GREATLEARNING

PREDICTIVE MODELING

PROJECT REPORT

BY,

RAGAVEDHNI K R

Problem 1: **Linear Regression**

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.

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

- The data is read from 'cubic_zirconia.csv' and the initial set of rows is shown as below:

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

- The number of rows and columns in the data can be viewed as the shape of the data:

```
The shape of the data: (26967, 10)
```

The information of the data type and individual non-null values in each column is:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 26967 entries, 0 to 26966
Data columns (total 10 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   carat    26967 non-null  float64
 1   cut      26967 non-null  object
 2   color    26967 non-null  object
 3   clarity  26967 non-null  object
 4   depth    26270 non-null  float64
 5   table    26967 non-null  float64
 6   x        26967 non-null  float64
 7   y        26967 non-null  float64
 8   z        26967 non-null  float64
 9   price    26967 non-null  int64
dtypes: float64(6), int64(1), object(3)
memory usage: 2.1+ MB
```

- To get the statistical summary of the data using the describe method as:

	carat	cut	color	clarity	depth	table	x	y	z	price
count	26967.000000	26967	26967	26967	26270.000000	26967.000000	26967.000000	26967.000000	26967.000000	26967.000000
unique	NaN	5	7	8	NaN	NaN	NaN	NaN	NaN	NaN
top	NaN	Ideal	G	SI1	NaN	NaN	NaN	NaN	NaN	NaN
freq	NaN	10816	5661	6571	NaN	NaN	NaN	NaN	NaN	NaN
mean	0.798375	NaN	NaN	NaN	61.745147	57.456080	5.729854	5.733569	3.538057	3939.518115
std	0.477745	NaN	NaN	NaN	1.412860	2.232068	1.128516	1.166058	0.720624	4024.864666
min	0.200000	NaN	NaN	NaN	50.800000	49.000000	0.000000	0.000000	0.000000	326.000000
25%	0.400000	NaN	NaN	NaN	61.000000	56.000000	4.710000	4.710000	2.900000	945.000000
50%	0.700000	NaN	NaN	NaN	61.800000	57.000000	5.690000	5.710000	3.520000	2375.000000
75%	1.050000	NaN	NaN	NaN	62.500000	59.000000	6.550000	6.540000	4.040000	5360.000000
max	4.500000	NaN	NaN	NaN	73.600000	79.000000	10.230000	58.900000	31.800000	18818.000000

- While checking the null values in the columns:

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

- The check for the duplicated rows is made on the data and dropped duplicates except for the first occurrence.

