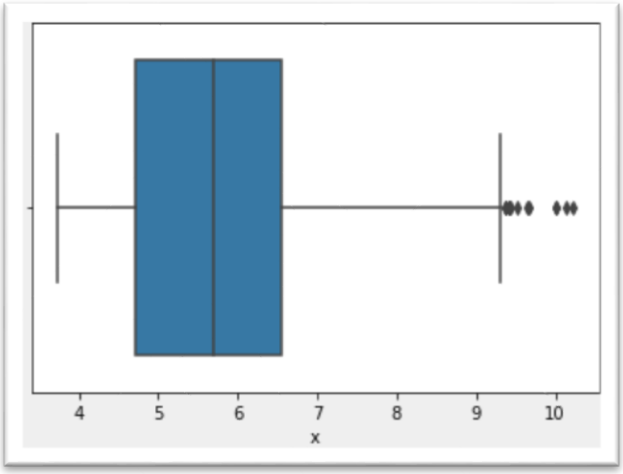
EDA-Step 3: Outlier Checks:



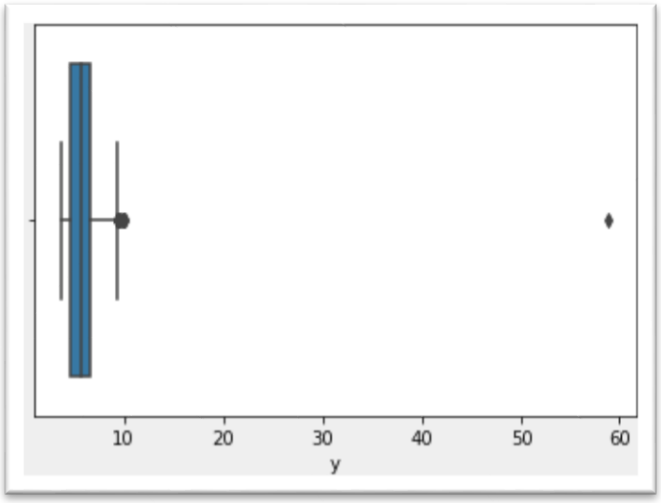
Graph 1.1 Box Plot



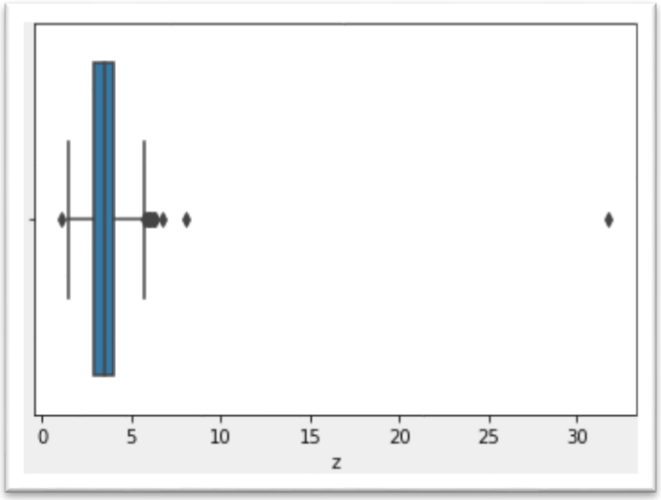
Graph 1.2 Box Plot



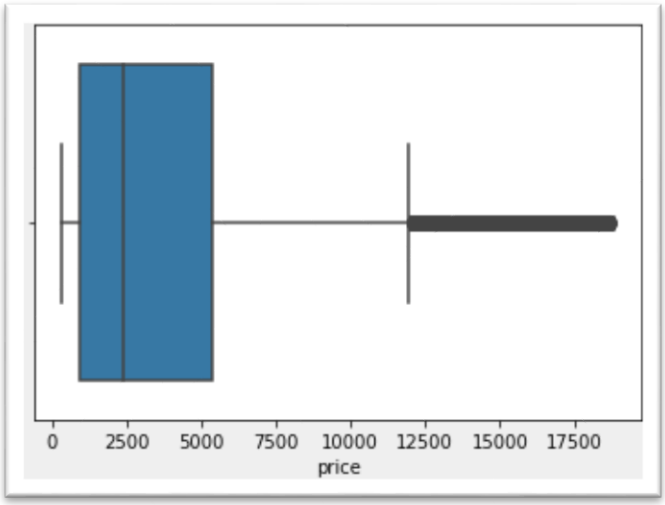
Graph 1.3 Box Plot



Graph 1.4 Box Plot

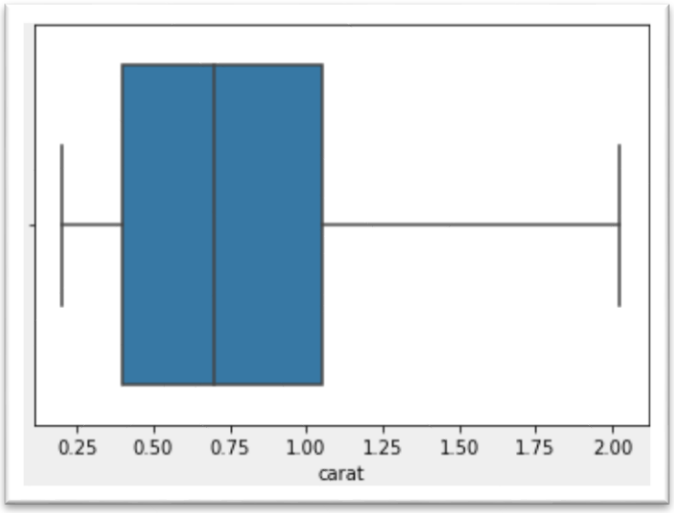


Graph 1.5 Box Plot

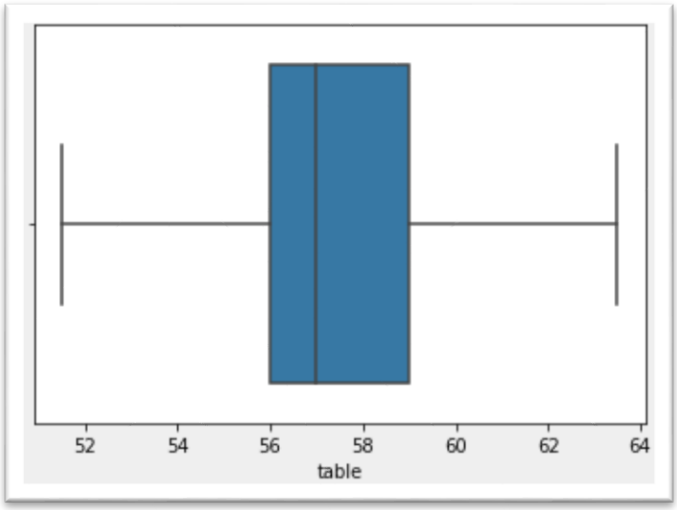


Graph 1.6 Box Plot

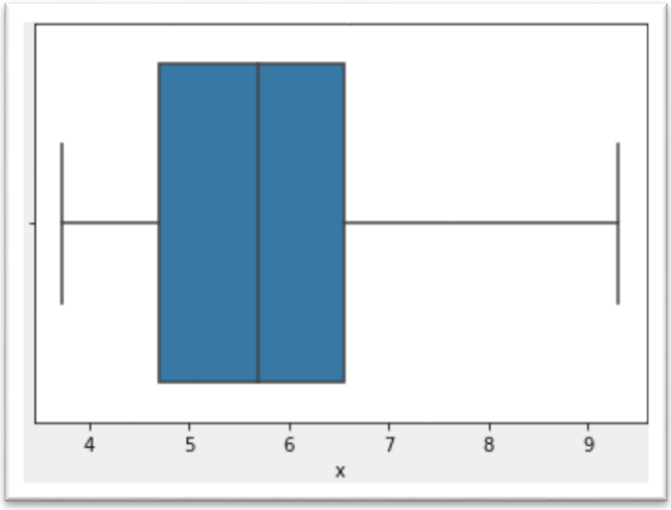
Outlier treatment:



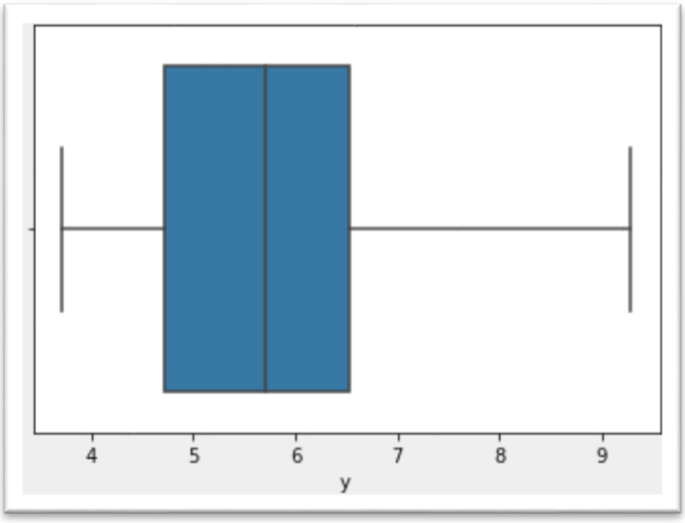
Graph 1.7 Box Plot



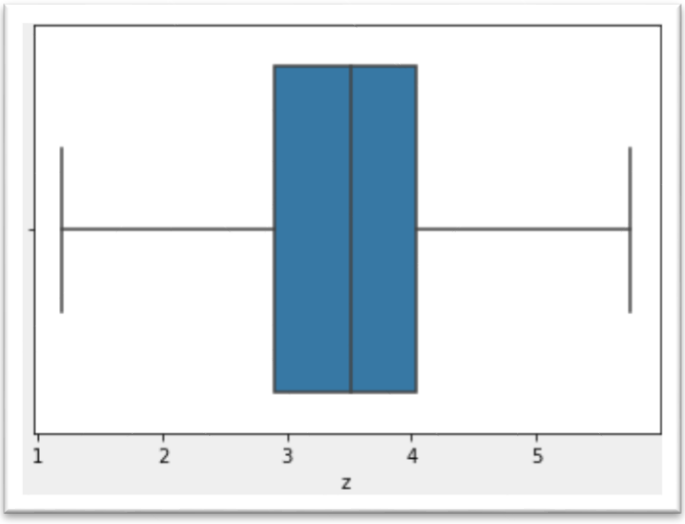
Graph 1.8 Box Plot



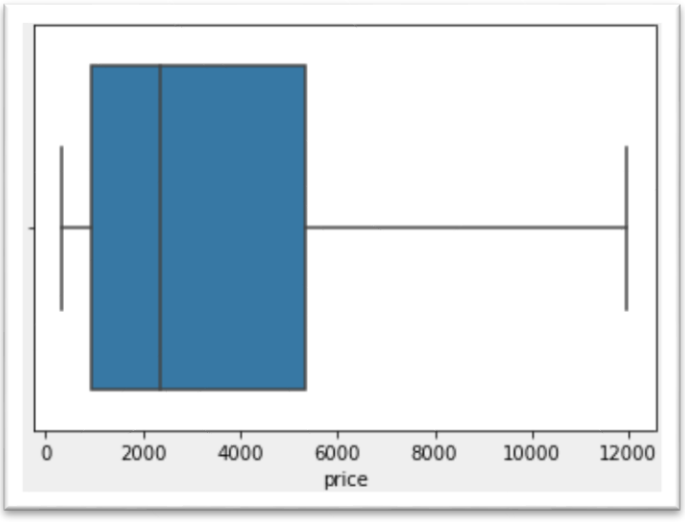
Graph 1.9 Box Plot



Graph 1.10 Box Plot

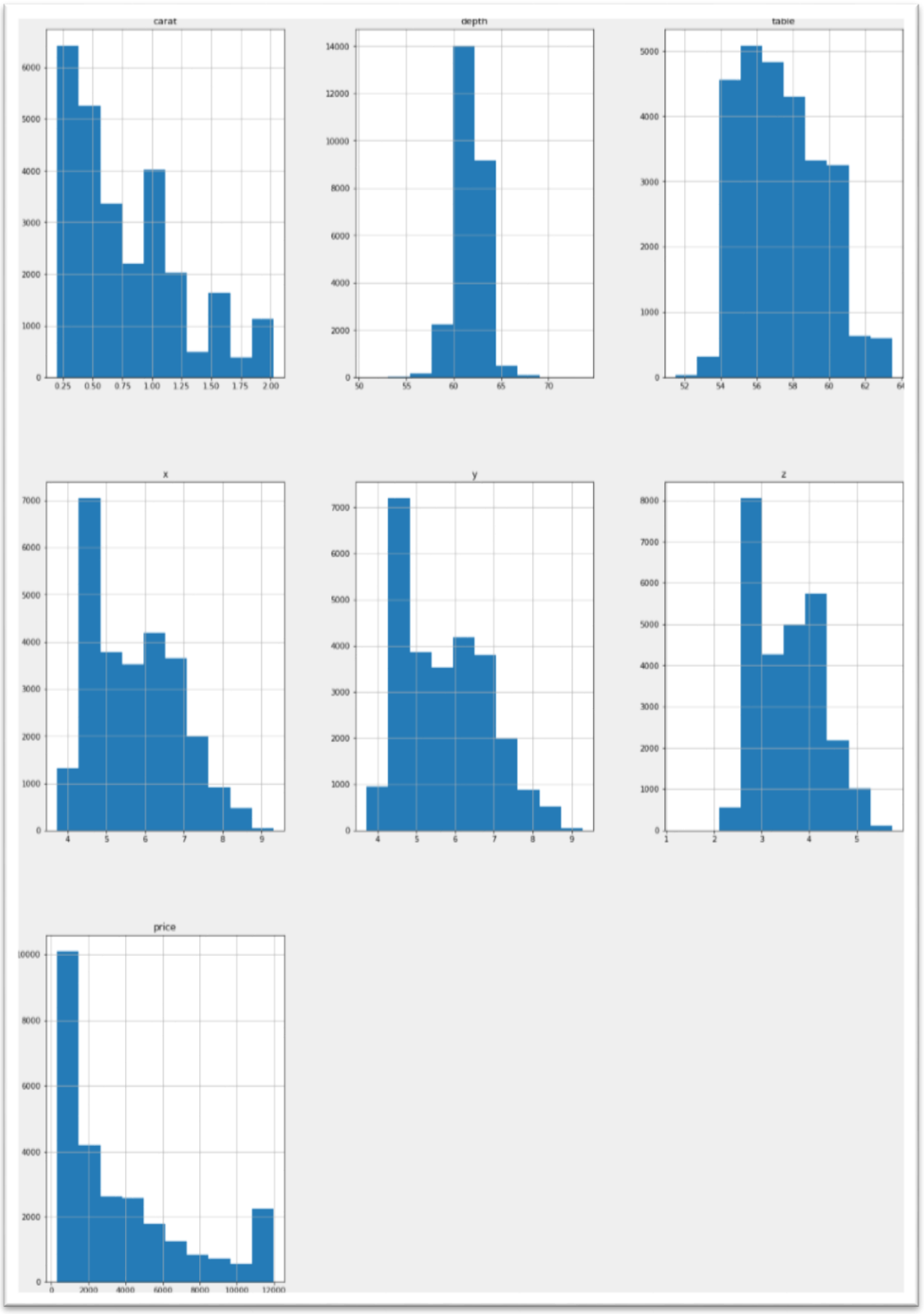


Graph 1.11 Box Plot



Graph 1.12 Box Plot

EDA-Step 4: Univariate Analysis:



Graph 1.13 Univariate Analysis

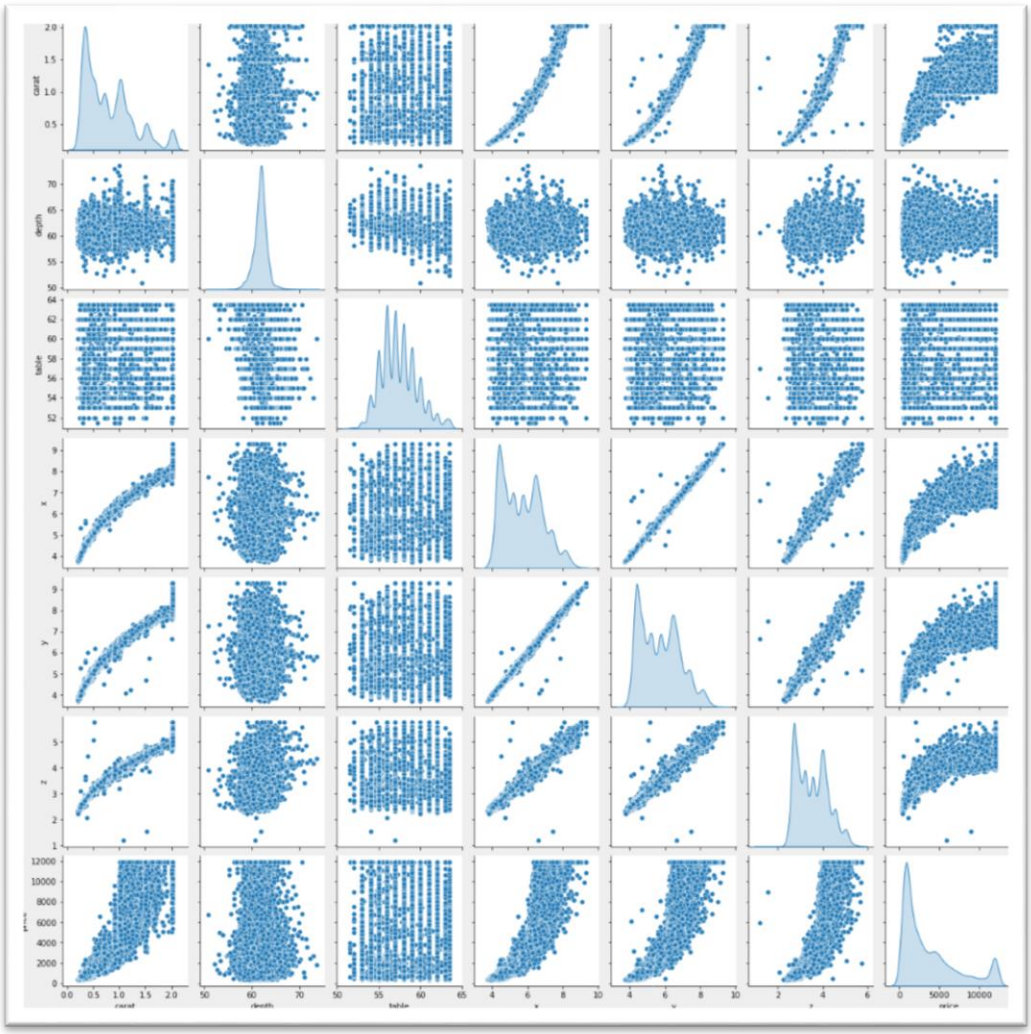
To measure the skeweness of every attribute:

carat	0.917214
depth	-0.025042
table	0.480476
x	0.397696
y	0.394060
z	0.394819
price	1.157121
dtype:	float64

Table 1.22 Measure of Skewness

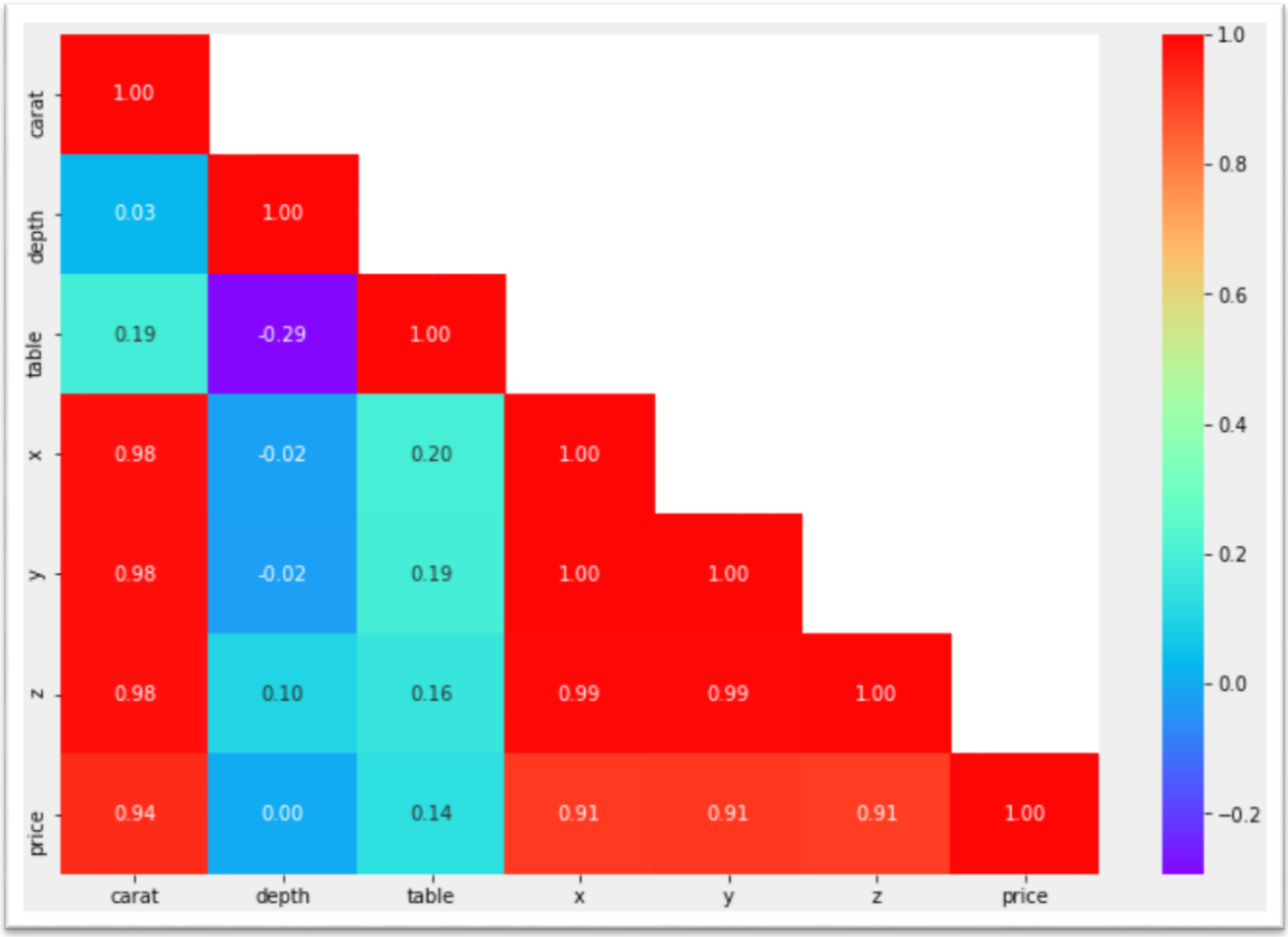
There is significant amount of outlier present in some variable.
We can see that the distribution of some quantitative features like "carat" and the target feature "price" are heavily "right-skewed".