```
Shape of data before checking the duplicates: (26967, 10)
Total Duplicates in the data: 34
Shape of data after discarding the duplicates: (26933, 10)
```

carat	cut	color	clarity	depth	table	x	y	z	price

- Checking the unique categorical variables in the data.

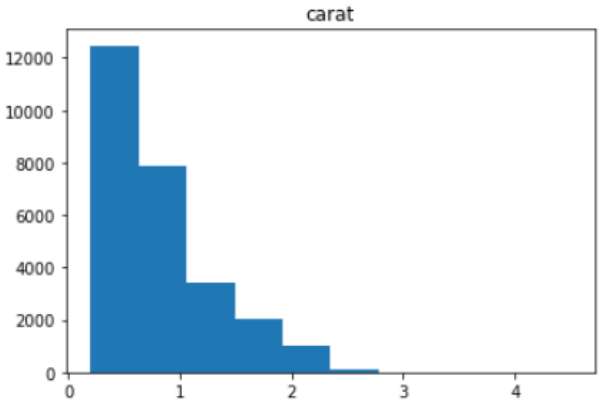
```
CUT :  5
Fair          780
Good         2435
Very Good    6027
Premium      6886
Ideal       10805
Name: cut, dtype: int64
```

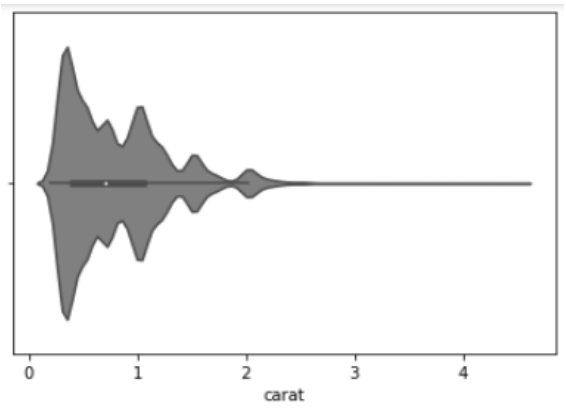
```
COLOR :  7
J    1440
I    2765
D    3341
H    4095
F    4723
E    4916
G    5653
Name: color, dtype: int64
```

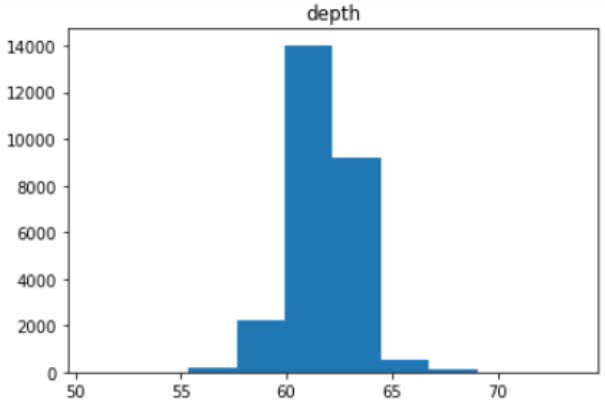
```
CLARITY :  8
I1       364
IF       891
VVS1    1839
VVS2    2530
VS1     4087
SI2     4564
VS2     6093
SI1     6565
Name: clarity, dtype: int64
```

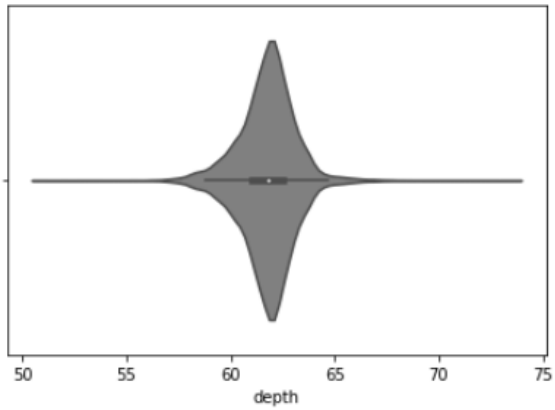
UNIVARIATE ANALYSIS:

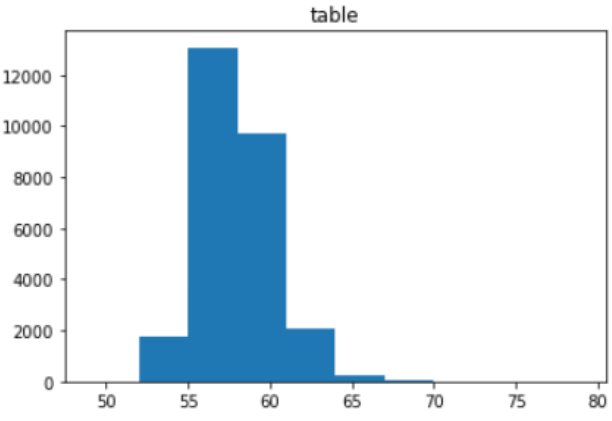
- The plots provide information about the distribution of the observations in the single data variable.
- Here we consider the hist plot (from matplotlib package) and violinplot (from sns package) to know the distribution of the individual variables from the dataset.

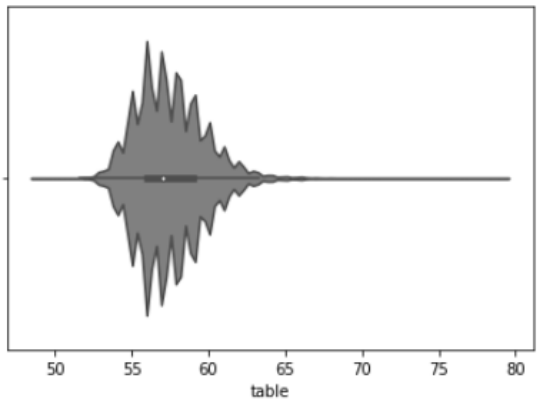


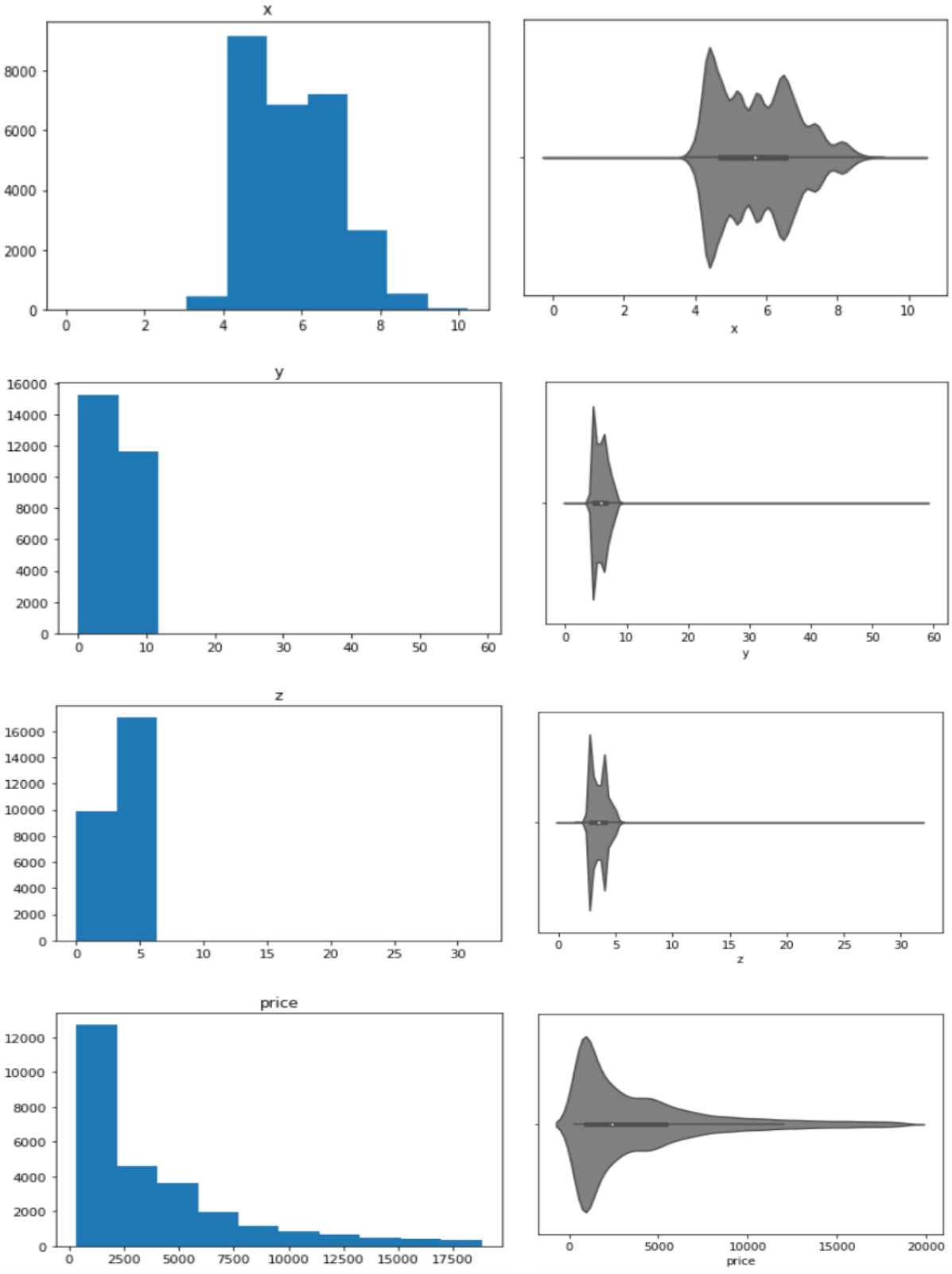






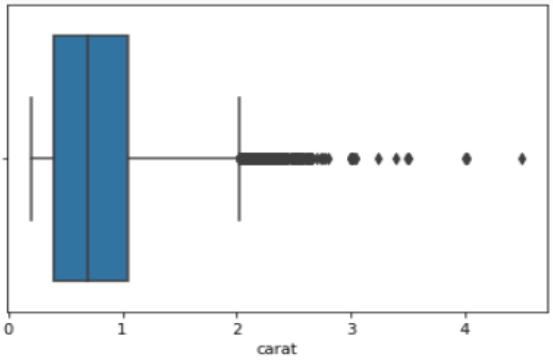


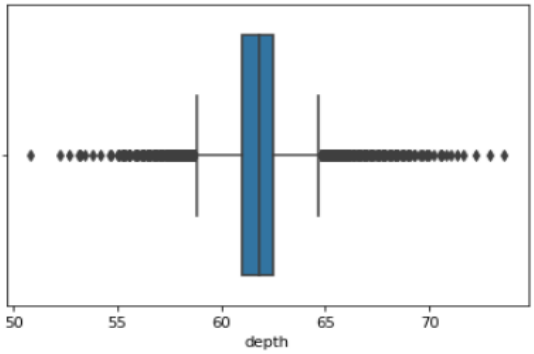


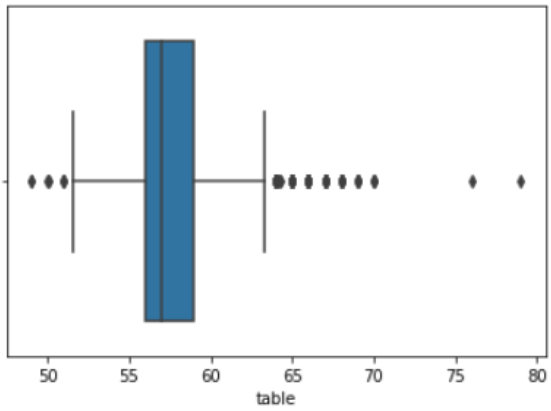


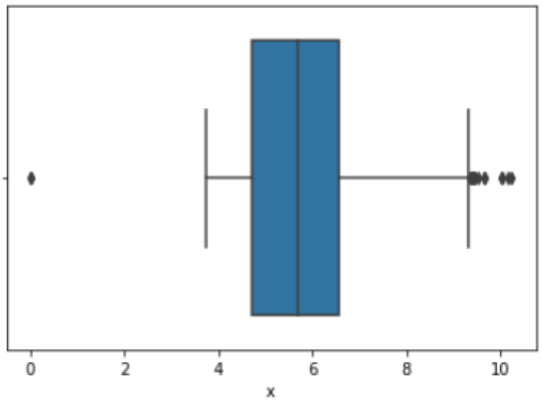
- The right skewed (or positively skewed) distribution has large occurrence in the left side and few in the right side. Here, mean is greater than the median.
- The left skewed (or negatively skewed) distribution has large occurrence in the right side and few in the left side. Here, mean is less than the median

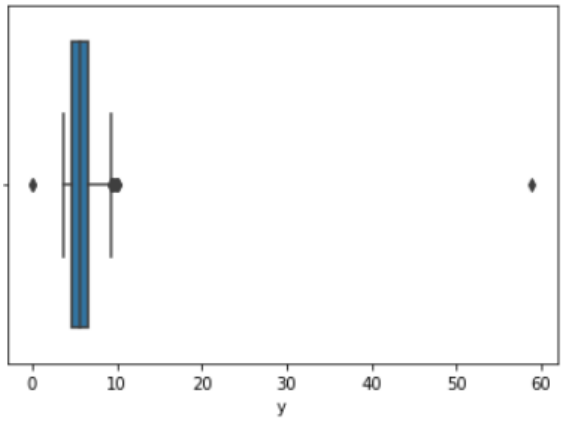
- The symmetric distribution is the bell-shaped or normal distribution.
- Checking the outliers using BoxPlot from sns package.

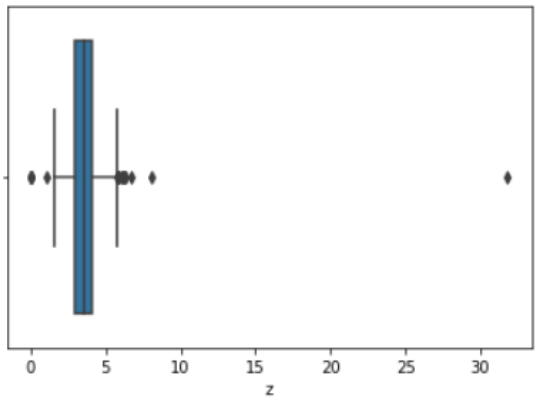


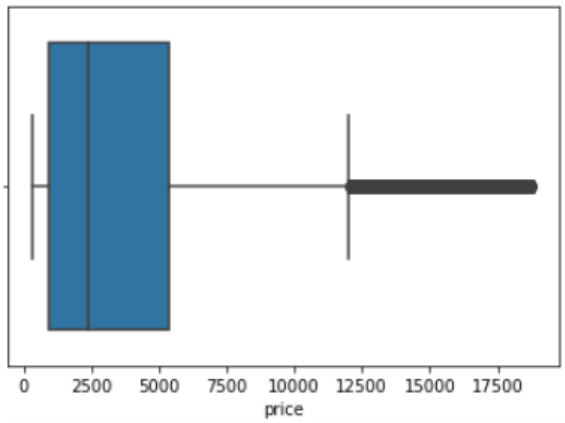




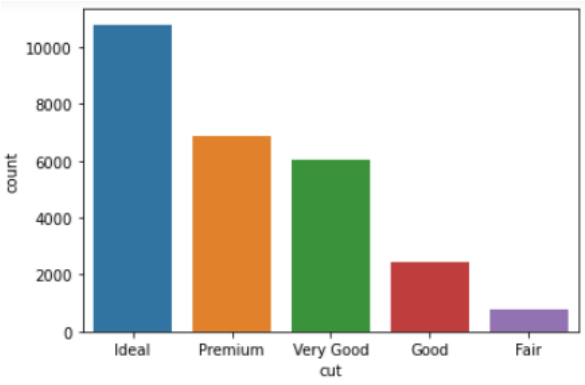


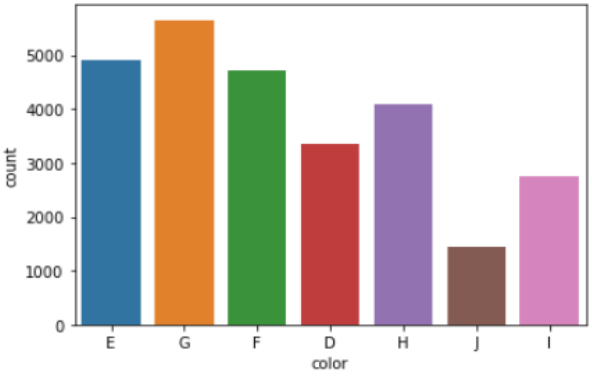


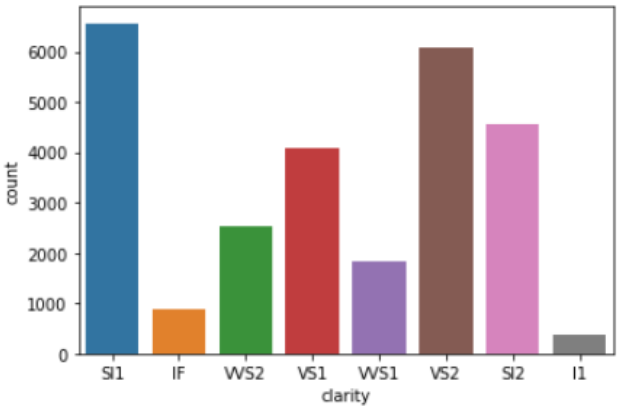




- The categorical variables are visualized using countplot from sns package,it shows the count of observations in each categorical bin.

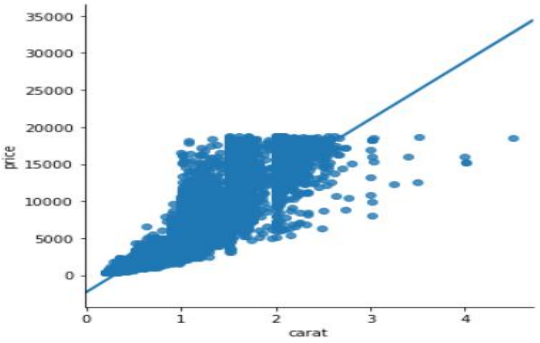


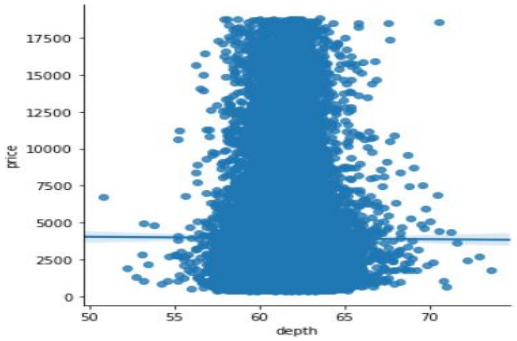


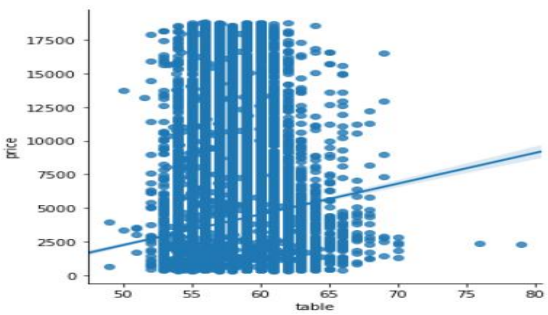


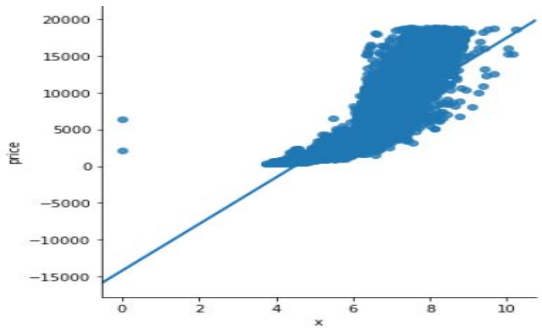
BIVARIATE ANALYSIS:

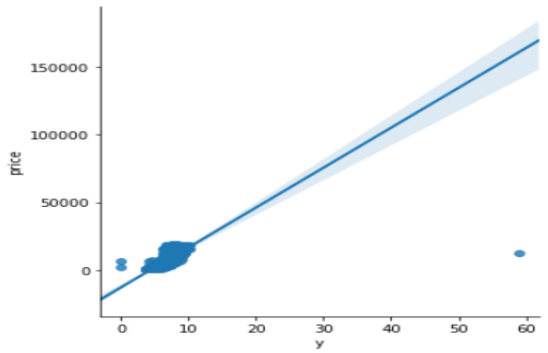
- The lmplot from sns package, to see the correlation between two attributes plotted against the x-axis and y-axis in a 2-dimensional plane. X – Predictor variable, Y – Target variable.