EDA-Step 5: Bivariate Analysis:



Graph 1.14 Bivariate Analysis

Heat Map:



Graph 1.15 Heat Map

How each feature affects the price of diamonds:

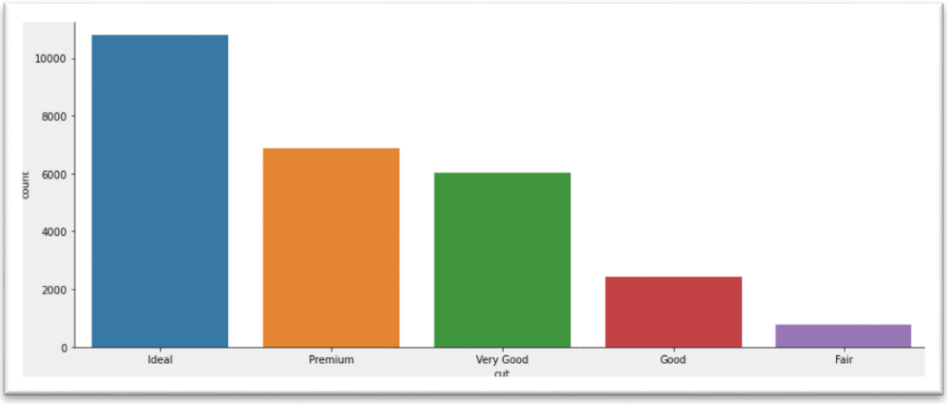
```
price    1.000000
carat    0.936765
y        0.914838
x        0.913409
z        0.908599
table    0.137915
depth    0.000313
Name: price, dtype: float64
```

Table 1.23 How each feature affects the price of diamonds

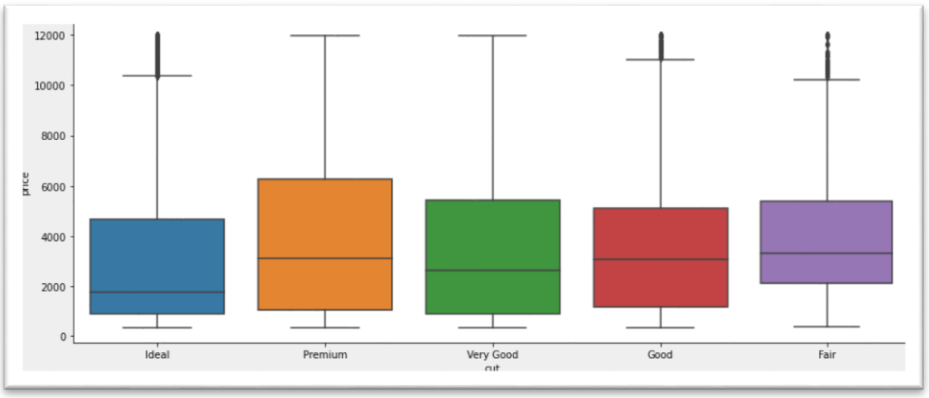
It can be inferred that most features correlate with the price of Diamond. The notable exception is "depth" which has a negligible correlation (<1%).

EDA for Categorical variable:

- **EDA for categorical columns 'CUT':**



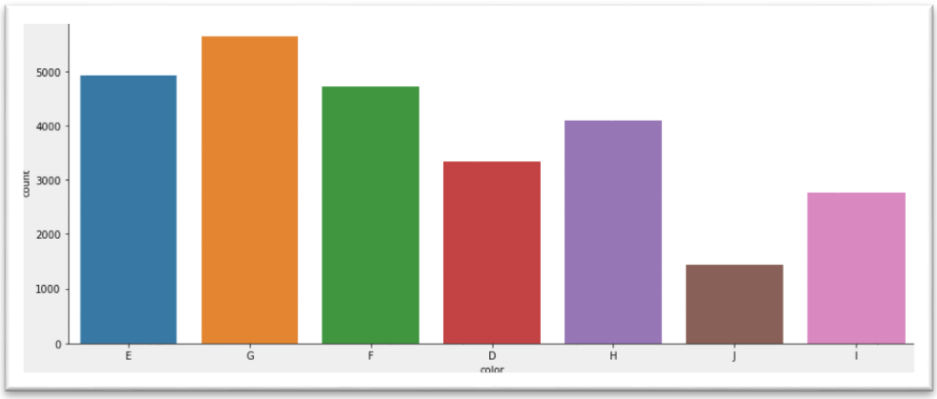
Graph 1.16 Count Plot



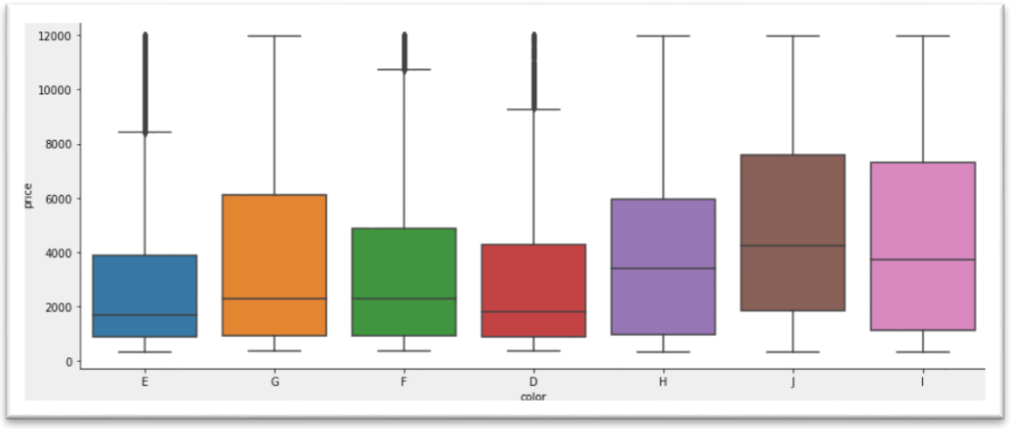
Graph 1.17 Box Plot

The Premium Cut on Diamonds are the most Expensive, followed by Very Good Cut.

- **EDA for categorical columns 'Color':**

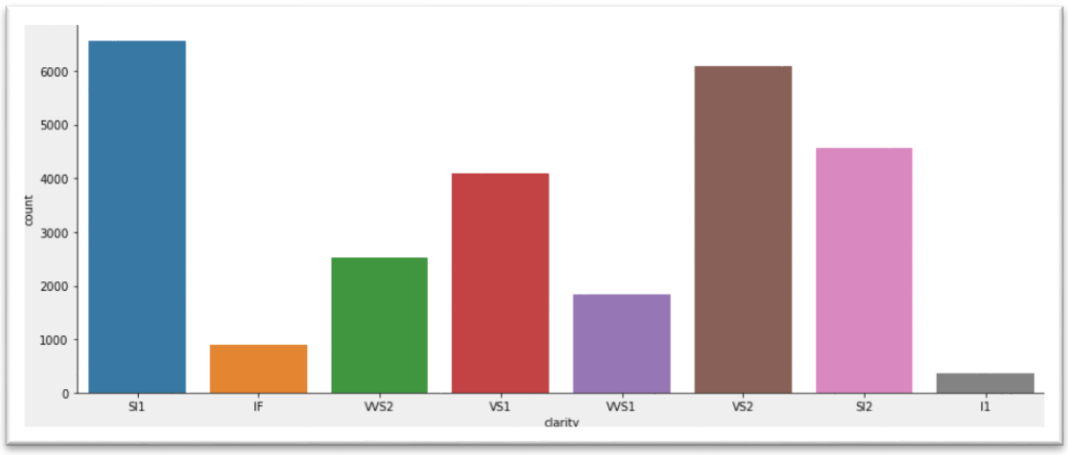


Graph 1.18 Count Plot

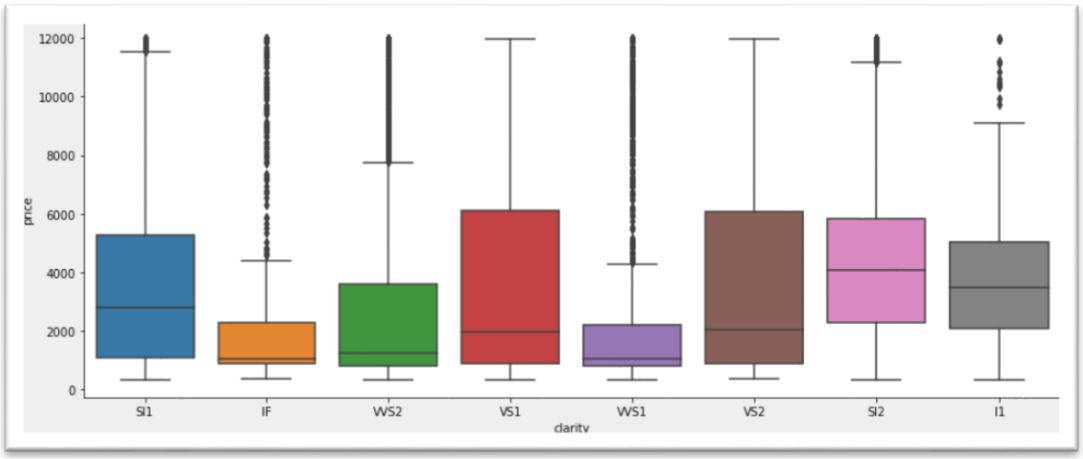


Graph 1.19 Box Plot

- **EDA for categorical columns 'Clarity':**



Graph 1.20 Count Plot



Graph 1.21 Box Plot

The Diamonds clarity with VS1 & VS2 are the most Expensive.

The inferences drawn from the above Exploratory Data analysis:

Observation-1:

- 'Price' is the target variable while all others are the predictors.
- The data set contains 26967 row, 11 column.
- In the given data set there are 2 Integer type features,6 Float type features. 3 Object type features. Where 'price' is the target variable and all other are predector variable.
- The first column is an index ("Unnamed: 0") as this only serial no, we can remove it.

Observation-2:

- On the given data set the mean and median values does not have much difference.
- We can observe Min value of "x", "y", "z" are zero this indicates that they are faulty values. As we know dimensionless or 2-dimensional diamonds are not possible. So, we have filter out those as it clearly faulty data entries.
- There are three object data type 'cut', 'color' and 'clarity'.

Observation-3:

- We can observe there are 697 missing values in the depth column. There are some duplicate row present. (33 duplicate rows out of 26958). which is nearly 0.12 % of the total data. So, on this case we have dropped the duplicated row.

Observation-4:

- There are significant amount of outlier present in some variable, the features with datapoint that are far from the rest of dataset which will affect the outcome of our regression model. So, we have treat the outlier. We can see that the distribution of some quantitative features like "carat" and the target feature "price" are heavily "right-skewed".

Observation-5:

- It looks like most features do correlate with the price of Diamond. The notable exception is "depth" which has a negligible correlation (~1%). Observation on 'CUT': The Premium Cut on Diamonds are the most Expensive, followed by Very Good Cut.

1.2 Impute null values if present, also check for the values which are equal to zero. Do they have any meaning or do we need to change them or drop them? Check for the possibility of combining the sub levels of a ordinal variables and take actions accordingly. Explain why you are combining these sub levels with appropriate reasoning.

As we have checked, there are some missing values:

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

Table 1.24 Null Values

carat	0.70
depth	61.80
table	57.00
x	5.69
y	5.70
z	3.52
price	2373.00
dtype: float64	

Table 1.25 Median Scores

Imputing missing values:

After replacing the missing values with median values, then we do not find any missing values.

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

Table 1.26 after Imputing Missing values

Checking for the values which are equal to zero. In Qs.1.1 we have already checked for 'Zero' value and we can observe there are some amounts of 'Zero' value present on the data set on variable 'x', 'y','z'.
This indicates that they are faulty values.
As we know dimensionless or 2-dimensional diamonds are not possible. So, we have filter out those as it clearly faulty data entries.

After removing 'zero value' from data set the data shape became as follows:

```
lr_df.shape
(26925, 10)
```

Table 1.27 Shape

Scaling is necessary as:

Scaling or standardizing the features around the centre and 0 with a standard deviation of 1 is important when we compare measurements that have different units. Variables that are measured at different scales do not contribute equally to the analysis and might end up creating a bias.

For example, A variable that ranges between 0 and 1000 will outweigh a variable that ranges between 0 and 1. Using these variables without standardization will give the variable with the larger range weight of 1000 in the analysis. Transforming the data to comparable scales can prevent this problem. In this data set we can see the all the variable are in different scale i.e price are in 1000s unit and depth and table are in 100s unit, and carat is in 10s.

So, it’s necessary to scale or standardise the data to allow each variable to be compared on a common scale. With data measured in different "units" or on different scales (as here with different means and variances) this is an important data processing step if the results are to be meaningful or not dominated

by the variables that have large variances.

But is scaling necessary in this case?

No, it is not necessary, we'll get an equivalent solution whether we apply some kind of linear scaling or not. But recommended for regression techniques as well because it would help gradient descent to converge fast and reach the global minima. When number of features becomes large, it helps is running model quickly else the starting point would be very far from minima, if the scaling is not done in pre-processing.
For now, we will process the model without scaling and later we will check the output with scaled data of regression model output.

1.3 Encode the data (having string values) for Modelling. Split the data into train and test (70:30). Apply Linear regression using scikit learn. Perform checks for significant variables using appropriate method from statsmodel. Create multiple models and check the performance of Predictions on Train and Test sets using Rsquare, RMSE & Adj Rsquare. Compare these models and select the best one with appropriate reasoning.

Encode the data (having string values):

Getting unique counts of all Objects:

```
cut
 Ideal        10805
Premium       6880
Very Good     6027
Good          2434
Fair           779
Name: cut, dtype: int64


color
 G    5650
E    4916
F    4722
H    4091
D    3341
I    2765
J    1440
Name: color, dtype: int64


clarity
 SI1     6564
VS2     6092
SI2     4561
VS1     4086
VVS2    2530
VVS1    1839
IF       891
I1       362
Name: clarity, dtype: int64
```

Table 1.28 Getting unique counts of all Objects

Converting objects to categorical codes:

```
carat      float64
cut         object
color       object
clarity     object
depth      float64
table      float64
x          float64
y          float64
z          float64
price      float64
dtype: object
```

Table 1.29 Data Type

Although, 'cut', 'color', 'clarity' column still showing as 'Object'.

After converting all of them in float type:

```
carat      float64
cut        float64
color      float64
clarity    float64
depth      float64
table      float64
x          float64
y          float64
z          float64
price      float64
dtype: object
```

Table 1.30 After Conversion

	carat	cut	color	clarity	depth	table	x	y	z	price
0	0.30	4.0	5.0	2.0	62.1	58.0	4.27	4.29	2.66	499.0
1	0.33	3.0	3.0	7.0	60.8	58.0	4.42	4.46	2.70	984.0
2	0.90	2.0	5.0	5.0	62.2	60.0	6.04	6.12	3.78	6289.0
3	0.42	4.0	4.0	4.0	61.6	56.0	4.82	4.80	2.96	1082.0
4	0.31	4.0	4.0	6.0	60.4	59.0	4.35	4.43	2.65	779.0
5	1.02	4.0	6.0	3.0	61.5	56.0	6.46	6.49	3.99	9502.0
6	1.01	1.0	2.0	2.0	63.7	60.0	6.35	6.30	4.03	4836.0
7	0.50	3.0	5.0	2.0	61.5	62.0	5.09	5.06	3.12	1415.0
8	1.21	1.0	2.0	2.0	63.8	63.5	6.72	6.63	4.26	5407.0
9	0.35	4.0	4.0	3.0	60.5	57.0	4.52	4.60	2.76	706.0

Table 1.31 Data set

Train-Test Split:

```
# Copy all the predictor variables into X dataframe
X = lr_df.drop('price', axis=1)

# Copy target into the y dataframe. This is the dependent variable
y = lr_df[['price']]
```

	carat	cut	color	clarity	depth	table	x	y	z
0	0.30	4.0	5.0	2.0	62.1	58.0	4.27	4.29	2.66
1	0.33	3.0	3.0	7.0	60.8	58.0	4.42	4.46	2.70
2	0.90	2.0	5.0	5.0	62.2	60.0	6.04	6.12	3.78
3	0.42	4.0	4.0	4.0	61.6	56.0	4.82	4.80	2.96
4	0.31	4.0	4.0	6.0	60.4	59.0	4.35	4.43	2.65

Table 1.32 Data set

Let us break the X and y dataframes into training set and test set. For this we will use Sklearn package's data splitting function which is based on random function:

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

Split X and y into training and test set in 70:30 ratio:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30 , random_state=1)
```

Invoke the Linear Regression function and find the best fit model on training data:

```
regression_model = LinearRegression()
regression_model.fit(X_train, y_train)
```

```
LinearRegression()
```

Let us explore the coefficients for each of the independent attributes:

The coefficient for carat is 8901.941225070897
The coefficient for cut is 109.18812485149378
The coefficient for color is 272.92132964490355
The coefficient for clarity is 436.441104215491
The coefficient for depth is 8.236971791615655
The coefficient for table is -17.3451703843684
The coefficient for x is -1417.908930444948
The coefficient for y is 1464.8272701468081
The coefficient for z is -711.225032681409

Execution using papermill encountered an exception here and stopped:

Observation-1:

Y=mx +c (m= m1,m2,m3...m9) here 9 diferent co-efficients will learn aling with the intercept which is "c" from the model.

From the above coefficients for each of the independent attributes we can conclude that the one unit increase in carat increases price by 8901.941. The one unit increase in cut increases price by 109.188. The one unit increase in color increases price by 272.921. The one unit increase in clarity increases price by 436.441. The one unit increase in y increases price by 1464.827. The one unit increase in depth increases price by 8.236.

But the one unit increase in table decreases price by -17.345. The one unit increase in x decreases price by -1417.908. The one unit increase in z decreases price by -711.225.

Let us check the intercept for the model:

```
The intercept for our model is -3171.950447307768
```

Table 1.33 intercept for the model

The intercept (often labelled the constant) is the expected mean value of Y when all X=0. If X never equals 0, then the intercept has no intrinsic meaning.

The intercept for our model is -3171.950447307667. In preset case when the other predictor variable are zero i.e like carat,cut, color, clarity all are zero then the C=-3172. (Y = m1X1 + m2X2+ ….. + mnXn + C + e) that means price is -3172. which is meaningless. We can do Z score or scaling the data and make it nearly zero.

R square on training data:

```
0.9311935886926559
```

R square on testing data:

```
0.931543712584074
```

R-square is the percentage of the response variable variation that is explained by a linear model. Or:
R-square = Explained variation / Total variation
R-squared is always between 0 and 100%: 0% indicates that the model explains none of the variability of the response data around its mean.100% indicates that the model explains all the variability of the response data around its mean. In this regression model we can see the R-square value on Training and Test data respectively 0.9311935886926559 and 0.931543712584074.

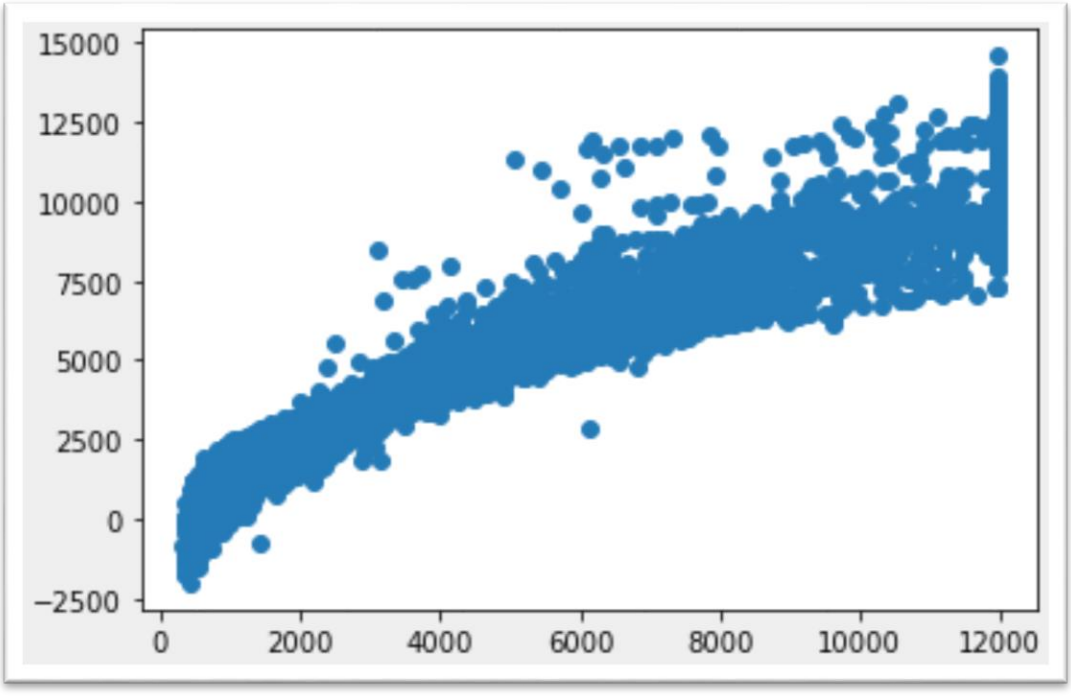
RMSE on Training data:

```
907.1312415459146
```

RMSE on Testing data:

```
911.8447345328439
```

Since this is regression, plot the predicted y value vs actual y values for the test data
A good model's prediction will be close to actual leading to high R and R2 values.



Graph 1.22 Scatter Plot

We can see that the is a linear plot, very strong corelation between the predicted y and actual y. But there are lots of spread. That indicated some kind noise present on the data set i.e Unexplained variances on the output.

Linear regression Performance Metrics:

Intercept for the model: -3171.950447307667 R square on training data: 0.9311935886926559 R square on testing data: 0.931543712584074 RMSE on Training data: 907.1312415459143 RMSE on Testing data: 911.8447345328436 As the training data & testing data score are almost inline, we can conclude this model is a Right-Fit Model.

Applying zscore statsmodels:

```
from scipy.stats import zscore

X_train_scaled  = X_train.apply(zscore)
X_test_scaled = X_test.apply(zscore)
y_train_scaled = y_train.apply(zscore)
y_test_scaled = y_test.apply(zscore)
```

Invoking the Linear Regression function and find the best fit model on training data:

```
regression_model = LinearRegression()
regression_model.fit(X_train_scaled, y_train_scaled)
```

```
LinearRegression()
```

Let us explore the coefficients for each of the independent attributes:

The coefficient for carat is 1.1837737061779434
The coefficient for cut is 0.0351250006552973
The coefficient for color is 0.1344926928764153
The coefficient for clarity is 0.2080977932562188
The coefficient for depth is 0.0033262937188390197
The coefficient for table is -0.010815851633643304
The coefficient for x is -0.4596898424125263
The coefficient for y is 0.4716627091792404
The coefficient for z is -0.14249737973827165

```
The intercept for our model is -5.879615251304727e-16
```

Table 1.34 intercept for the model

```
# Model score - R2 or coeff of determinant
# R^2=1-RSS / TSS

regression_model.score(X_test_scaled, y_test_scaled)
```

```
0.9315051288558229
```

Table 1.35 Regression Model Score

Now we can observe by applying z score the intercept became -5.87961525130473e-16. Earlier it was -3171.950447307667. the co-efficient has changed, the bias became nearly zero but the overall accuracy still same.

Check Multi-collinearity using VIF:

carat ---> 121.96543302739589
cut ---> 10.388738909800345
color ---> 5.546407587131625
clarity ---> 5.455999699082339
depth ---> 1218.3824913329145
table ---> 878.3985698779234
x ---> 10744.05623520385
y ---> 9482.053091580401
z ---> 3697.5688286012546

We can observe there are very strong multi collinearity present in the data set. Ideally it should be within 1 to 5. We are exploring the Linear Regression using statsmodels as we are interested in some more statistical metrics of the model.

Linear Regression using statsmodels:

Concatenate X and y into a single data frame:

	carat	cut	color	clarity	depth	table	x	y	z	price
5030	1.10	1.0	5.0	1.0	63.3	56.0	6.53	6.58	4.15	4065.0
12108	1.01	2.0	6.0	1.0	64.0	56.0	6.30	6.38	4.06	5166.0
20181	0.67	1.0	1.0	3.0	60.7	61.4	5.60	5.64	3.41	1708.0
4712	0.76	1.0	3.0	2.0	57.7	63.0	6.05	5.97	3.47	2447.0
2548	1.01	3.0	3.0	4.0	62.8	59.0	6.37	6.34	3.99	6618.0

Table 1.36 Modified Data Set

Fitting the data set:

Intercept	-3171.950447
carat	8901.941225
cut	109.188125
color	272.921330
clarity	436.441104
depth	8.236972
table	-17.345170
x	-1417.908930
y	1464.827270
z	-711.225033

dtype: float64

Table 1.37 Fitting the data set

Inferential statistics:

OLS Regression Results

Dep. Variable:	price	R-squared:	0.931
Model:	OLS	Adj. R-squared:	0.931
Method:	Least Squares	F-statistic:	2.833e+04
Date:	Mon, 24 Jan 2022	Prob (F-statistic):	0.00
Time:	14:21:27	Log-Likelihood:	-1.5510e+05
No. Observations:	18847	AIC:	3.102e+05
Df Residuals:	18837	BIC:	3.103e+05
Df Model:	9		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-3171.9504	787.532	-4.028	0.000	-4715.583	-1628.318
carat	8901.9412	82.792	107.521	0.000	8739.661	9064.222
cut	109.1881	7.268	15.024	0.000	94.943	123.433
color	272.9213	4.105	66.478	0.000	264.874	280.968
clarity	436.4411	4.473	97.581	0.000	427.674	445.208
depth	8.2370	10.876	0.757	0.449	-13.080	29.554
table	-17.3452	3.904	-4.443	0.000	-24.998	-9.693
x	-1417.9089	136.590	-10.381	0.000	-1685.637	-1150.181
y	1464.8273	136.068	10.765	0.000	1198.122	1731.533
z	-711.2250	156.187	-4.554	0.000	-1017.366	-405.084

Omnibus:	2652.028	Durbin-Watson:	2.005
Prob(Omnibus):	0.000	Jarque-Bera (JB):	9642.429
Skew:	0.687	Prob(JB):	0.00
Kurtosis:	6.223	Cond. No.	1.03e+04

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.03e+04. This might indicate that there are
strong multicollinearity or other numerical problems.

Table 1.38 Inferential statistics

Assuming null hypothesis is true, i.e there is no relationship between this variable with price. from that universe we have drawn the sample and, on this sample, we have found this co-efficient for the variable shown above.

Now we can ask what is the probability of finding this co-efficient in this drawn sample if in the real world the co-efficient is zero. As we see here the overall P value is less than alpha, so rejecting H0 and accepting Ha that at least 1 regression co-efficient is not '0'. Here all regression co-efficient are not '0'.

For an example: we can see the p value is showing 0.449 for 'depth' variable, which is much higher than 0.05. That means this dimension is useless. So, we can say that the attribute which are having p value greater than 0.05 are poor predictor for price.

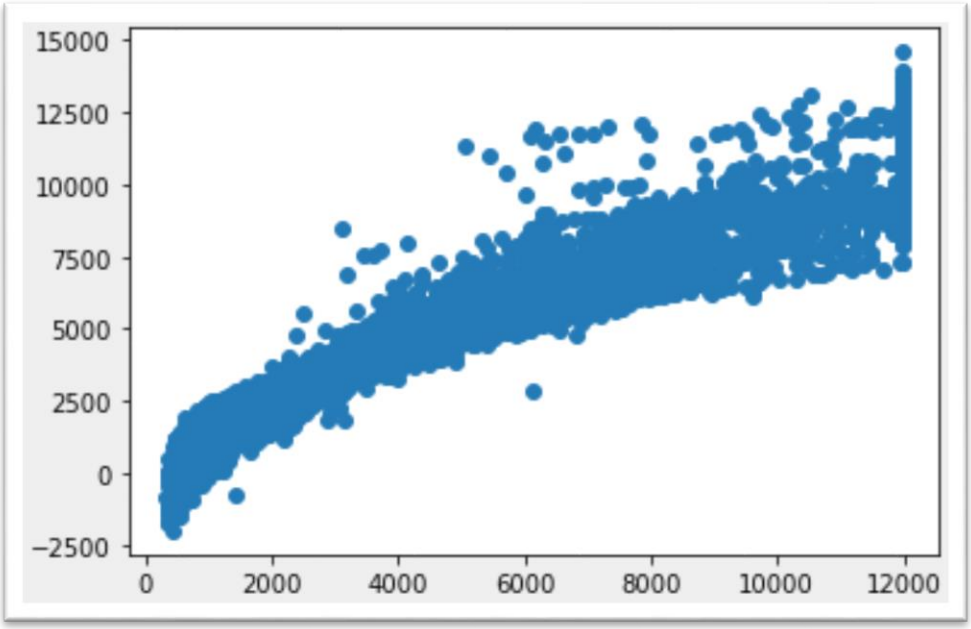
```
# Calculate MSE
mse = np.mean((lm1.predict(data_train.drop('price',axis=1))-data_train['price'])**2)

# Calculate MSE
mse1 = np.mean((lm1.predict(data_test.drop('price',axis=1))-data_test['price'])**2)
```

Root Mean Squared Error – RMSE:

```
907.1312415459133
```

Root Mean Squared Error – RMSE:

```
911.8447345328426
```



Graph 1.23 Scatter Plot

```
for i,j in np.array(lm1.params.reset_index()):
    print('({}) * {} +'.format(round(j,2),i),end=' ')
```

```
(-3171.95) * Intercept + (8901.94) * carat + (109.19) * cut + (272.92) * color + (436.44) * clarity + (8.24) * depth + (-17.35)
* table + (-1417.91) * x + (1464.83) * y + (-711.23) * z +
```

The final Linear Regression equation is:
price = b0 + b1 *carat[T.True] + b2 * cut + b3 * color + b4 * clarity+ b5 * depth + b6 * table + b7 * x + b8 * y + b9 *z True
price = (-3171.95) * Intercept + (8901.94) * carat + (109.19) * cut + (272.92) * color + (436.44) * clarity + (8.24) * depth + (-17.35) * table + (-1417.91)) * x + (1464.83) * y + (-711.23) * z _True

When carat increases by 1 unit, diamond price increases by 8901.94 units, keeping all other predictors constant. When cut increases by 1 unit, diamond price increases by 109.19 units, keeping all other predictors constant. When color increases by 1 unit, diamond price increases by 272.92 units, keeping all other predictors constant. When clarity increases by 1 unit, diamond price increases by 436.44 units, keeping all other predictors constant. When y increases by 1 unit, diamond price increases by 1464.83 units, keeping all other predictors constant. As per model these five attributes that are most important attributes 'Carat', 'Cut', 'color', 'clarity' and width i.e 'y' for predicting the price.

There are also some negative co-efficient values, for instance, corresponding co-efficient (-1417.91) for 'x',(-711.23) for z and (-17.35) for table This implies, these are inversely proportional with diamond price.

On the given data set we can see the 'X' i.e Length of the cubic zirconia in mm. having negative co-efficient. And the p value is less than 0.05, so can conclude that as higher the length of the stone is a lower profitable stone.
Similarly, for the 'z' variable having negative co-efficient i.e -711.23. And the p value is less than 0.05, so we can conclude that as higher the 'z' of the stone is a lower profitable stone.
Also, we can see the 'y' width in mm having positive co-efficient. And the p value is less than 0.05, so we can conclude that higher the width of the stone is a higher profitable stone.
Finally, we can conclude that best 5 attributes that are most important are 'Carat', 'Cut', 'color', 'clarity' and width i.e 'y' for predicting the price.

1.4 Inference: Basis on these predictions, what are the business insights and recommendations.

Inference:

We can see that the from the linear plot, very strong corelation between the predicted y and actual y. But there are lots of spread. That indicates some kind noise present on the data set i.e Unexplained variances on the output.

Linear regression Performance Metrics:

intercept for the model: -3171.950447307667 R square on training data: 0.9311935886926559 R square on testing data: 0.931543712584074 RMSE on Training data: 907.1312415459143 RMSE on Testing data: 911.8447345328436

As the training data & testing data score are almost inline, we can conclude this model is a Right-Fit Model.

Impact of scaling:

Now we can observe by applying z score the intercept became -5.87961525130473e-16. Earlier it was -3171.950447307667. the co-efficient has changed, the bias became nearly zero but the overall accuracy still same.

Multi collinearity:

We can observe there are very strong multi collinearity present in the data set.

From statsmodels:

We can see R-squared:0.931 and Adj. R-squared: 0.931 are same. The overall P value is less than alpha.

- Finally, we can conclude that Best 5 attributes that are most important are 'Carat', 'Cut', 'color', 'clarity' and width i.e 'y' for predicting the price.
- When 'carat' increases by 1 unit, diamond price increases by 8901.94 units, keeping all other predictors constant.
- When 'cut' increases by 1 unit, diamond price increases by 109.19 units, keeping all other predictors constant.
- When 'color' increases by 1 unit, diamond price increases by 272.92 units, keeping all other predictors constant.
- When 'clarity' increases by 1 unit, diamond price increases by 436.44 units, keeping all other predictors constant.
- When 'y' increases by 1 unit, diamond price increases by 1464.83 units, keeping all other predictors constant.
- we can see the p value is showing 0.449 for depth variable, which is much greater than 0.05. That means this attribute is useless.

- There are also some negative co-efficient values, we can see the 'X' i.e Length of the cubic zirconia in mm. having negative co-efficient -1417.9089. And the p value is less than 0.05, so can conclude that as higher the length of the stone is a lower profitable stone.

- Similarly, for the 'z' variable having negative co-efficient i.e -711.23. And the p value is less than 0.05, so we can conclude that as higher the 'z' of the stone is a lower profitable stone.

Recommendations:

- The Gem Stones company should consider the features 'Carat', 'Cut', 'color', 'clarity' and width i.e., 'y' as most important for predicting the price.
- To distinguish between higher profitable stones and lower profitable stones so as to have better profit share.
- As we can see from the model Higher the width('y') of the stone is higher the price.
- So, the stones having higher width('y') should consider in higher profitable stones. The 'Premium Cut' on Diamonds are the most Expensive, followed by 'Very Good' Cut, these should consider in higher profitable stones.
- The Diamonds clarity with 'VS1' &'VS2' are the most Expensive. So, these two categories also consider in higher profitable stones.
- As we see for 'X' i.e., Length of the stone, higher the length of the stone is lower the price.
- So higher the Length('x') of the stone are lower is the profitability.
- Higher the 'z' i.e., Height of the stone is, lower the price. This is because if a Diamond's Height is too large Diamond will become 'Dark' in appearance because it will no longer return an Attractive amount of light. That is why Stones with higher 'z' is also are lower in profitability.

Problem 2:

Executive Summary:

The given data set has number columns and rows in them. The given data set is provided details of 872 employees of a company. Here we need to identify and help the company whether an employee will opt for the package or not on the basis of the information given in the data set. It will also help the company in analyzing particular employees to sell their packages.

Background of the problem:

You are hired by a tour and travel agency which deals in selling holiday packages. You are provided details of 872 employees of a company. Among these employees, some opted for the package and some didn't. You have to help the company in predicting whether an employee will opt for the package or not on the basis of the information given in the data set. Also, find out the important factors on the basis of which the company will focus on particular employees to sell their packages.

Data Dictionary:

Variable Name	Description
Holiday_Package	Opted for Holiday Package yes/no?
Salary	Employee salary
age	Age in years
edu	Years of formal education
no_young_children	The number of young children (younger than 7 years)
no_older_children	Number of older children
foreign	foreigner Yes/No

Data Description:

The shape of the data set seems to be with 872 rows and 8 columns.

```
df.shape
(872, 8)
```

Table 2.1 Shape

- No null values in the dataset.
- The Data set is having object and Integer Data types.
- There are no duplicate values present in the data set.

- Also, the entire data set does not have any null or missing values.

```
Unnamed: 0           0
Holliday_Package     0
Salary               0
age                  0
educ                 0
no_young_children    0
no_older_children    0
foreign              0
dtype: int64
```

Table 2.2 Null Values

```
Number of duplicate rows = 0
```

Table 2.3 Count of Duplicate Rows

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 872 entries, 0 to 871
Data columns (total 8 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Unnamed: 0         872 non-null    int64
 1   Holliday_Package   872 non-null    object
 2   Salary             872 non-null    int64
 3   age                872 non-null    int64
 4   educ               872 non-null    int64
 5   no_young_children  872 non-null    int64
 6   no_older_children  872 non-null    int64
 7   foreign            872 non-null    object
dtypes: int64(6), object(2)
memory usage: 54.6+ KB
```

Table 2.4 Data Information

Sample Data Set:

	Unnamed: 0	Holliday_Package	Salary	age	educ	no_young_children	no_older_children	foreign
0	1	no	48412	30	8	1	1	no
1	2	yes	37207	45	8	0	1	no
2	3	no	58022	46	9	0	0	no
3	4	no	66503	31	11	2	0	no
4	5	no	66734	44	12	0	2	no

Table 2.5 Data set

The above table represents the actual representation of the data set.

2.1 Data Ingestion: Read the dataset. Do the descriptive statistics and do null value condition check, write an inference on it. Perform Univariate and Bivariate Analysis. Do exploratory data analysis.

General Data Set:

	Unnamed: 0	Holliday_Package	Salary	age	educ	no_young_children	no_older_children	foreign
0	1	no	48412	30	8	1	1	no
1	2	yes	37207	45	8	0	1	no
2	3	no	58022	46	9	0	0	no
3	4	no	66503	31	11	2	0	no
4	5	no	66734	44	12	0	2	no

Table 2.7 Data set

	Unnamed: 0	Holliday_Package	Salary	age	educ	no_young_children	no_older_children	foreign
867	868	no	40030	24	4	2	1	yes
868	869	yes	32137	48	8	0	0	yes
869	870	no	25178	24	6	2	0	yes
870	871	yes	55958	41	10	0	1	yes
871	872	no	74659	51	10	0	0	yes

Table 2.8 Data set

Null Information:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 872 entries, 0 to 871
Data columns (total 8 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Unnamed: 0         872 non-null    int64
 1   Holliday_Package   872 non-null    object
 2   Salary             872 non-null    int64
 3   age                872 non-null    int64
 4   educ               872 non-null    int64
 5   no_young_children  872 non-null    int64
 6   no_older_children  872 non-null    int64
 7   foreign            872 non-null    object
dtypes: int64(6), object(2)
memory usage: 54.6+ KB
```

Table 2.9 Null Values

Shape:

```
df.shape
```

```
(872, 8)
```

Table 2.10 Shape

Descriptive Analysis:

	Unnamed: 0	Salary	age	educ	no_young_children	no_older_children
count	872.000000	872.000000	872.000000	872.000000	872.000000	872.000000
mean	436.500000	47729.172018	39.955275	9.307339	0.311927	0.982798
std	251.869014	23418.668531	10.551675	3.036259	0.612870	1.086786
min	1.000000	1322.000000	20.000000	1.000000	0.000000	0.000000
25%	218.750000	35324.000000	32.000000	8.000000	0.000000	0.000000
50%	436.500000	41903.500000	39.000000	9.000000	0.000000	1.000000
75%	654.250000	53469.500000	48.000000	12.000000	0.000000	2.000000
max	872.000000	236961.000000	62.000000	21.000000	3.000000	6.000000

Table 2.11 Descriptive Analysis

We have integer and continuous data, Holiday package is our target variable salary, age, educ and number young children, number older children of employee have the went to foreign, these are the attributes we have to cross examine and help the company predict weather the person will opt for holiday package or not.

Check for Missing values:

```
Unnamed: 0          0
Holliday_Package    0
Salary              0
age                 0
educ                0
no_young_children   0
no_older_children   0
foreign             0
dtype: int64
```

Table 2.12 Missing values

Checking for Duplicates:

```
dups = df.duplicated()
print('Number of duplicate rows = %d' % (dups.sum()))
#df[dups]
```

```
Number of duplicate rows = 0
```

Table 2.13 Checking for Duplicates

Unique values in the categorical data:

```
HOLLIDAY_PACKAGE :  2
yes    401
no     471
Name: Holliday_Package, dtype: int64


FOREIGN :  2
yes    216
no     656
Name: foreign, dtype: int64
```

Table 2.14 Unique values in the categorical data

Percentage of target:

```
<bound method IndexOpsMixin.value_counts of 0       no
1      yes
2       no
3       no
4       no
      ...
867     no
868    yes
869     no
870    yes
871     no
Name: Holliday_Package, Length: 872, dtype: object>
```

Table 2.15 Percentage of target