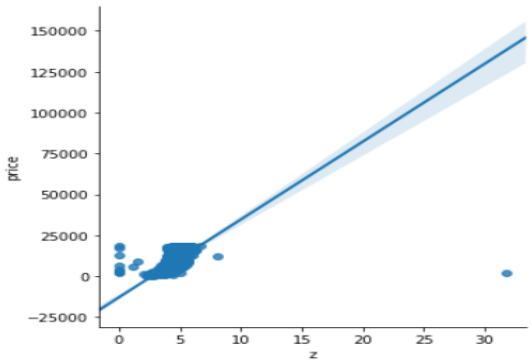




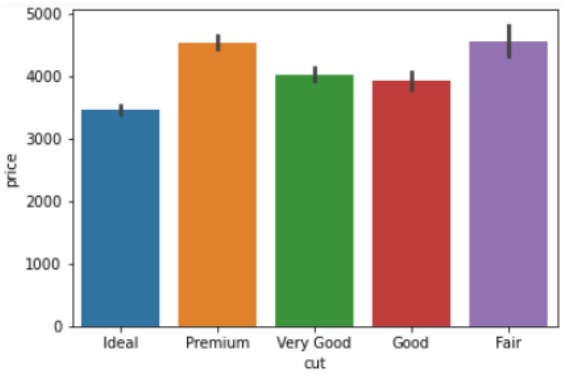


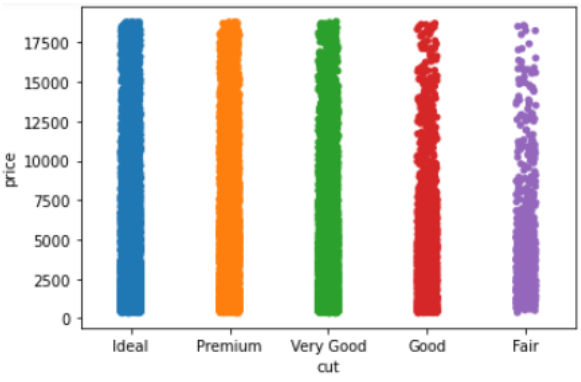


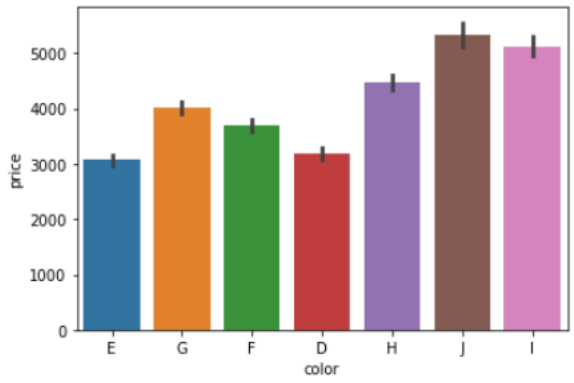


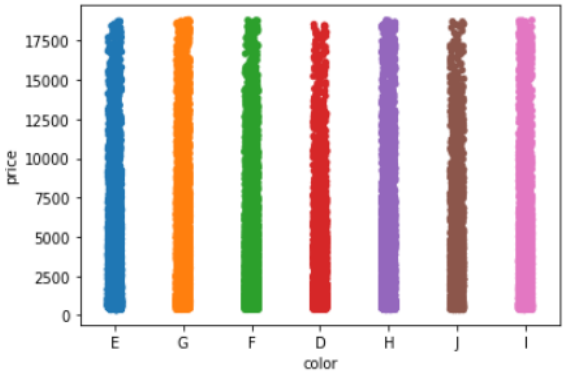


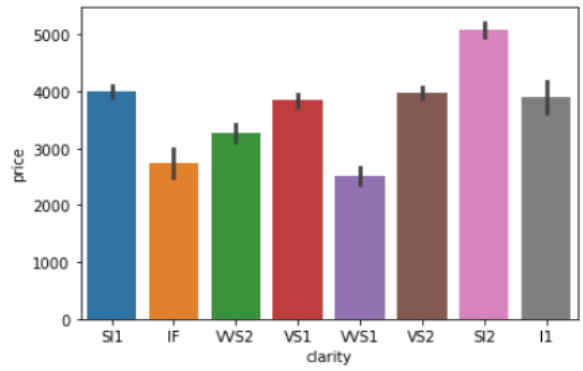
- The barplot from sns package shows the distribution of values at each level of the categorical variables as point estimates and confidence intervals as rectangular bars to make comparison against a quantitative variable.
- The stripplot from sns package shows all observations along with some representation of the underlying distribution where one variable is categorical.

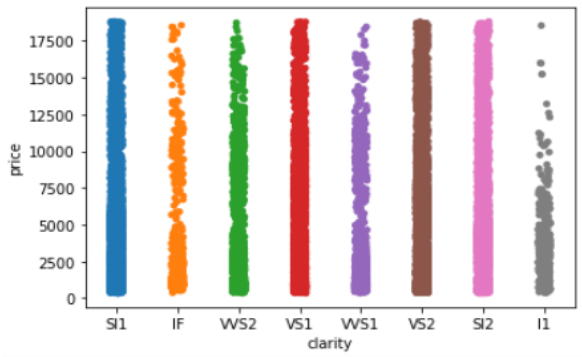






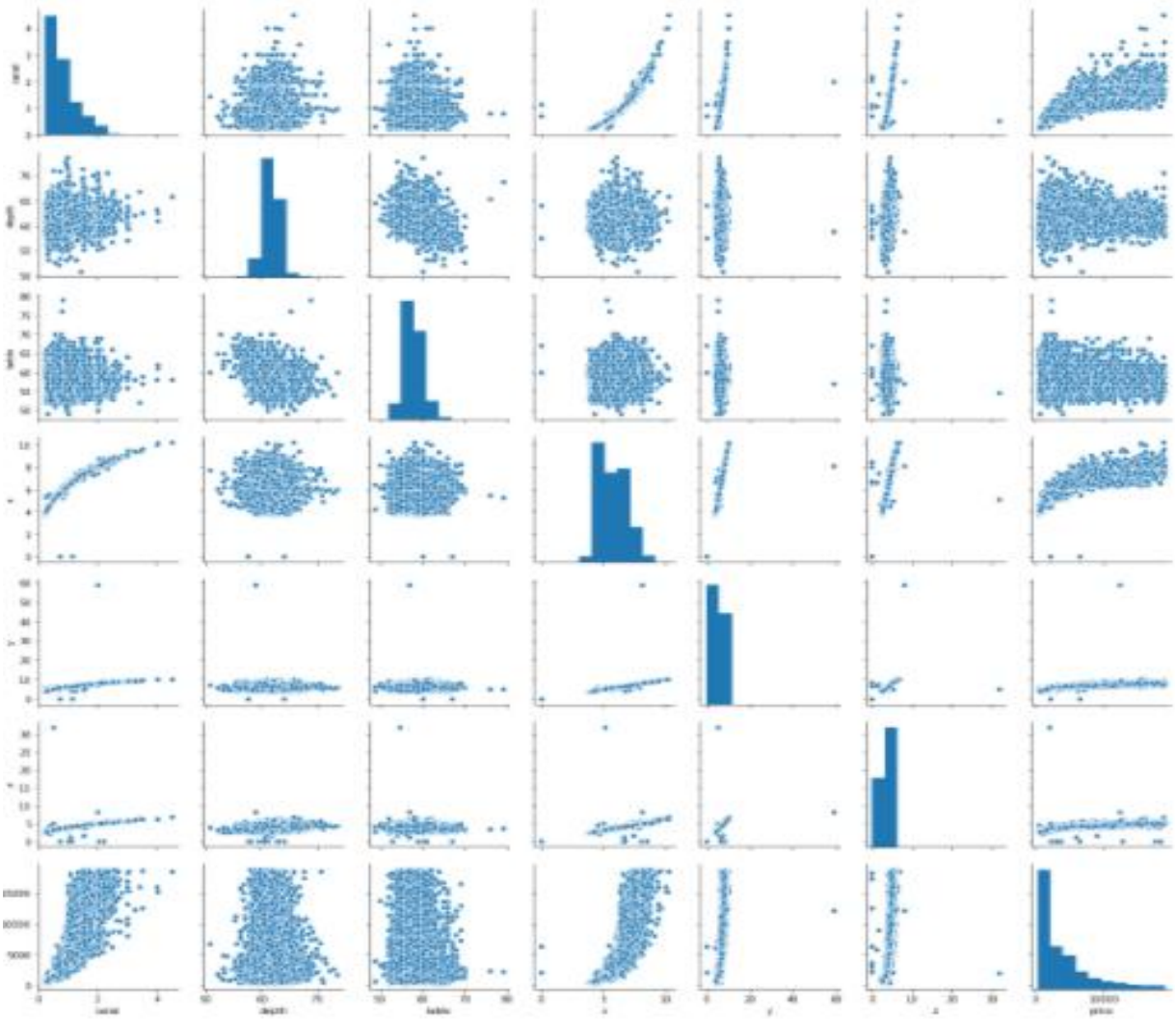






MUTLI VARIATE:

- The pairplot (from sns package) is used for visualizing the pair wise relationship across entire dataframe.
- There is a linear relationship between the variable carat and the variables x, y, z.
- For the target column price, we can see good linear relationship with the columns carat and x, and fair linear relationship with the columns y and z.

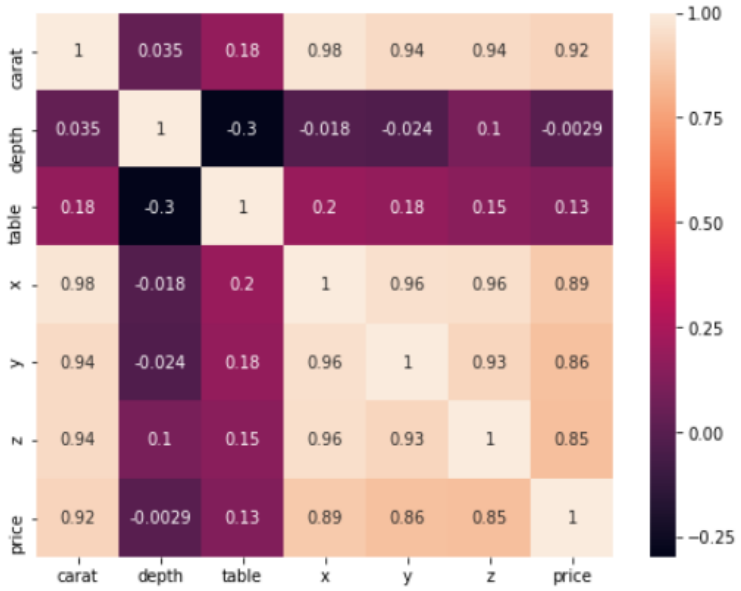


- The correlation across all the numerical or continuous variables can be found using

corr() function in a matrix form.

- To indicate and visualize the clusters within the data using the heatmap (from sns package).
- The predictor variables in the columns carat, x, y, z are highly correlated with the target column price.
- The predictor column table is fairly correlated to target column price, as they are close to 0.
- The variable depth is not much correlated with the target variable; it might be due to presence of large null values present.

	carat	depth	table	x	y	z	price
carat	1.000000	0.035240	0.181539	0.976858	0.941442	0.940982	0.922409
depth	0.035240	1.000000	-0.297768	-0.018401	-0.024453	0.101973	-0.002895
table	0.181539	-0.297768	1.000000	0.196254	0.182352	0.148994	0.126844
x	0.976858	-0.018401	0.196254	1.000000	0.962601	0.956490	0.886554
y	0.941442	-0.024453	0.182352	0.962601	1.000000	0.928725	0.856441
z	0.940982	0.101973	0.148994	0.956490	0.928725	1.000000	0.850682
price	0.922409	-0.002895	0.126844	0.886554	0.856441	0.850682	1.000000



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? Do you think scaling is necessary in this case?

- Checking the null values in the dataset.
- The dataset has null values in the depth column for 697 rows. The depth column has huge outliers present in them, so the preferred type of imputation is median imputation.

```
The Null values in the data  697 for the column  depth
```

Checking Null values:

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

After Median Imputation:

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

- After the Median imputation for depth column, we check for the null values in the dataset.
- We check for the values which are equal to zero, if they are valid or we need to change or drop them.

```
In the column  x :
```

	carat	cut	color	clarity	depth	table	x	y	z	price
5821	0.71	Good	F	SI2	64.1	60.0	0.0	0.0	0.0	2130
17506	1.14	Fair	G	VS1	57.5	67.0	0.0	0.0	0.0	6381

```
In the column  y :
```

	carat	cut	color	clarity	depth	table	x	y	z	price
5821	0.71	Good	F	SI2	64.1	60.0	0.0	0.0	0.0	2130
17506	1.14	Fair	G	VS1	57.5	67.0	0.0	0.0	0.0	6381

```
In the column  z :
```

	carat	cut	color	clarity	depth	table	x	y	z	price
5821	0.71	Good	F	SI2	64.1	60.0	0.00	0.00	0.0	2130
6034	2.02	Premium	H	VS2	62.7	53.0	8.02	7.95	0.0	18207
10827	2.20	Premium	H	SI1	61.2	59.0	8.42	8.37	0.0	17265
12498	2.18	Premium	H	SI2	59.4	61.0	8.49	8.45	0.0	12631
12689	1.10	Premium	G	SI2	63.0	59.0	6.50	6.47	0.0	3696
17506	1.14	Fair	G	VS1	57.5	67.0	0.00	0.00	0.0	6381
18194	1.01	Premium	H	I1	58.1	59.0	6.66	6.60	0.0	3167
23758	1.12	Premium	G	I1	60.4	59.0	6.71	6.67	0.0	2383

- The column x (Length) and y (Width) has 2 rows, while the column z (Height) has 8 rows with value 0.
- We know that Depth is measured using Height of the cubic zirconia and Table is measured using Width of the cubic zirconia.
- Hence, the values cannot be 0 in all 3 columns, while their depth and table values are non-zero and considering that the values might have been missed unintentionally.
- The rows with the value 0 are replaced as null values, which are further imputed with Median Imputation.

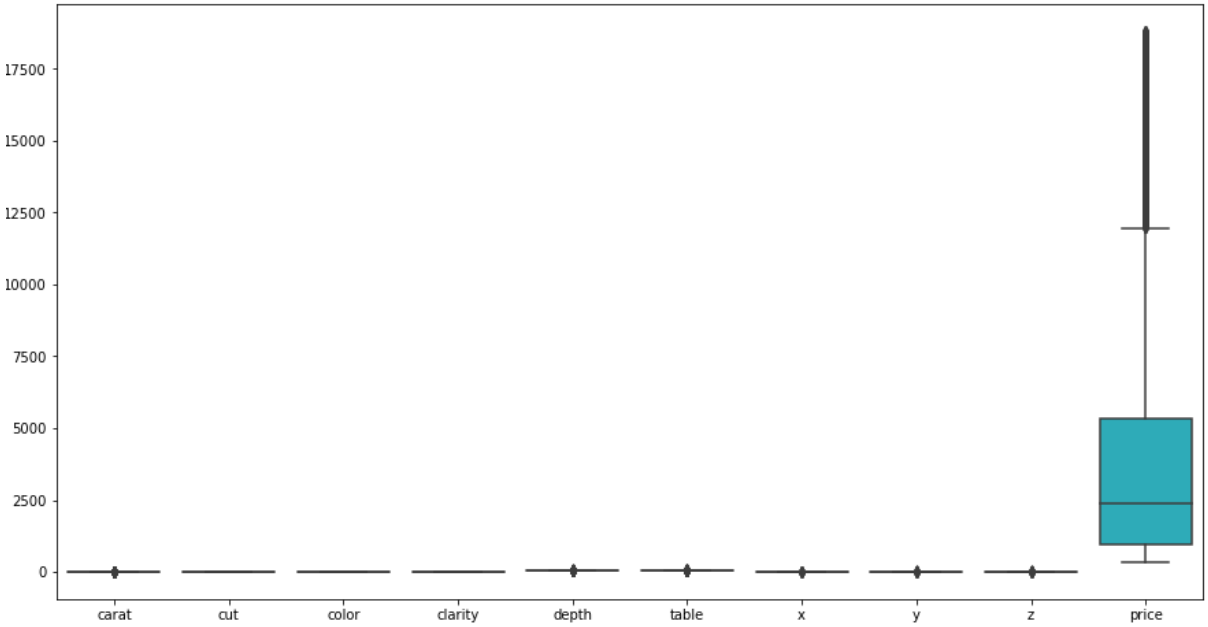
```
In the column  x :
Empty DataFrame
Columns: [carat, cut, color, clarity, depth, table, x, y, z, price]
Index: []
In the column  y :
Empty DataFrame
Columns: [carat, cut, color, clarity, depth, table, x, y, z, price]
Index: []
In the column  z :
Empty DataFrame
Columns: [carat, cut, color, clarity, depth, table, x, y, z, price]
Index: []
```

Outlier Treatment:

- In Data Pre-processing step, the outliers have to be checked.
- If outliers are present, then they have to be treated or removed.