```
no     0.540138
yes    0.459862
Name: Holliday_Package, dtype: float64
```

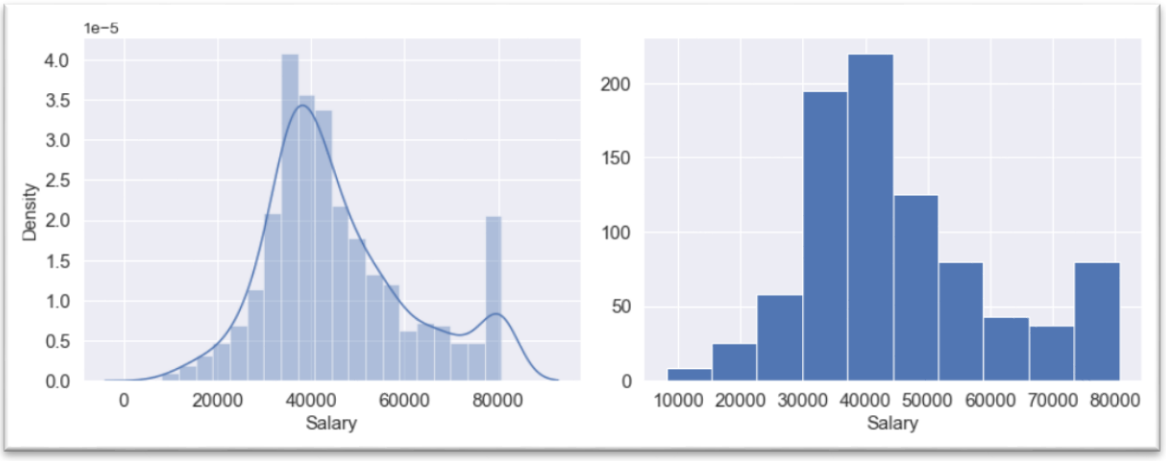
Table 2.16 Percentage of target

This split indicates that 45% of employees are interested in the holiday package.

Univariate Analysis:

1) Salary Variable:

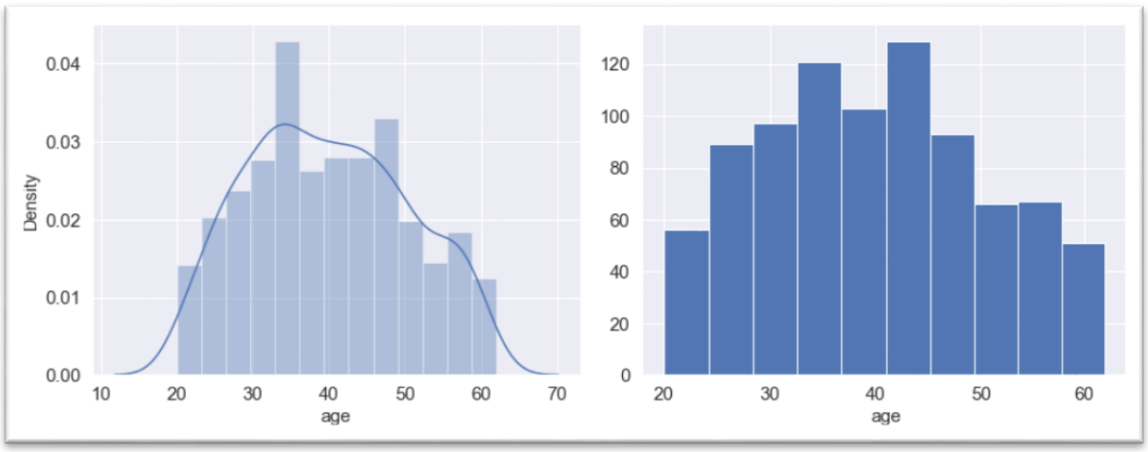
Range of values: 72582.0
Minimum Age: 8105.75
Maximum Age: 80687.75
Mean value: 45608.33686926605
Median value: 41903.5
Standard deviation: 15699.745150717654
Null values: False



Graph 2.1 Dist-Plot || Histogram

2) Age Variable:

Range of values: 42.0
Minimum Duration: 20.0
Maximum Duration: 62.0
Mean value: 39.955275229357795
Median value: 39.0
Standard deviation: 10.551674590487607
Null values: False

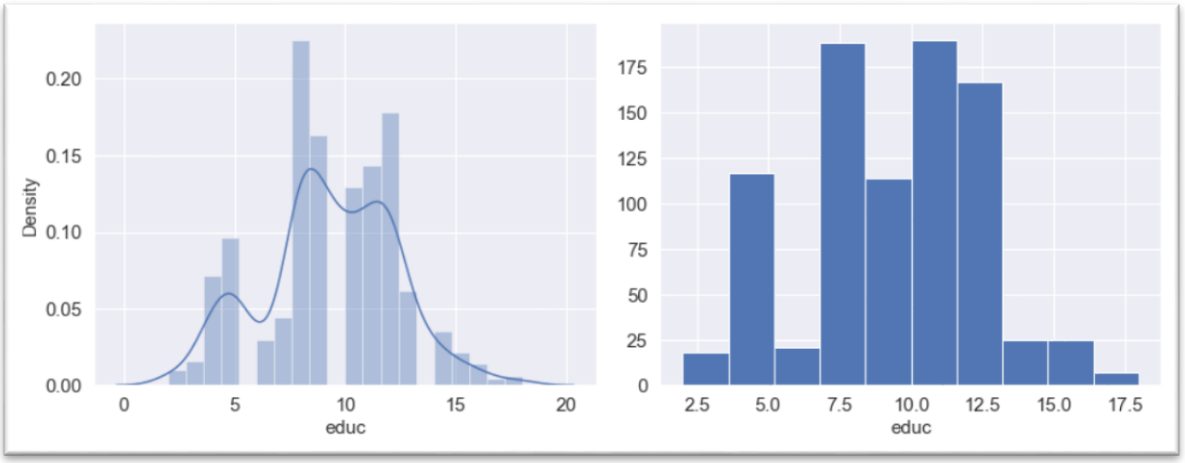


Graph 2.2 Dist-Plot || Histogram

3) Education Variable:

Range of values: 16.0
Minimum Sales: 2.0
Maximum Sales: 18.0
Mean value: 9.302752293577981

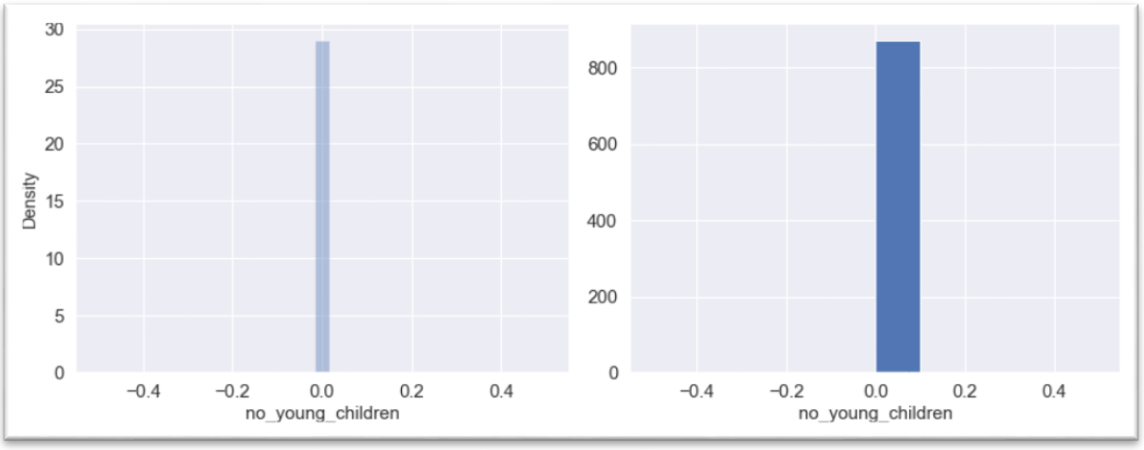
Median value: 9.0
Standard deviation: 3.0147119925149255
Null values: False



Graph 2.3 Dist-Plot || Histogram

4) No Young Children Variable:

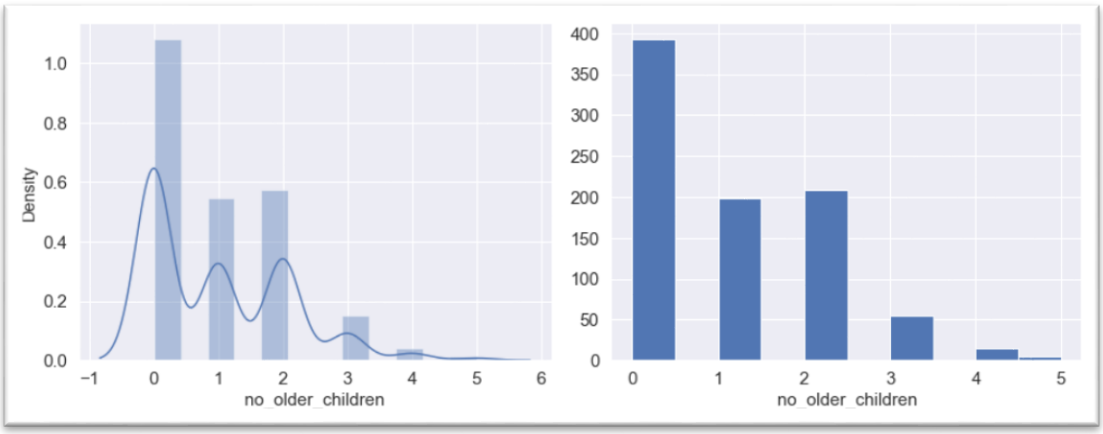
Range of values: 0.0
Minimum Sales: 0.0
Maximum Sales: 0.0
Mean value: 0.0
Median value: 0.0
Standard deviation: 0.0
Null values: False



Graph 2.4 Dist-Plot || Histogram

5) No Older Children Variable:

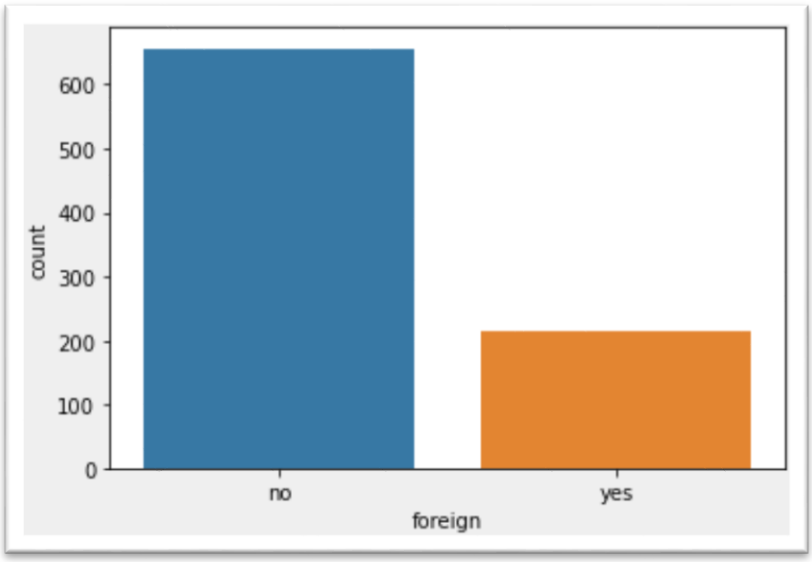
Range of values: 5.0
Minimum Sales: 0.0
Maximum Sales: 5.0
Mean value: 0.9805045871559633
Median value: 1.0
Standard deviation: 1.0771974192861413
Null values: False



Graph 2.5 Dist-Plot || Histogram

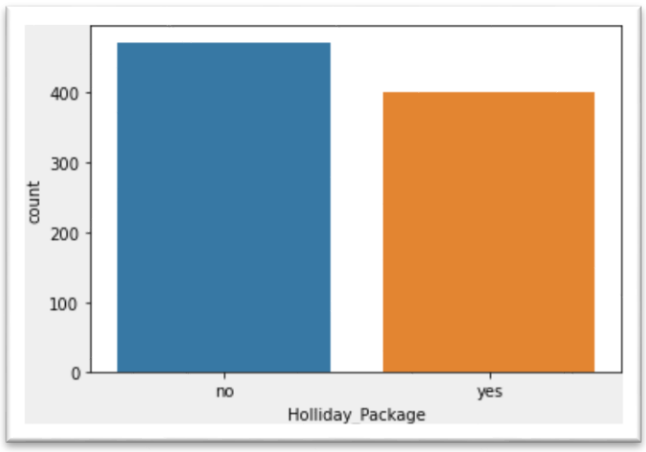
CATEGORICAL UNIVARIATE ANALYSIS:

1) FOREIGN:



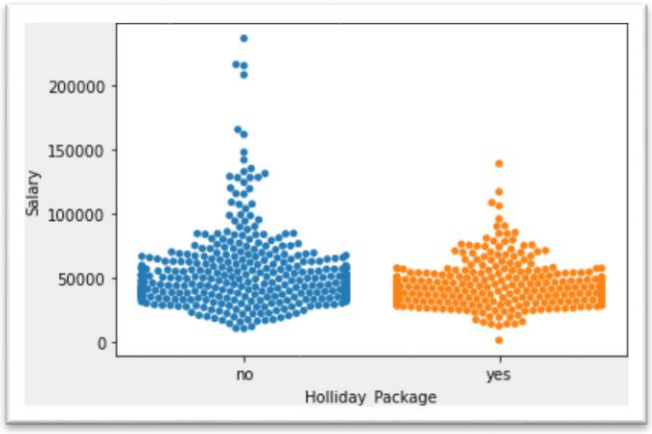
Graph 2.6 Count Plot

2) HOLIDAY PACKAGE:



Graph 2.7 Count Plot

3) HOLIDAY PACKAGE VS SALARY:



Graph 2.8 Swarm Plot

We can see employee below salary 150000 have always opted for holiday package

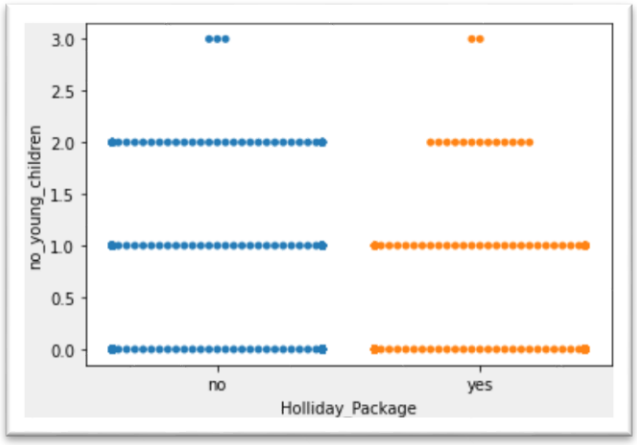
4) HOLIDAY PACKAGE VS AGE:

Graph 2.9 Swarm Plot

5) HOLIDAY PACKAGE VS EDUCATION:

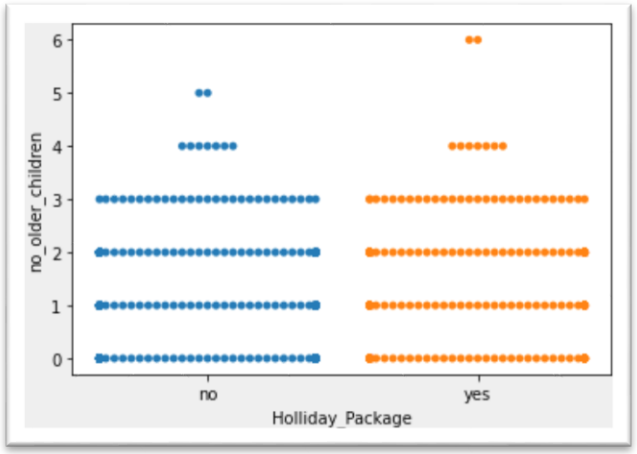
Graph 2.10 Swarm Plot

6) HOLIDAY PACKAGE VS NO YOUNG CHILDREN:



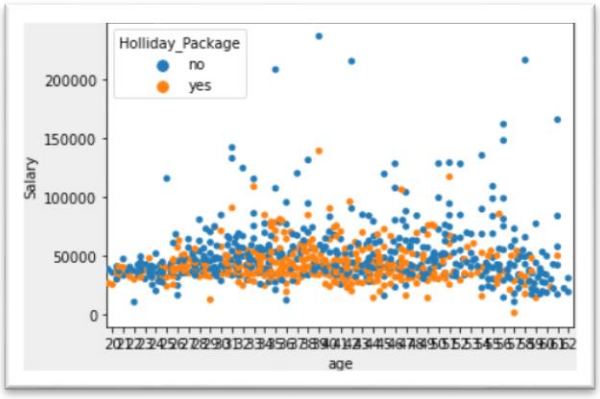
Graph 2.11 Swarm Plot

7) HOLIDAY PACKAGE VS OLDER CHILDREN:



Graph 2.12 Swarm Plot

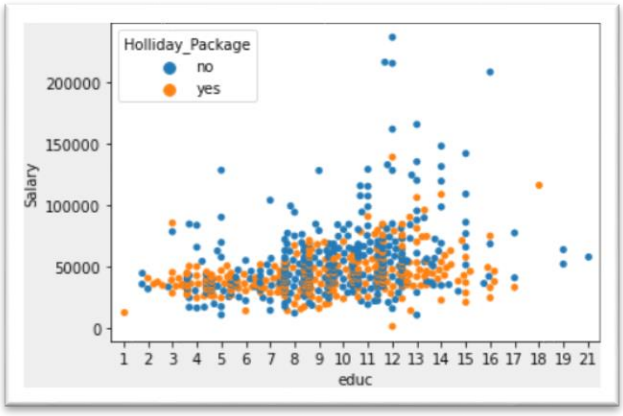
8) AGE VS SALARY VS HOLIDAY PACKAGE:



Graph 2.13 Swarm Plot

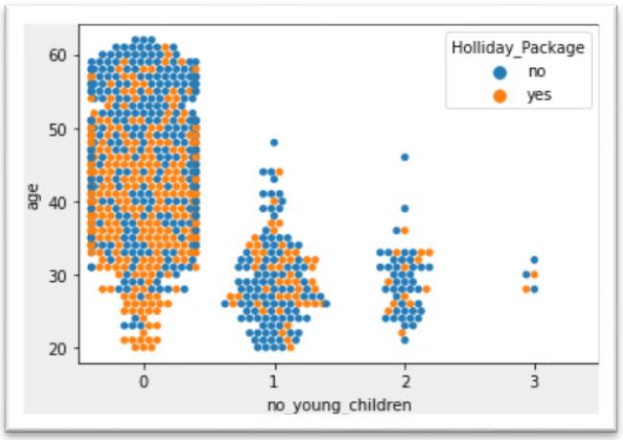
Employee age over 50 to 60 have seems to be not taking the holiday package, whereas in the age 30 to 50 and salary less than 50000 people have opted more for holiday package.

9) EDUC VS SALARY VS HOLIDAY PACKAGE:



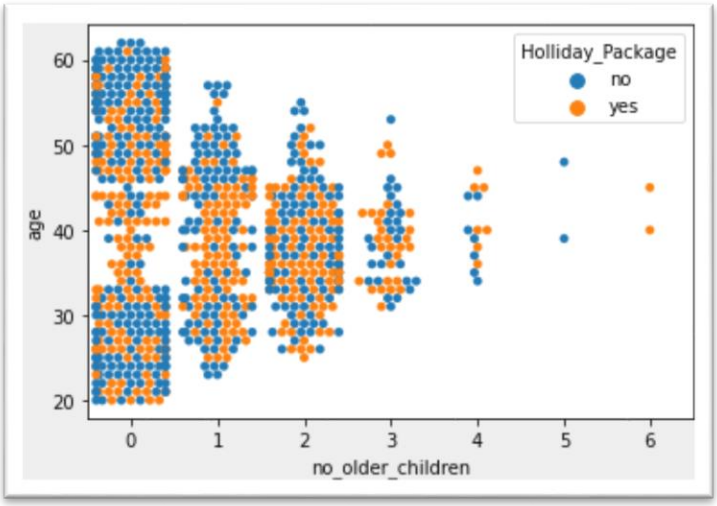
Graph 2.14 Swarm Plot

10) NO YOUNG CHILDREN VS AGE VS HOLIDAY PACKAGE:



Graph 2.15 Swarm Plot

11) NO OLDER CHILDREN VS AGE VS HOLIDAY_PACKAGE:



Graph 2.16 Swarm Plot

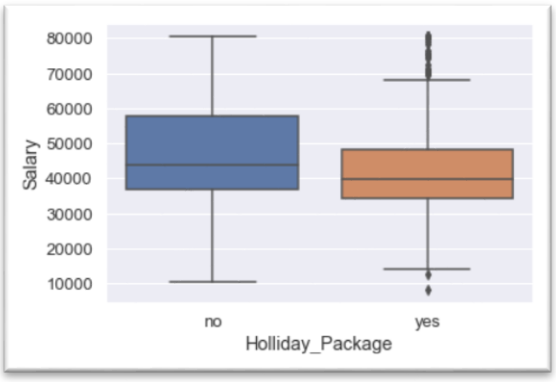
BIVARITE ANALYSIS:
DATA DISTRIBUTION:

Graph 2.17 BIVARITE ANALYSIS

There is no correlation between the data, the data seems to be normal. There is no huge difference in the data distribution among the holiday package, I don't see any clear two different distribution in the data.

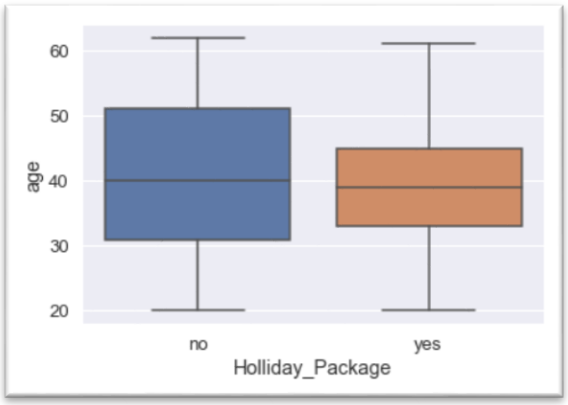
Bi-Variate Analysis with Target variable:

Target Variable with Salary



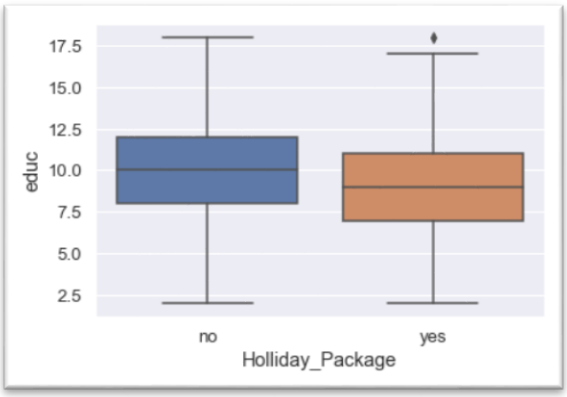
Graph 2.18 Box Plot

Target Variable with age



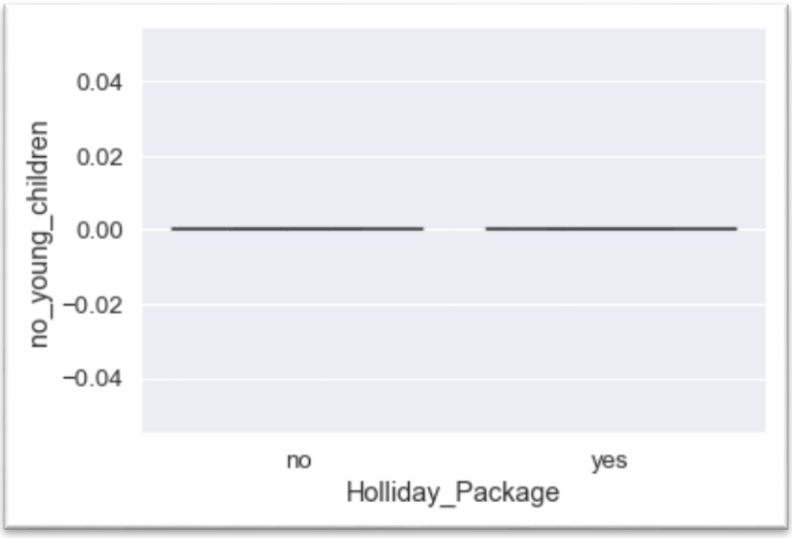
Graph 2.19 Box Plot

Target Variable with education



Graph 2.20 Box Plot

Target Variable with No Young Children

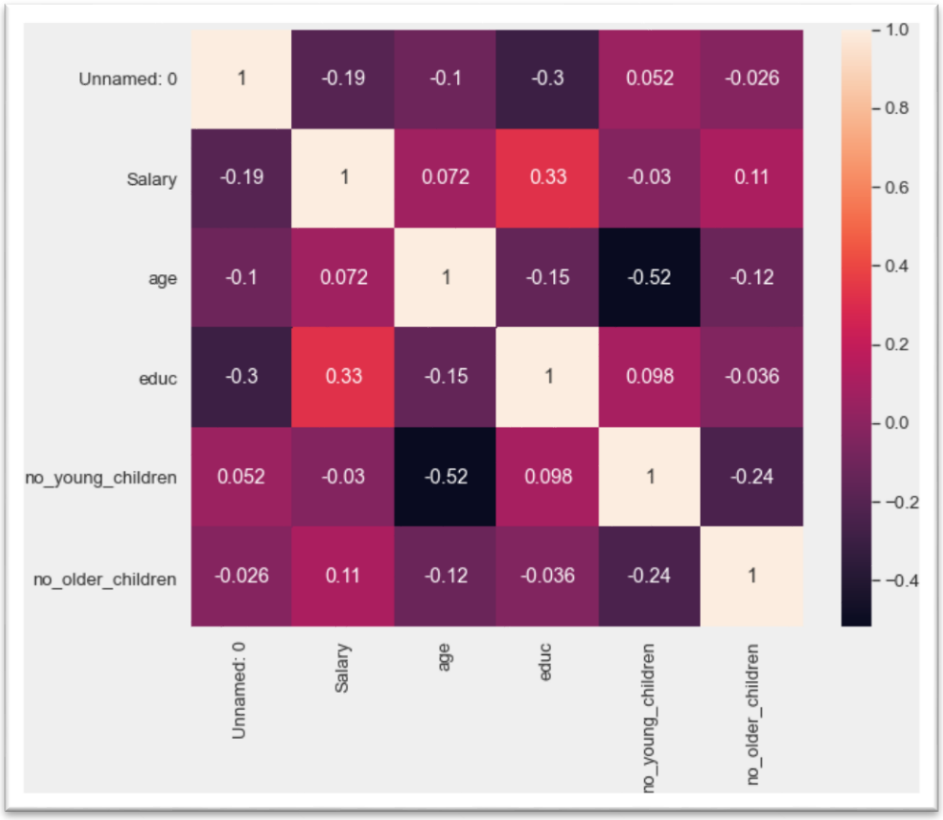


Graph 2.21 Box Plot

Target Variable with No Older Children

Graph 2.22 Box Plot

Checking for Correlations:



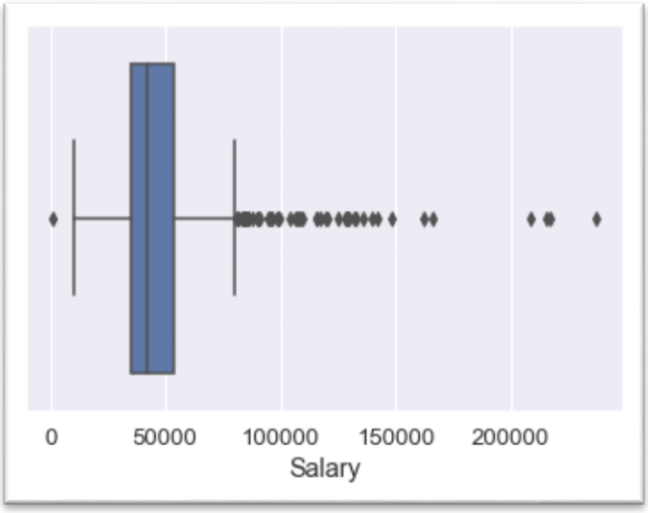
Graph 2.23 Heat Map

No multi collinearity in the dat

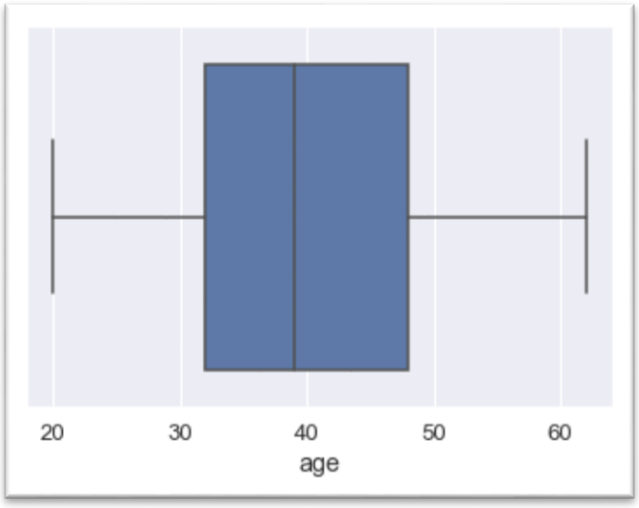
Checking for Outliers:

BEFORE OUTLIER TREATMENT

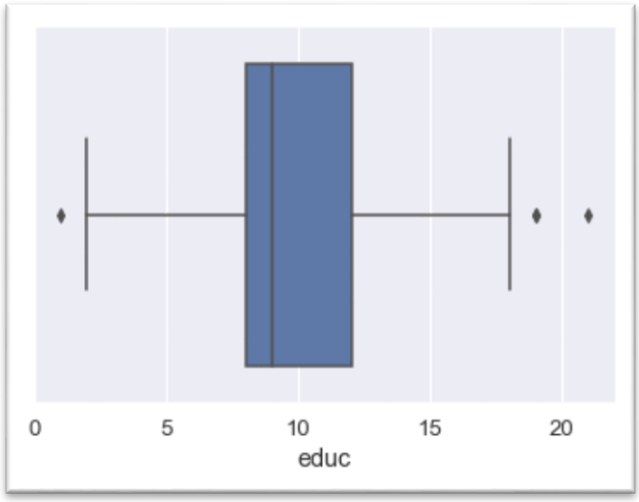
we have outliers in the dataset, as LDA works based on numerical computation treating outliers will help perform the model better



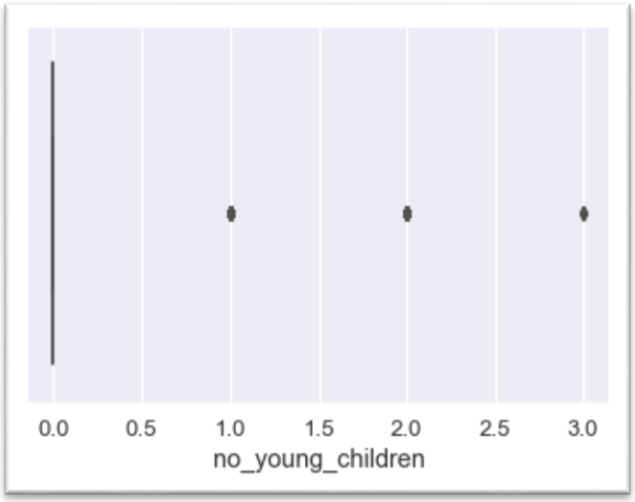
Graph 2.24 Box Plot



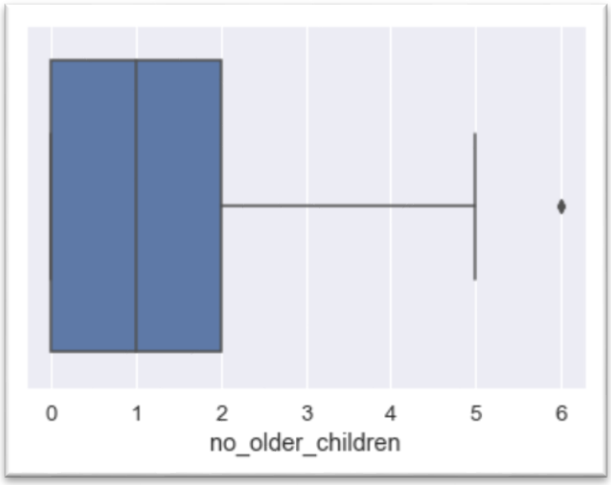
Graph 2.25 Box Plot



Graph 2.26 Box Plot



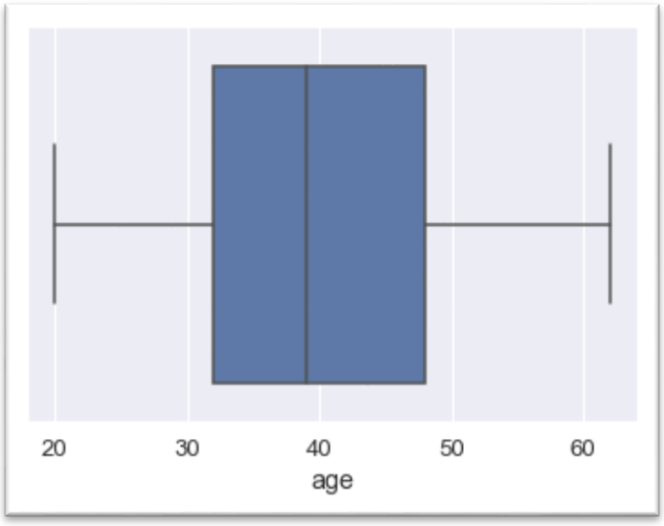
Graph 2.27 Box Plot



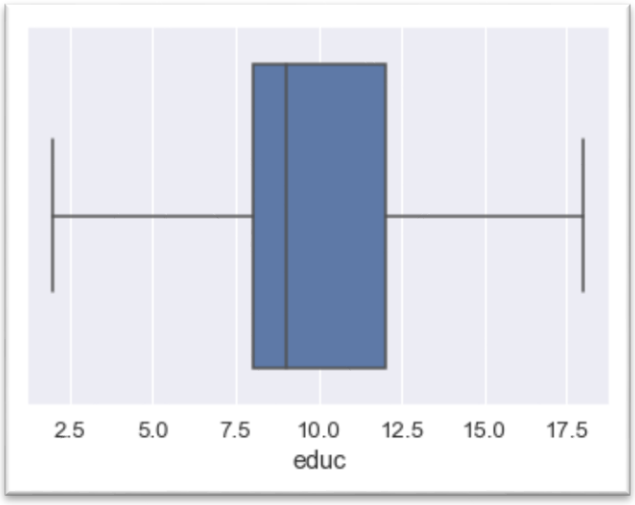
Graph 2.28 Box Plot

AFTER OUTLIER TREATMENT:

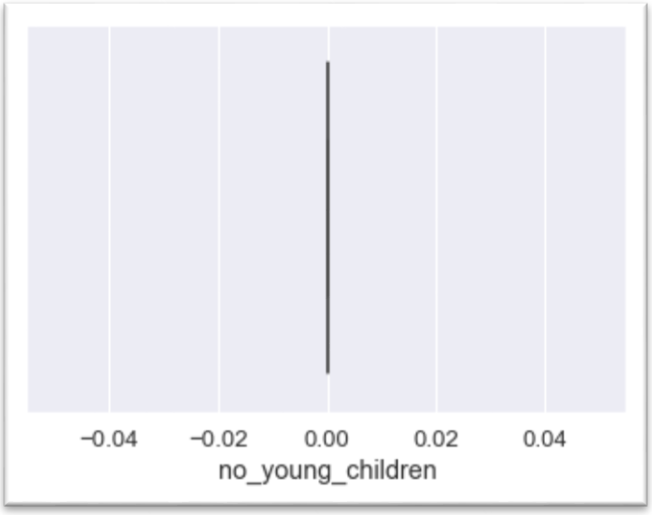
Graph 2.29 Box Plot



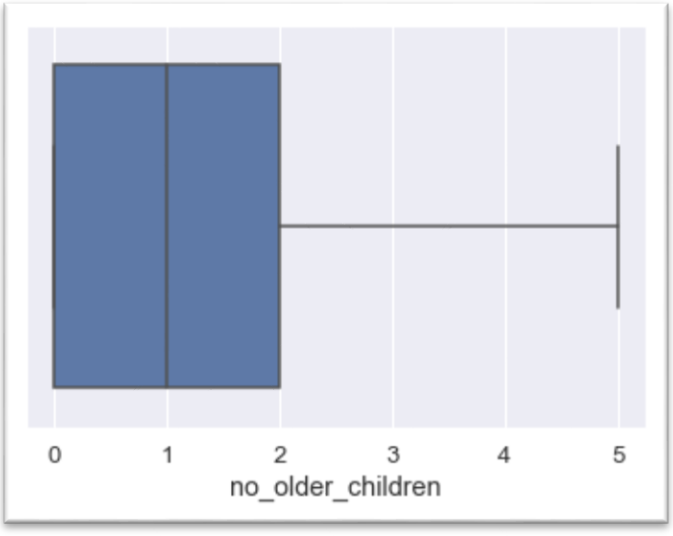
Graph 2.30 Box Plot



Graph 2.31 Box Plot



Graph 2.32 Box Plot



Graph 2.33 Box Plot

2.2 Do not scale the data. Encode the data (having string values) for Modelling. Data Split: Split the data into train and test (70:30). Apply Logistic Regression and LDA (linear discriminant analysis).

ENCODING CATEGORICAL VARIABLE:

```
df=pd.get_dummies(df, columns=['Holliday_Package', 'foreign'], drop_first =True)
```

```
df.drop('Unnamed: 0', axis='columns', inplace=True)
```

```
df.head()
```

	Salary	age	educ	no_young_children	no_older_children	Holliday_Package_yes	foreign_yes
0	48412.0	30.0	8.0	0.0	1.0	0	0
1	37207.0	45.0	8.0	0.0	1.0	1	0
2	58022.0	46.0	9.0	0.0	0.0	0	0
3	66503.0	31.0	11.0	0.0	0.0	0	0
4	66734.0	44.0	12.0	0.0	2.0	0	0

Table 2.17 Modified data set

Train / Test split:

```
# Copy all the predictor variables into X dataframe:

X = df.drop('Holliday_Package_yes', axis=1)

# Copy Target into the Y dataframe:

Y = df['Holliday_Package_yes']
```

```
# Split X and Y into Training and Test Set into 70:30 Ratio:

X_train, X_test, Y_train, Y_test = model_selection.train_test_split(X, Y, test_size=0.30, random_state=1, stratify = Y)
```

```
print('Number of rows and columns of the training set for the independent variables:',X_train.shape)
print('Number of rows and columns of the training set for the dependent variable:',Y_train.shape)
print('Number of rows and columns of the test set for the independent variables:',X_test.shape)
print('Number of rows and columns of the test set for the dependent variable:',Y_test.shape)
```

```
Number of rows and columns of the training set for the independent variables: (610, 6)
Number of rows and columns of the training set for the dependent variable: (610,)
Number of rows and columns of the test set for the independent variables: (262, 6)
Number of rows and columns of the test set for the dependent variable: (262,)
```

Table 2.18 Test / Train Split

Train Value Count:

```
0    0.539344
1    0.460656
Name: Holliday_Package_yes, dtype: float64
```

Table 2.19 Train Value Count

Test Value Count:

```
0    0.541985
1    0.458015
Name: Holliday_Package_yes, dtype: float64
```

Table 2.20 Test Value Count

Logistic Regression Model:

```
# Fit the Logistic Regression model
model = LogisticRegression(solver='newton-cg',max_iter=10000,penalty='none',verbose=True,n_jobs=2)
model.fit(X_train, Y_train)
```

```
[Parallel(n_jobs=2)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=2)]: Done   1 out of   1 | elapsed:    0.8s finished
```

```
LogisticRegression(max_iter=10000, n_jobs=2, penalty='none', solver='newton-cg',
                   verbose=True)
```

Predicting on Training and Test dataset:

```
Ytrain_predict = model.predict(X_train)
Ytest_predict = model.predict(X_test)
```

Getting the Predicted Classes and Probs:

	0	1
0	0.640766	0.359234
1	0.569940	0.430060
2	0.655265	0.344735
3	0.564177	0.435823
4	0.538867	0.461133

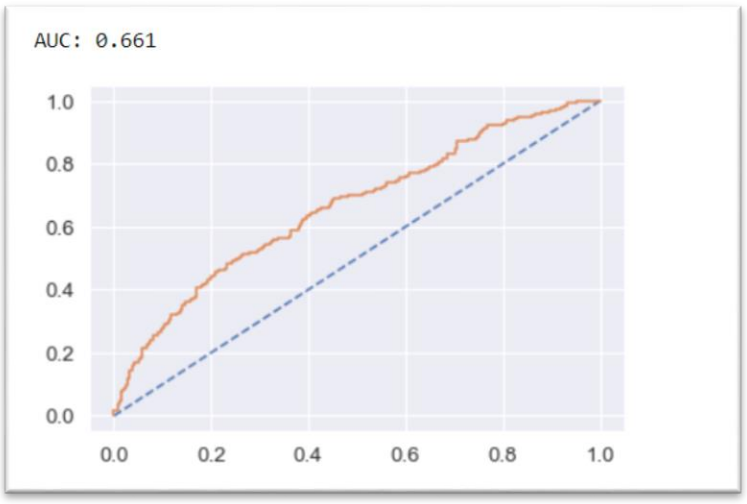
Table 2.21 Getting the Predicted Classes and Probs

Model Evaluation:

Accuracy - Training Data

```
0.6344262295081967
```

AUC and ROC for the Training data:



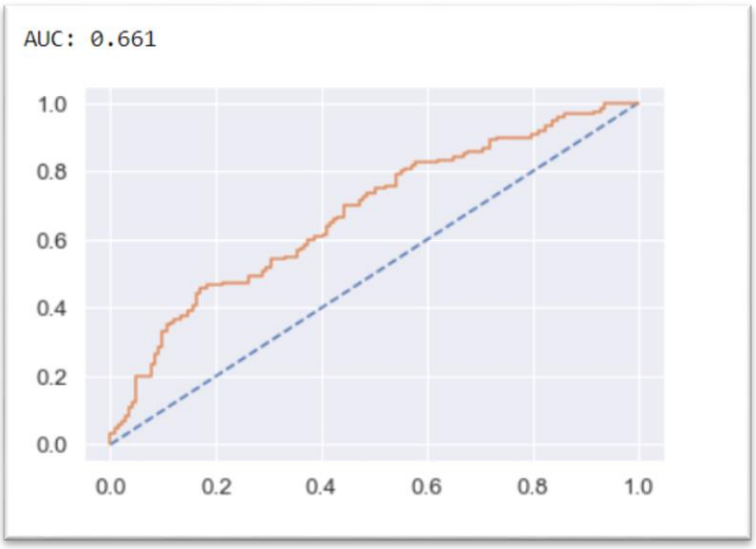
Graph 2.34 ROC Curve

Model Evaluation:

Accuracy - Test Data:

```
0.6603053435114504
```

AUC and ROC for the Test data:

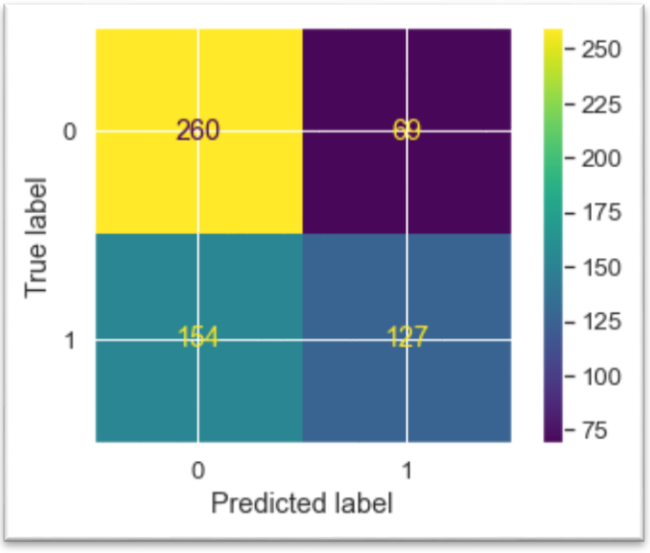


Graph 2.35 ROC Curve

Confusion Matrix for the training data:

```
array([[260,  69],
       [154, 127]], dtype=int64)
```

Table 2.22 Confusion Matrix for the training data



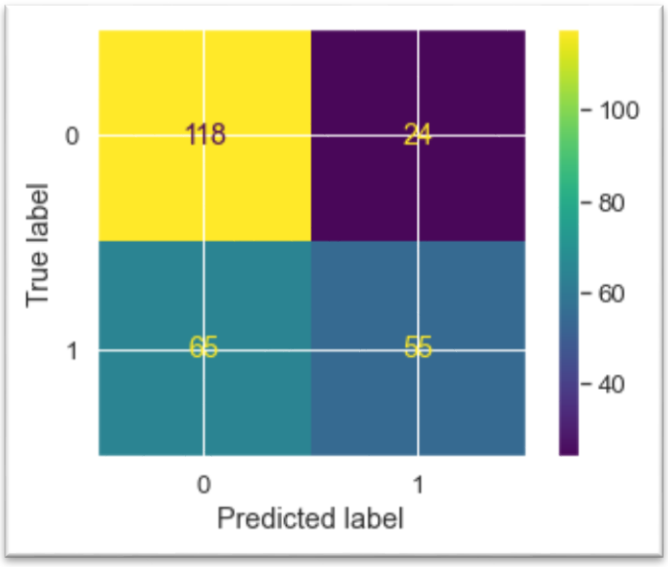
Graph 2.36 Confusion Matrix

Classification Report of the Train Data:

	precision	recall	f1-score	support
0	0.63	0.79	0.70	329
1	0.65	0.45	0.53	281
accuracy			0.63	610
macro avg	0.64	0.62	0.62	610
weighted avg	0.64	0.63	0.62	610

Table 2.23 Classification Report of the Train Data

Confusion Matrix for Test data:



Graph 2.37 Confusion Matrix

Classification Report of the Test Data:

	precision	recall	f1-score	support
0	0.64	0.83	0.73	142
1	0.70	0.46	0.55	120
accuracy			0.66	262
macro avg	0.67	0.64	0.64	262
weighted avg	0.67	0.66	0.65	262

Table 2.24 Classification Report of the Test Data

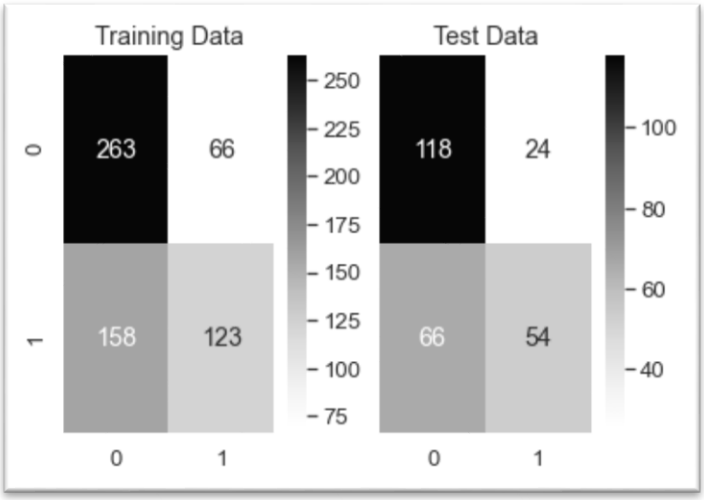
LDA:

```
#Build LDA Model
clf = LinearDiscriminantAnalysis()
model=clf.fit(X_train,Y_train)
```

Prediction:

```
# Training Data Class Prediction with a cut-off value of 0.5
pred_class_train = model.predict(X_train)

# Test Data Class Prediction with a cut-off value of 0.5
pred_class_test = model.predict(X_test)
```

Training Data and Test Data Confusion Matrix Comparison:



Graph 2.38 Confusion Matrix

Training Data and Test Data Classification Report Comparison:

Model Score:

```
0.6327868852459017
```

Classification Report of the Training Data:

Classification Report of the training data:

	precision	recall	f1-score	support
0	0.62	0.80	0.70	329
1	0.65	0.44	0.52	281
accuracy			0.63	610
macro avg	0.64	0.62	0.61	610
weighted avg	0.64	0.63	0.62	610

Table 2.25 Classification Report of the Training Data

Confusion Matrix Array:

```
array([[263,  66],
       [158, 123]], dtype=int64)
```

Table 2.26 Confusion Matrix Array

Model score:

```
0.6564885496183206
```

Classification Report of the Test data:

Classification Report of the test data:

	precision	recall	f1-score	support
0	0.64	0.83	0.72	142
1	0.69	0.45	0.55	120
accuracy			0.66	262
macro avg	0.67	0.64	0.63	262
weighted avg	0.66	0.66	0.64	262

Table 2.27 Classification Report of the Testing data

Confusion Matrix Array:

```
array([[118,  24],
       [ 66,  54]], dtype=int64)
```

Table 2.28 Confusion Matrix

Probability prediction for the training and test data:

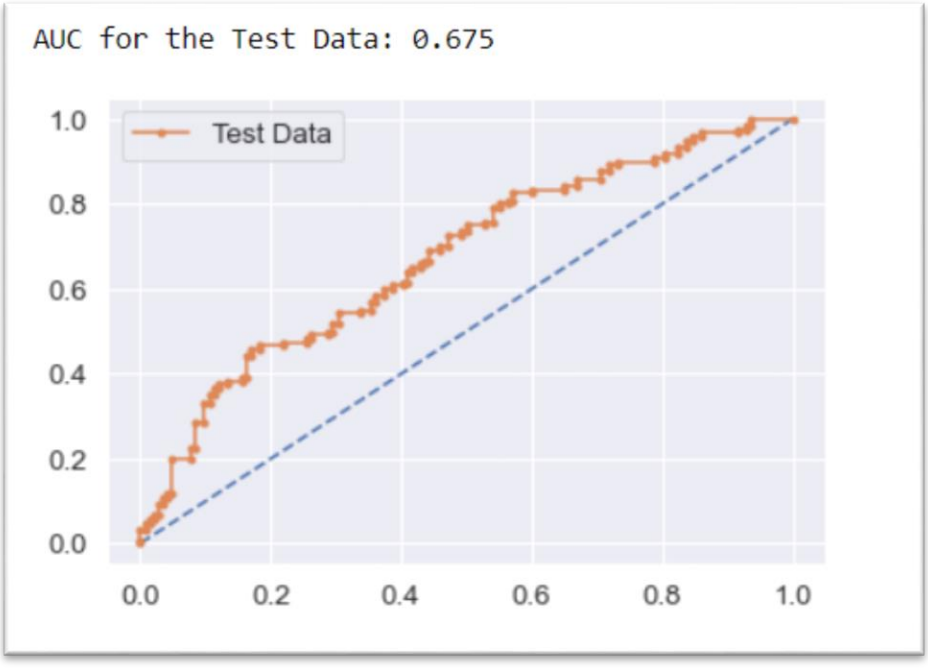
array([0.73614833, 0.64936462, 0.45296681, 0.67842845, 0.66903745,
0.3774365 , 0.32146187, 0.43700845, 0.3601507 , 0.62535644,
0.23288822, 0.26149271, 0.44813877, 0.48143931, 0.29614015,
0.46145671, 0.43653859, 0.30536315, 0.54175682, 0.66481517,
0.49405489, 0.31531235, 0.75146336, 0.43228357, 0.31741838,
0.74742033, 0.24732567, 0.7256214 , 0.49778025, 0.4701221 ,
0.2458476 , 0.59535195, 0.39263888, 0.38905706, 0.34870239,
0.34966751, 0.47524197, 0.50165433, 0.47132887, 0.20930823,
0.22990207, 0.67104116, 0.40365647, 0.62197077, 0.64959972,
0.34741408, 0.34082919, 0.73390632, 0.42023771, 0.62369484,
0.57834131, 0.3978419 , 0.49066256, 0.40405466, 0.22000295,
0.41866697, 0.25849518, 0.5276083 , 0.6674488 , 0.40702718,
0.59934243, 0.26473066, 0.43421298, 0.34478906, 0.5114521 ,
0.53275226, 0.41995444, 0.49835656, 0.40640466, 0.33834489,
0.25698034, 0.65937394, 0.48721781, 0.25848244, 0.71797697,
0.39324025, 0.48908617, 0.50504833, 0.62024606, 0.55260131,
0.39492894, 0.43666826, 0.73389137, 0.38397192, 0.53173819,
0.65168428, 0.25898704, 0.42233978, 0.65219145, 0.42349008,
0.48338736, 0.60326959, 0.64871589, 0.75257494, 0.46854155,
0.7247534 , 0.4887346 , 0.49487963, 0.40387276, 0.41974209,
0.32429701, 0.69297473, 0.45427337, 0.51206111, 0.27790423,
0.50621008, 0.38769227, 0.73285195, 0.42722683, 0.38489627,
0.49867156, 0.21576231, 0.61043892, 0.25256523, 0.28511341,
0.33782852, 0.3829038 , 0.628171 , 0.42103401, 0.37782328,
0.32283739, 0.53798713, 0.43993741, 0.44983844, 0.38167872,
0.72006318, 0.24448207, 0.65366929, 0.50765829, 0.39004421,
0.58895499, 0.4088867 , 0.69989527, 0.40408162, 0.48149176,
0.42592822, 0.4161605 , 0.37907762, 0.1903723 , 0.39261196,
0.63163585, 0.71253473, 0.20636562, 0.74473227, 0.56796441,
0.40596278, 0.71296378, 0.57889081, 0.39585726, 0.67683056,
0.69743975, 0.71847052, 0.37151856, 0.34900276, 0.41006687,
0.35078747, 0.25642249, 0.37594364, 0.36402388, 0.48613845,
0.33564296, 0.29376678, 0.30573437, 0.72693944, 0.4536416 ,
0.38351076, 0.40221331, 0.53889692, 0.61615712, 0.70944804,

0.29421506, 0.30965888, 0.6683032 , 0.29471755, 0.43194729,
0.33236522, 0.37719275, 0.37872247, 0.32619416, 0.31520047,
0.40906014, 0.46767398, 0.7130688 , 0.29033017, 0.33478761,
0.70818695, 0.34690974, 0.36624712, 0.45316002, 0.67481085,
0.4689055 , 0.41545483, 0.60809333, 0.42192183, 0.34993664,
0.46492822, 0.63632545, 0.47555264, 0.52354727, 0.69615225,
0.24253693, 0.32626628, 0.41133661, 0.50085677, 0.63897094,
0.43785406, 0.66126068, 0.40811347, 0.74593657, 0.44601261,
0.39529661, 0.32737773, 0.31576221, 0.47574222, 0.65974822,
0.41189414, 0.39370075, 0.75435881, 0.49761827, 0.48463423,
0.33963624, 0.39039361, 0.41006545, 0.56232731, 0.42670054,
0.51609922, 0.24627049, 0.35949776, 0.40303767, 0.52498975,
0.7152786 , 0.34587192, 0.37787533, 0.68679399, 0.64033296,
0.35246483, 0.78113576, 0.67705693, 0.63319045, 0.28481706,
0.48125928, 0.71780916, 0.74503993, 0.69803642, 0.44383889,
0.34011563, 0.44340732, 0.20762023, 0.48484675, 0.42088848,
0.72471829, 0.2852136 , 0.57083767, 0.67999258, 0.3535601 ,
0.29536363, 0.43903259, 0.21171125, 0.45557715, 0.65571748,
0.40860887, 0.65654681, 0.49975013, 0.3029154 , 0.34641904,
0.41659192, 0.71230821, 0.3329712 , 0.4042915 , 0.45711451,
0.32334985, 0.47049381, 0.41070286, 0.6518861 , 0.46893992,
0.22274536, 0.46292341, 0.69287809, 0.51153926, 0.40289234,
0.6681511 , 0.37684978, 0.42866869, 0.32642362, 0.4034879 ,
0.51941988, 0.18304187, 0.36095354, 0.43067053, 0.17577322,
0.34539909, 0.46507156, 0.47579528, 0.6227013 , 0.3917243 ,
0.30279221, 0.29866545, 0.44062281, 0.20846998, 0.38176569,
0.46467979, 0.62064754, 0.41578771, 0.41218533, 0.4447573 ,
0.75412635, 0.3419936 , 0.43927021, 0.46385829, 0.34427335,
0.22333193, 0.41456259, 0.66073237, 0.29033257, 0.49551066,
0.40037955, 0.38083495, 0.2707942 , 0.41304548, 0.19775092,
0.43492668, 0.49785114, 0.27285804, 0.71366079, 0.24661037,
0.33178519, 0.3274077 , 0.73648509, 0.34641534, 0.68354503,
0.41496689, 0.43944487, 0.67779654, 0.60912213, 0.33782802,
0.33237931, 0.26109173, 0.47550864, 0.73096287, 0.67111096,
0.49164312, 0.69645349, 0.38261438, 0.33353915, 0.45109865,
0.44677868, 0.43278217, 0.45556416, 0.65443219, 0.67456632,
0.48944305, 0.40527008, 0.59927665, 0.26887915, 0.34618671,
0.39935286, 0.34703486, 0.4300262 , 0.46139173, 0.43558268,
0.39848668, 0.21189169, 0.45253029, 0.33656244, 0.37324064,
0.48061764, 0.33731159, 0.18659602, 0.74551345, 0.37210888,
0.43840595, 0.52300254, 0.35472911, 0.39333863, 0.72409752,
0.23734063, 0.33328512, 0.70829408, 0.45297865, 0.66195128,
0.37695348, 0.58543049, 0.3149635 , 0.48542004, 0.14235478,

0.30205991, 0.22022375, 0.50147165, 0.34424459, 0.456065 ,
0.31159963, 0.32119905, 0.64082664, 0.51554573, 0.57858854,
0.68535818, 0.34219253, 0.43969694, 0.4706637 , 0.66804497,
0.47724094, 0.28356175, 0.31790078, 0.25832456, 0.28554891,
0.46387696, 0.38899031, 0.70193564, 0.4835478 , 0.47882565,
0.28799029, 0.40290915, 0.36343027, 0.45361943, 0.70971919,
0.40102281, 0.41199039, 0.54530414, 0.2365716 , 0.3738932 ,
0.71821055, 0.19403498, 0.76472471, 0.76971297, 0.64546606,
0.427082 , 0.4075344 , 0.27801244, 0.46554852, 0.41110647,
0.67803199, 0.67778171, 0.45384356, 0.70288098, 0.49589845,
0.33521412, 0.67085191, 0.28106945, 0.29555869, 0.79846318,
0.45102153, 0.3766174 , 0.70865144, 0.62552675, 0.45351656,
0.37558868, 0.28100754, 0.43122492, 0.49173493, 0.45789465,
0.69994627, 0.2346725 , 0.66342243, 0.38509685, 0.56293899,
0.51623693, 0.46984396, 0.63533874, 0.30807592, 0.45965632,
0.31211914, 0.6603035 , 0.23771173, 0.45172685, 0.59219831,
0.69657082, 0.37903697, 0.66784892, 0.23511688, 0.42365556,
0.56746406, 0.36772866, 0.62763363, 0.36102623, 0.51167312,
0.4816321 , 0.46400392, 0.3029109 , 0.53929915, 0.32031462,
0.76367516, 0.30999233, 0.42949746, 0.28428594, 0.35957699,
0.4102999 , 0.39792818, 0.34711551, 0.42686901, 0.58530715,
0.34126035, 0.56312586, 0.36548985, 0.71493759, 0.72430723,
0.42793974, 0.5976574 , 0.32734511, 0.29874224, 0.35792753,
0.4583545 , 0.54297118, 0.3090437 , 0.49198587, 0.34941334,
0.27754347, 0.44123681, 0.66041609, 0.75210876, 0.362279 ,
0.40137483, 0.60291957, 0.73499373, 0.51348446, 0.71378633,
0.40175571, 0.51895237, 0.8134646 , 0.65947701, 0.67823125,
0.39585658, 0.47033803, 0.29159387, 0.63323557, 0.40252538,
0.66315752, 0.19403498, 0.37135732, 0.29915948, 0.46845833,
0.72077689, 0.36887819, 0.36121344, 0.29981918, 0.23973407,
0.34171978, 0.32782223, 0.29481356, 0.67244403, 0.36920821,
0.21103227, 0.37802651, 0.29302044, 0.37438163, 0.44073788,
0.27810416, 0.29758946, 0.47427088, 0.37672225, 0.44649157,
0.37581168, 0.45210756, 0.67747076, 0.23566447, 0.52284276,
0.41583936, 0.48264037, 0.20500256, 0.21388442, 0.50674701,
0.40894837, 0.38581877, 0.52730513, 0.36305752, 0.73079706,
0.36729761, 0.3732552 , 0.69146807, 0.75001315, 0.33920191,
0.44586826, 0.47769028, 0.24922912, 0.44669284, 0.35485817,
0.45946401, 0.33105783, 0.41167352, 0.39813192, 0.48307752,
0.38778968, 0.31795408, 0.47365169, 0.47305093, 0.43605987,
0.67233549, 0.57328208, 0.29446604, 0.29259703, 0.66590803,
0.38933974, 0.33278845, 0.52448138, 0.44427442, 0.42765393,
0.44502617, 0.51544107, 0.42613252, 0.21715746, 0.55943394,
0.46590644, 0.30519844, 0.74822705, 0.46107516, 0.69784317,
0.37328422, 0.4544721, 0.44124492, 0.67608236, 0.45899145])

AUC and ROC Curve:

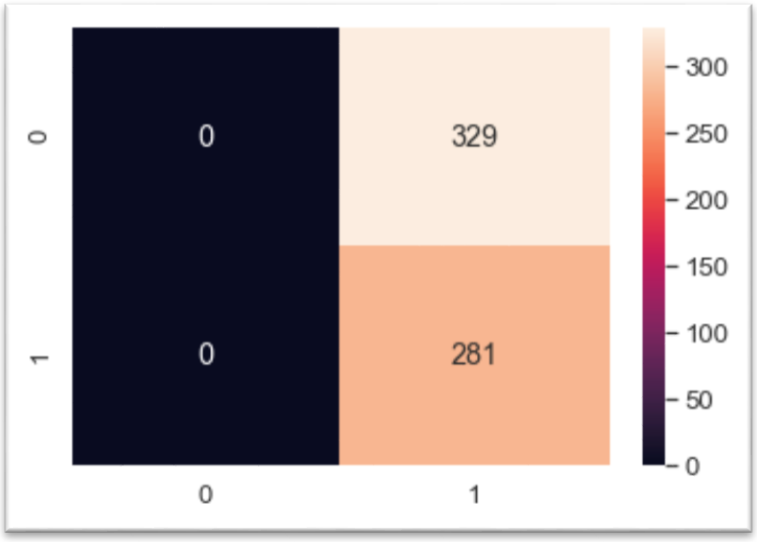
Graph 2.39 ROC Curve



Graph 2.40 ROC Curve

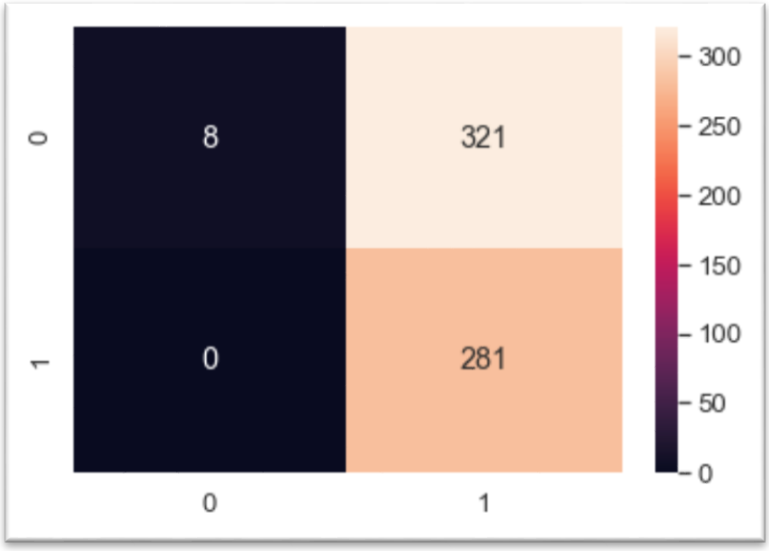
How to change the cut-off values for maximum accuracy?

0.1
Accuracy Score 0.4607
F1 Score 0.6308
Confusion Matrix



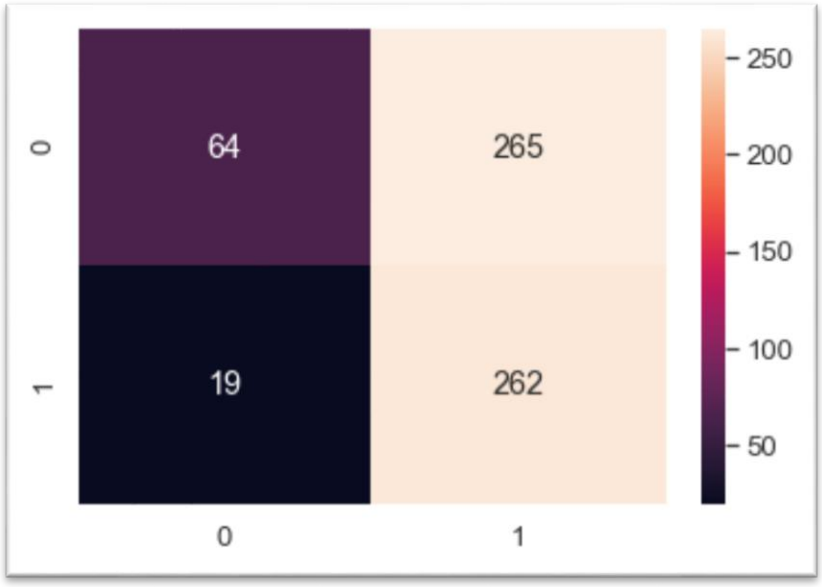
Graph 2.41 Confusion Matrix

0.2
Accuracy Score 0.4738
F1 Score 0.6365
Confusion Matrix



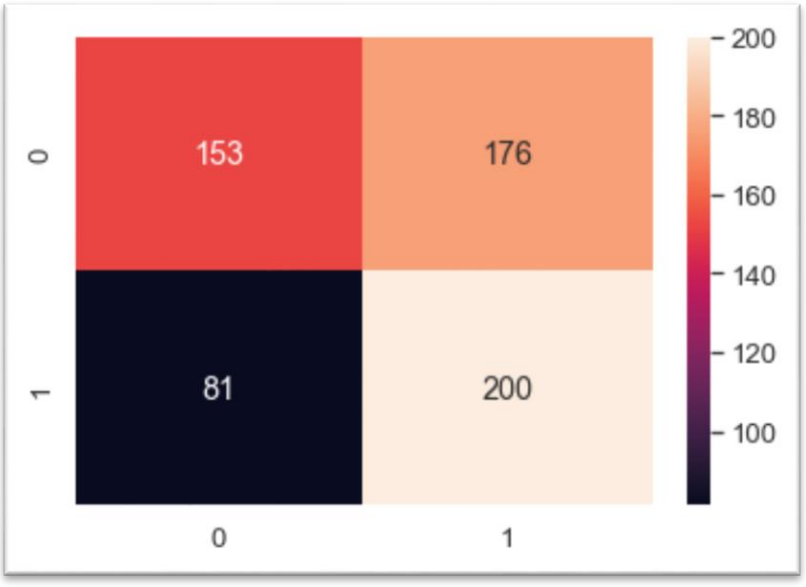
Graph 2.42 Confusion Matrix

0.3
Accuracy Score 0.5344
F1 Score 0.6485
Confusion Matrix



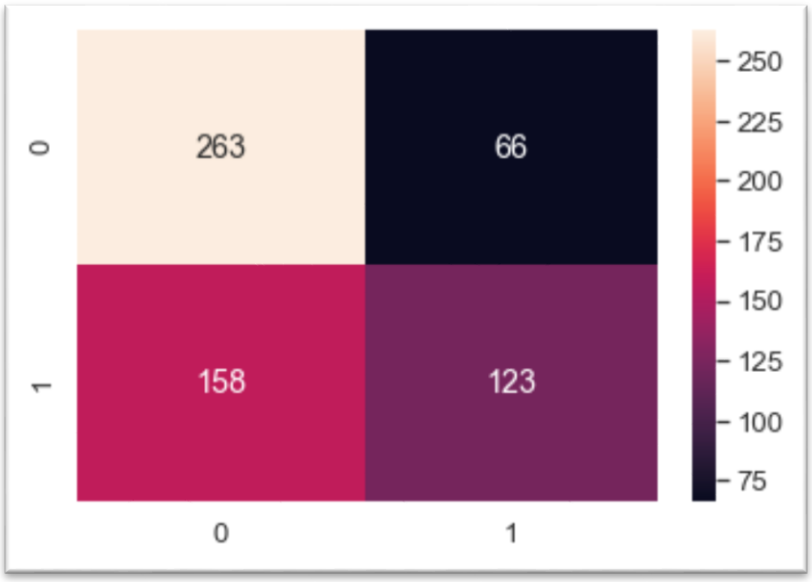
Graph 2.43 Confusion Matrix

0.4
Accuracy Score 0.5787
F1 Score 0.6088
Confusion Matrix



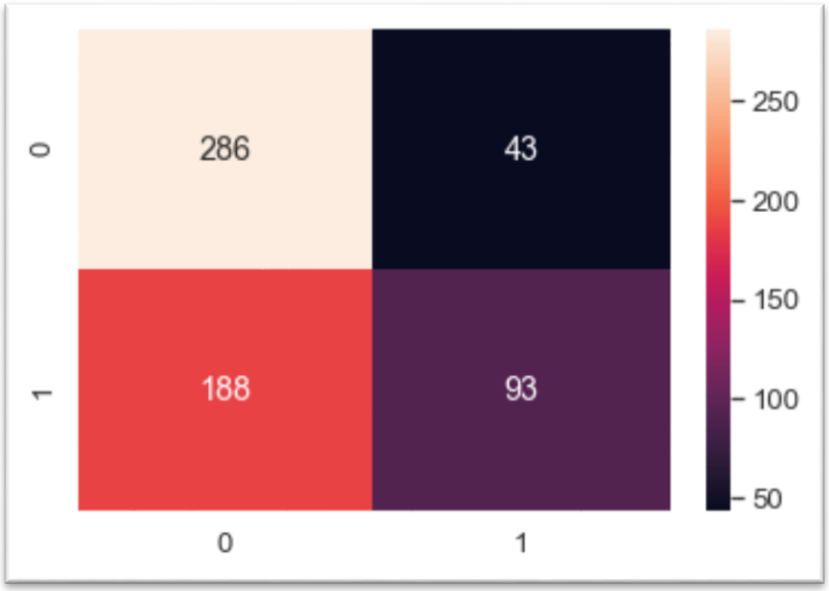
Graph 2.44 Confusion Matrix

0.5
Accuracy Score 0.6328
F1 Score 0.5234
Confusion Matrix



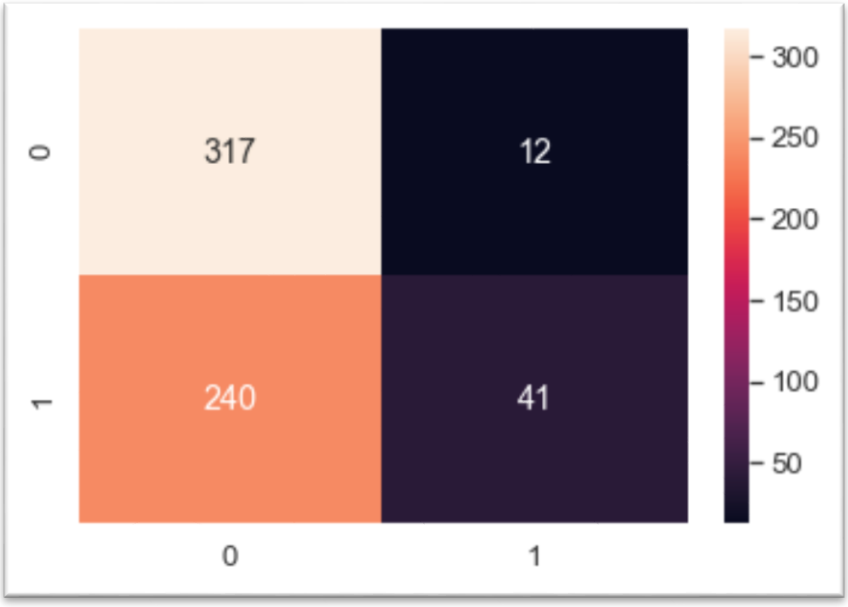
Graph 2.45 Confusion Matrix

0.6
Accuracy Score 0.6213
F1 Score 0.446
Confusion Matrix



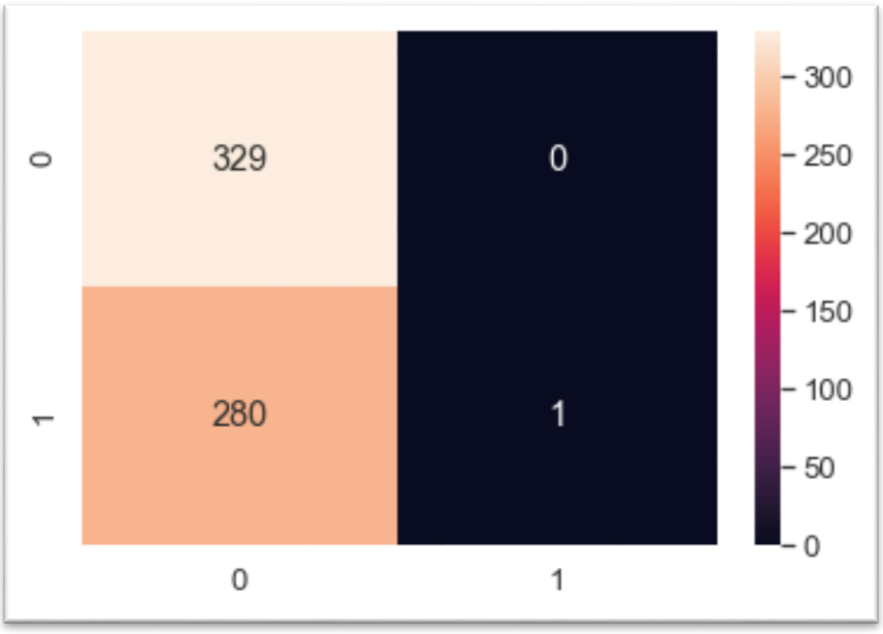
Graph 2.46 Confusion Matrix

0.7
Accuracy Score 0.5869
F1 Score 0.2455
Confusion Matrix



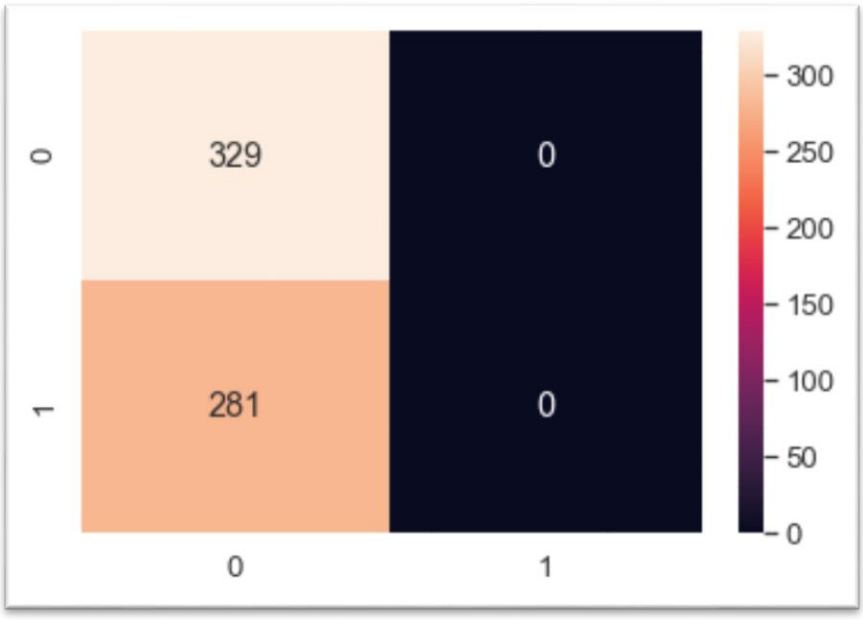
Graph 2.47 Confusion Matrix

0.8
Accuracy Score 0.541
F1 Score 0.0071
Confusion Matrix



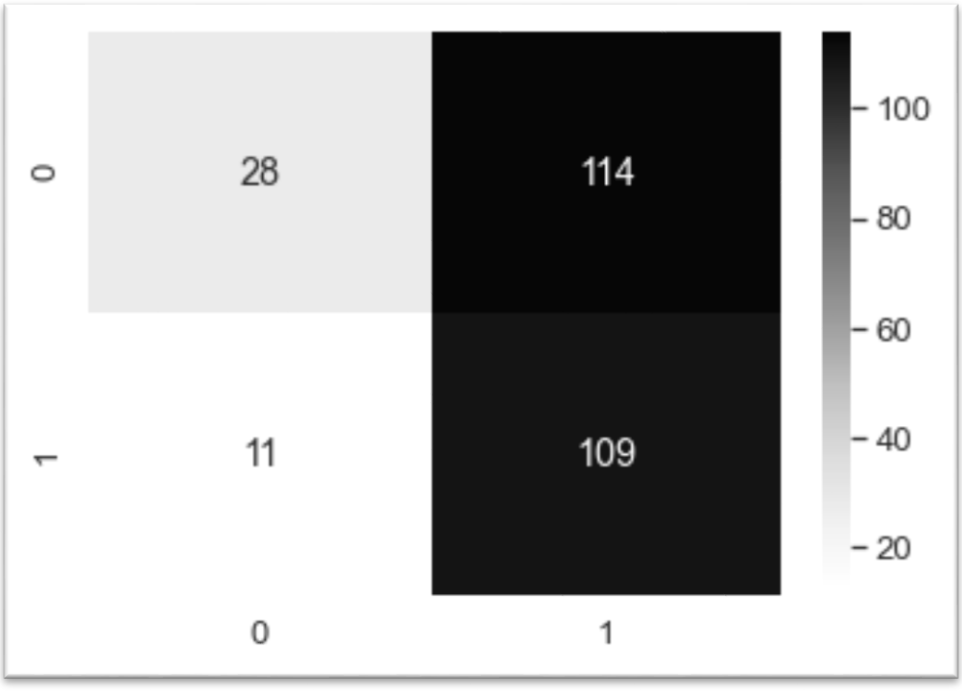
Graph 2.48 Confusion Matrix

0.9
Accuracy Score 0.5393
F1 Score 0.0
Confusion Matrix



Graph 2.49 Confusion Matrix

Predicting the classes on the test data:



Graph 2.50 Confusion Matrix

Classification Report of the default cut-off test data:

	precision	recall	f1-score	support
0	0.64	0.83	0.72	142
1	0.69	0.45	0.55	120
accuracy			0.66	262
macro avg	0.67	0.64	0.63	262
weighted avg	0.66	0.66	0.64	262

Classification Report of the custom cut-off test data:

	precision	recall	f1-score	support
0	0.72	0.20	0.31	142
1	0.49	0.91	0.64	120
accuracy			0.52	262
macro avg	0.60	0.55	0.47	262
weighted avg	0.61	0.52	0.46	262

Table 2.29 Classification Report of the Default / Custom cut off Test data

2.3 Performance Metrics: Check the performance of Predictions on Train and Test sets using Accuracy, Confusion Matrix, Plot ROC curve and get ROC_AUC score for each model Final Model: Compare Both the models and write inference which model is best/optimized.

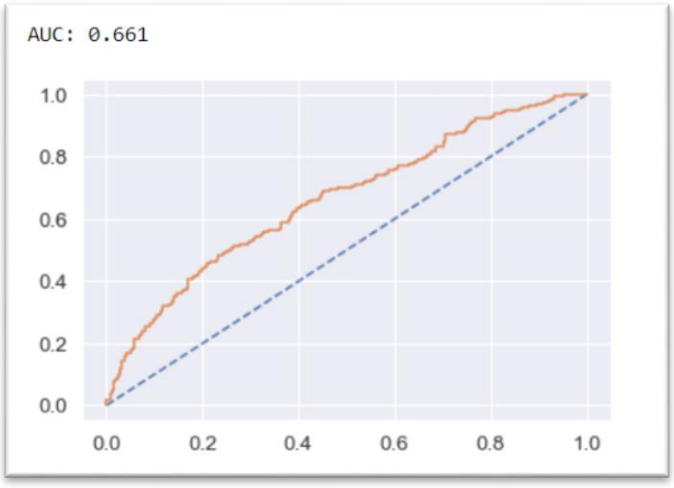
Logistic Regression:

Model Evaluation:

Accuracy Training Data:

```
0.6344262295081967
```

AUC and ROC for the Training data:

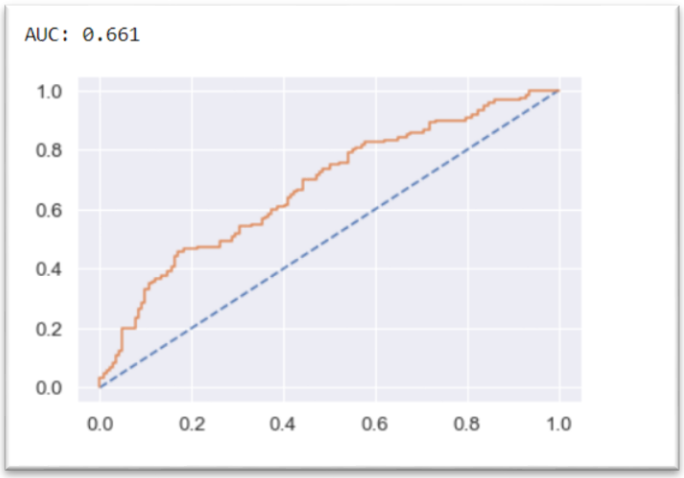


Graph 2.51 AUC Curve

Accuracy Test Data:

```
0.6603053435114504
```

AUC and ROC for the Test data:

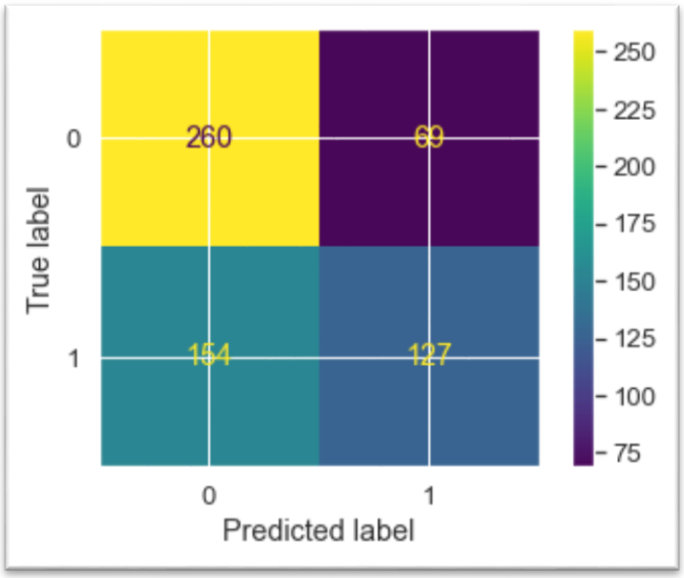


Graph 2.52 AUC Curve

Confusion Matrix for the training data:

```
array([[260,  69],
       [154, 127]], dtype=int64)
```

Table 2.30 Confusion Matrix for the training data

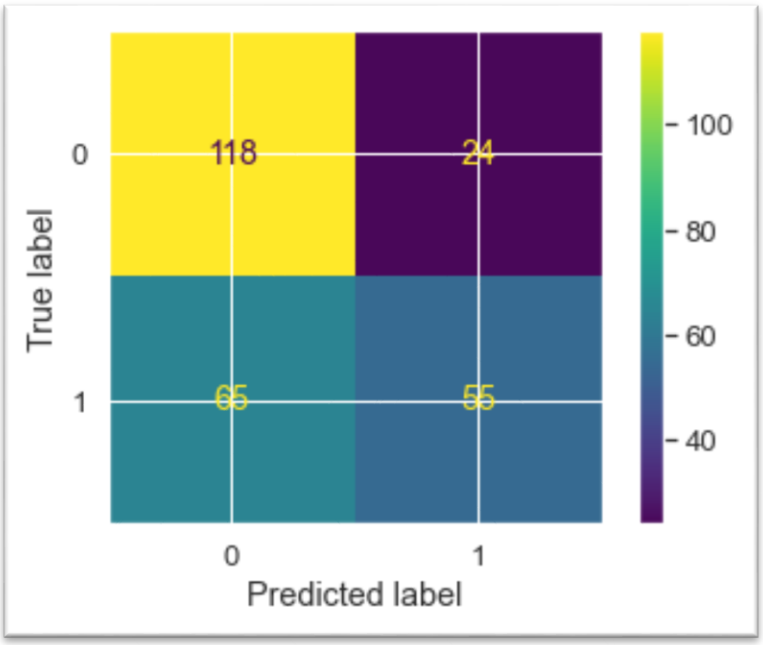


Graph 2.53 Confusion Matrix

	precision	recall	f1-score	support
0	0.63	0.79	0.70	329
1	0.65	0.45	0.53	281
accuracy			0.63	610
macro avg	0.64	0.62	0.62	610
weighted avg	0.64	0.63	0.62	610

Table 2.31 score card

Confusion Matrix for test data:



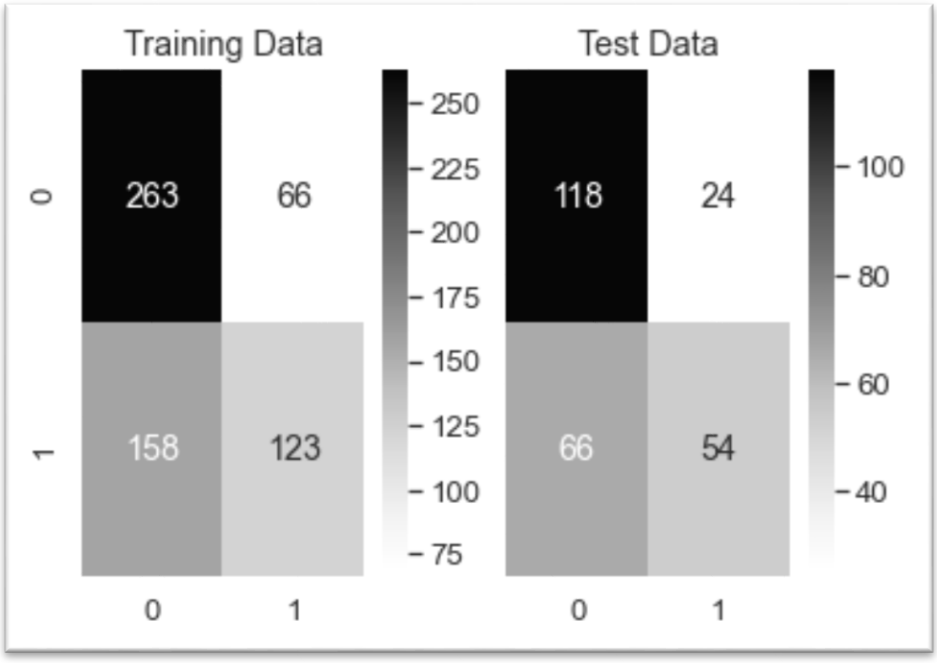
Graph 2.54 Confusion Matrix

	precision	recall	f1-score	support
0	0.64	0.83	0.73	142
1	0.70	0.46	0.55	120
accuracy			0.66	262
macro avg	0.67	0.64	0.64	262
weighted avg	0.67	0.66	0.65	262

Table 2.32 Score Card

LDA:

Training Data and Test Data Confusion Matrix Comparison:



Graph 2.55 Confusion Matrix

Training Data and Test Data Classification Report Comparison:

Model score:

```
0.6327868852459017
```

```
Classification Report of the training data:

              precision    recall  f1-score   support

           0       0.62      0.80      0.70       329
           1       0.65      0.44      0.52       281

    accuracy                           0.63       610
   macro avg       0.64      0.62      0.61       610
weighted avg       0.64      0.63      0.62       610
```

Table 2.33 Classification Report of the Training Data

```
array([[263,  66],
       [158, 123]], dtype=int64)
```

Table 2.34 Confusion Matrix

Model Score:

```
0.6564885496183206
```

```
Classification Report of the test data:

              precision    recall  f1-score   support

           0       0.64      0.83      0.72       142
           1       0.69      0.45      0.55       120

    accuracy                           0.66       262
   macro avg       0.67      0.64      0.63       262
weighted avg       0.66      0.66      0.64       262
```

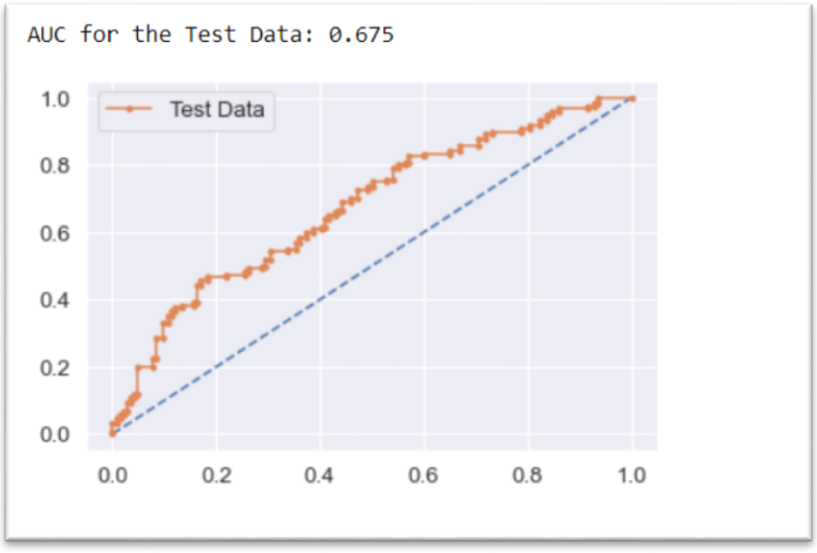
Table 2.35 Classification Report of Test data

```
array([[118,  24],
       [ 66,  54]], dtype=int64)
```

Table 2.36 Confusion Matrix

AUC and ROC Curve:

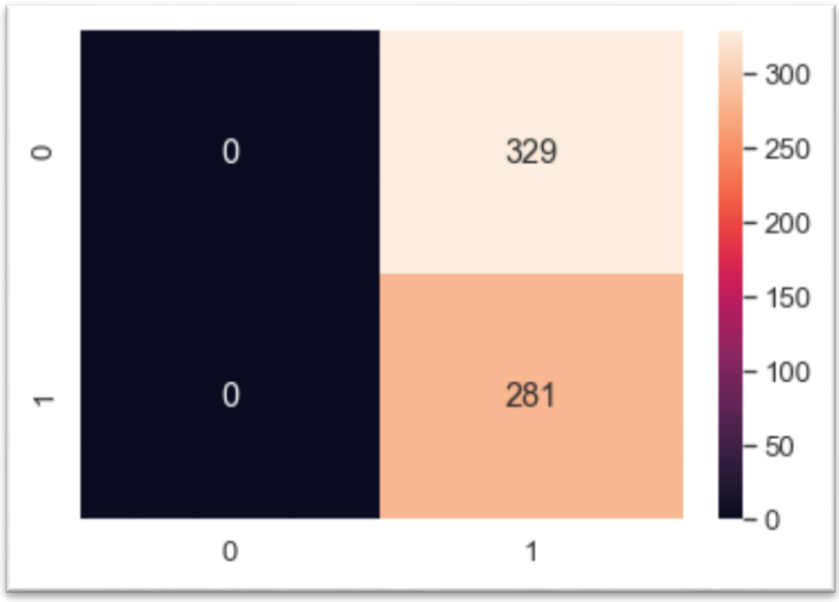
Graph 2.56 ROC Curve



Graph 2.57 ROC Curve

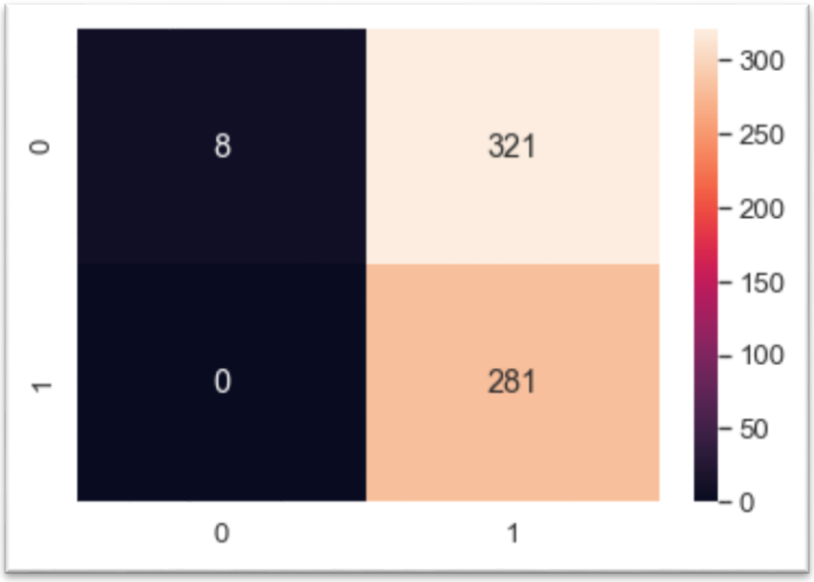
How to change the cut-off values for maximum accuracy?

0.1
Accuracy Score 0.4607
F1 Score 0.6308
Confusion Matrix



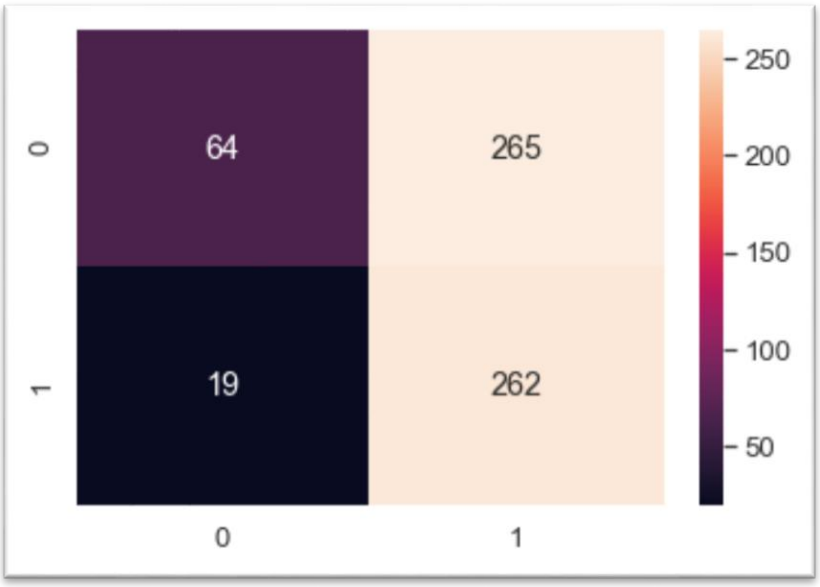
Graph 2.58 Confusion Matrix

0.2
Accuracy Score 0.4738
F1 Score 0.6365
Confusion Matrix



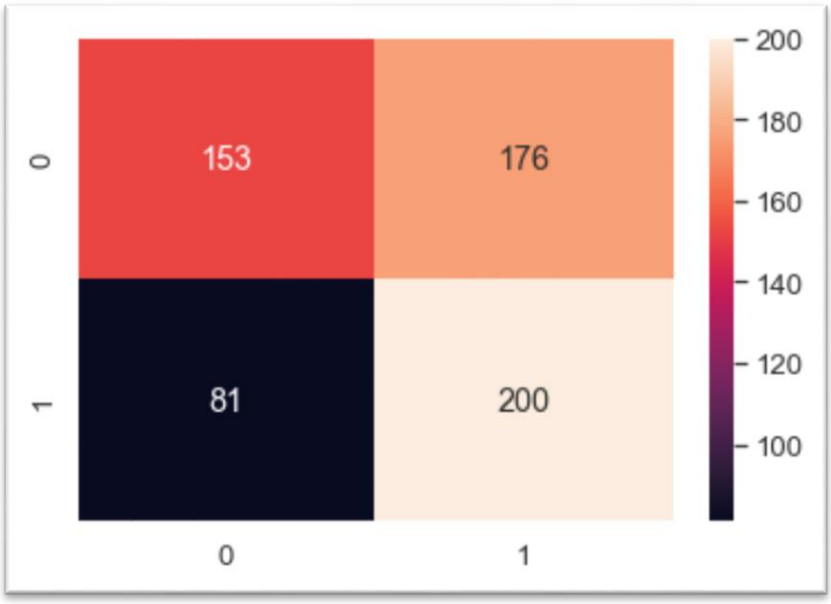
Graph 2.59 Confusion Matrix

0.3
Accuracy Score 0.5344
F1 Score 0.6485
Confusion Matrix



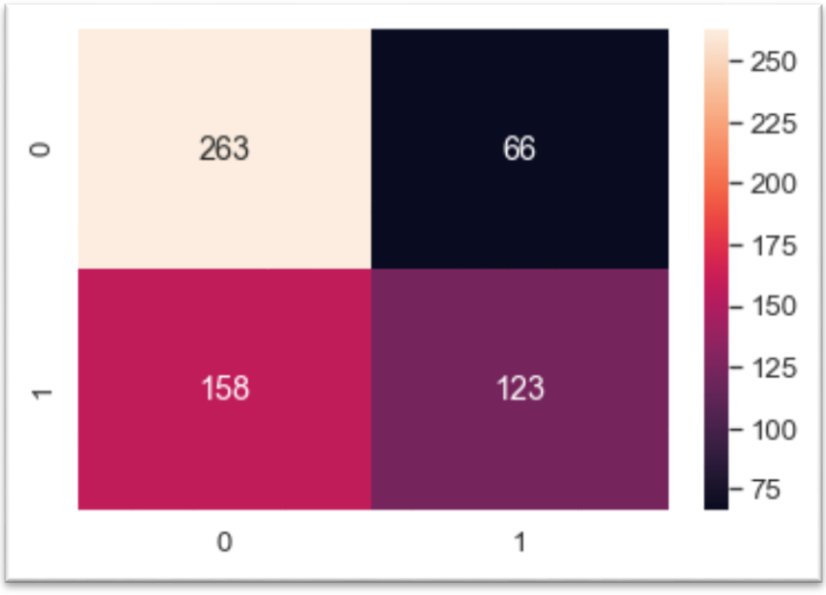
Graph 2.60 Confusion Matrix

0.4
Accuracy Score 0.5787
F1 Score 0.6088
Confusion Matrix



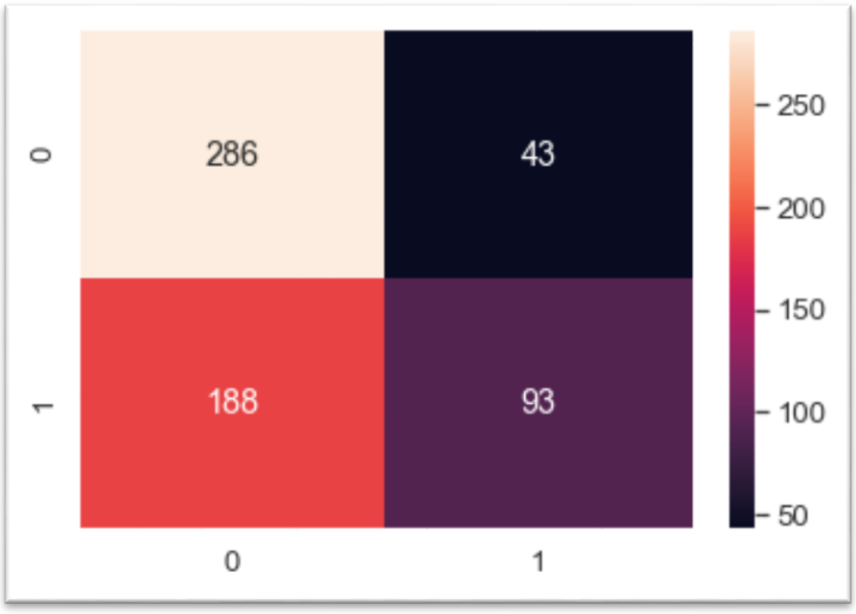
Graph 2.61 Confusion Matrix

0.5
Accuracy Score 0.6328
F1 Score 0.5234
Confusion Matrix



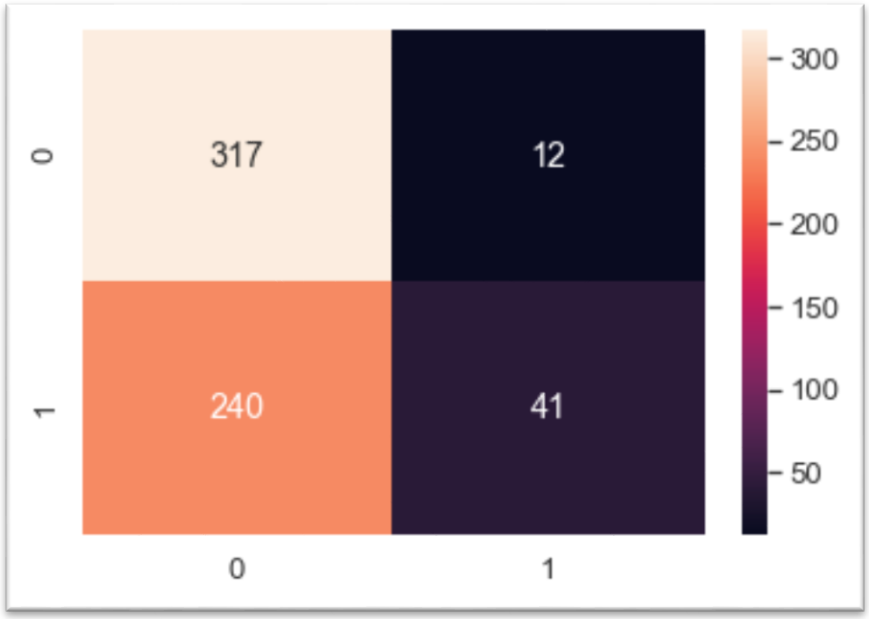
Graph 2.62 Confusion Matrix

0.6
Accuracy Score 0.6213
F1 Score 0.446
Confusion Matrix



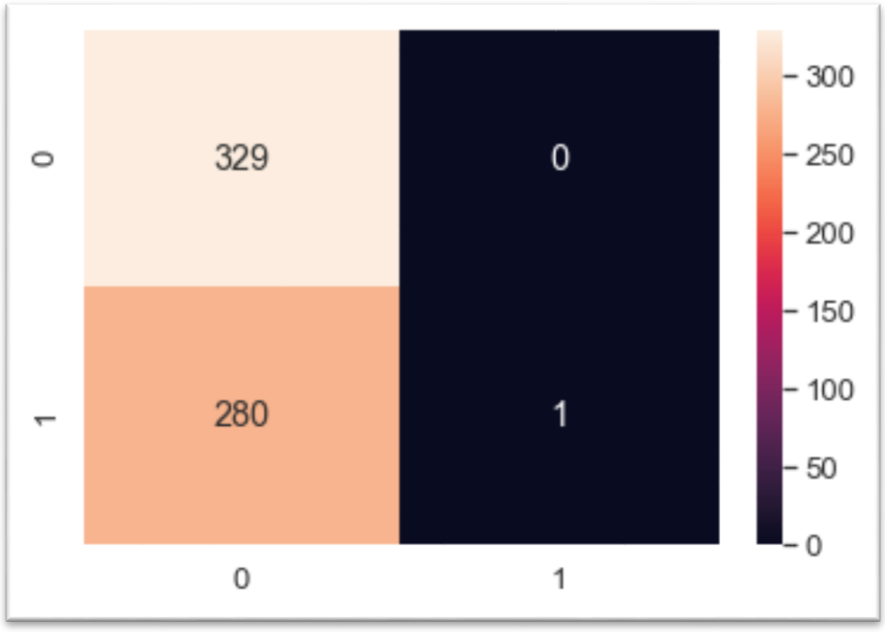
Graph 2.63 Confusion Matrix

0.7
Accuracy Score 0.5869
F1 Score 0.2455
Confusion Matrix



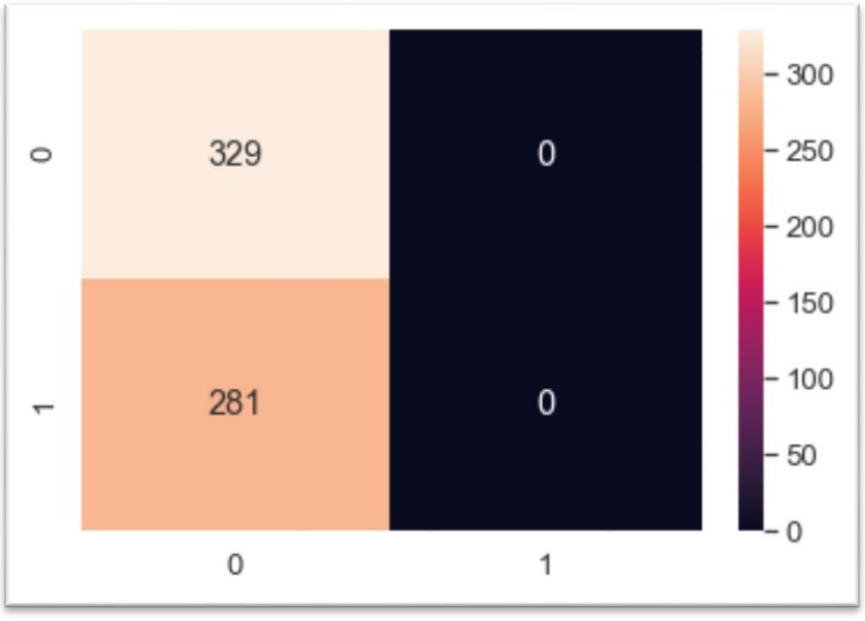
Graph 2.64 Confusion Matrix

0.8
Accuracy Score 0.541
F1 Score 0.0071
Confusion Matrix



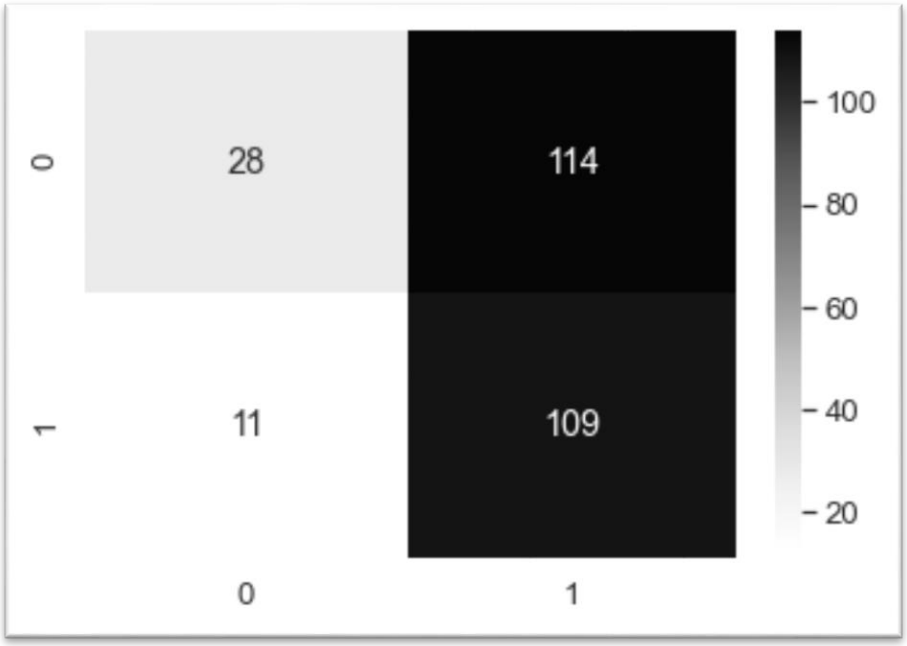
Graph 2.65 Confusion Matrix

0.9
Accuracy Score 0.5393
F1 Score 0.0
Confusion Matrix



Graph 2.66 Confusion Matrix

Predicting the classes on the test data:



Graph 2.67 Confusion Matrix

Classification Report of the default cut-off test data:

	precision	recall	f1-score	support
0	0.64	0.83	0.72	142
1	0.69	0.45	0.55	120
accuracy			0.66	262
macro avg	0.67	0.64	0.63	262
weighted avg	0.66	0.66	0.64	262

Classification Report of the custom cut-off test data:

	precision	recall	f1-score	support
0	0.72	0.20	0.31	142
1	0.49	0.91	0.64	120
accuracy			0.52	262
macro avg	0.60	0.55	0.47	262
weighted avg	0.61	0.52	0.46	262

Table 2.37 Classification Report of the Default / Custom cut off Test data

	LR Train	LR Test	LDA Train	LDA Test
Accuracy	0.63	0.66	0.63	0.66
AUC	0.66	0.68	0.66	0.68
Recall	0.45	0.45	0.44	0.45
Precision	0.65	0.69	0.65	0.69
F1 Score	0.53	0.55	0.52	0.55

Table 2.38 Score Card Comparison

Comparing both these models, we find both results are same, but LDA works better when there is category target variable.

2.4 Inference: Basis on these predictions, what are the insights and recommendations.

We had a business problem where we need predict whether an employee would opt for a holiday package or not, for this problem we had done predictions both logistic regression and linear discriminant analysis. Since both are results are same. The EDA analysis clearly indicates certain criteria where we could find people aged above 50 are not interested much in holiday packages. So this is one of the we find aged people not opting for holiday packages. People ranging from the age 30 to 50 generally opt for holiday packages. Employee age over 50 to 60 have seems to be not taking the holiday package, whereas in the age 30 to 50 and salary less than 50000 people have opted more for holiday package.

The important factors deciding the predictions are salary, age and educ.

Recommendations
1. To improve holiday packages over the age above 50 we can provide religious destination places.
2. For people earning more than 150000 we can provide vacation holiday packages.
3. For employee having more than number of older children we can provide packages in holiday vacation places.