```
Shape of the data without outlier treatment : (23618, 10)
Shape of the original data  : (26933, 10)
```

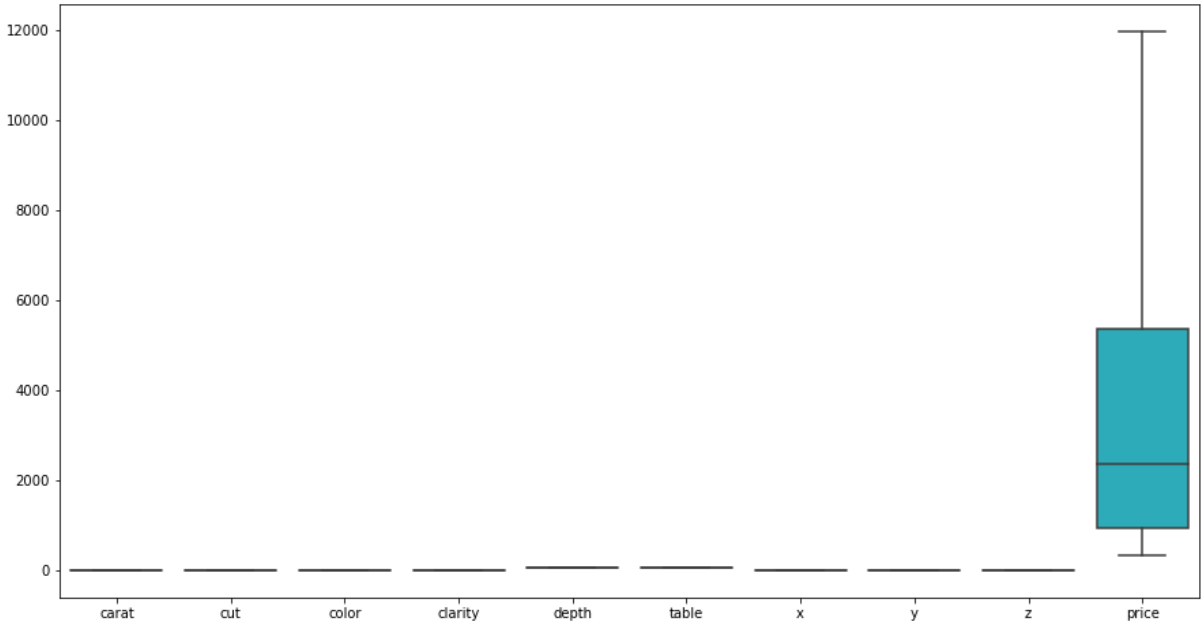
- The Linear Regression model is affected by the presence of outliers, hence they must be handled.
- We can know the outliers in the data, while we club all the variables together as they depend on each other and project the outliers clearly as seen below.



- Here, we use IQR (Inter-Quartile Range) imputation method for the outlier treatment.

```
Shape of the data after outlier treatment : (26933, 10)
Shape of the original data  : (26933, 10)
```

- Checking the data after outlier treatment using the boxplot with all the columns combined.
- We can see that all the columns are free from outliers.



Scaling Data after Outlier Treatment:

- To check if scaling is required or not for the data, initially we apply the Linear Regression model on the non-scaled data.
- We can see that the value of coefficients and intercept is large and difficult for interpretation.
- We can reduce them by applying scaling on the data and interpret results easily.
- The standardization can be interpreted as scaling the corresponding slopes, saying how much the target variable changes if we change an independent variable by 1 unit.

Before Scaling:

```
Intercept   -4807.704183
carat        8882.947011
cut           110.298698
color         278.253839
clarity       440.212687
depth          29.259797
table         -12.872217
x           -1244.454006
y            1415.009369
z            -892.235930
dtype: float64
```

After Scaling:

```
Intercept   -1.016982e-16
carat        1.179354e+00
cut          3.520992e-02
color        1.367900e-01
clarity      2.093647e-01
depth        1.021814e-02
table       -7.999050e-03
x           -4.028207e-01
y            4.548472e-01
z           -1.784779e-01
dtype: float64
```

- The various scaling techniques are Min-Max scaler, StandardScaler (Z-Score) and Log Transformation.
- Applying Z-Score scaling on the data, which has property of standard normal

distribution with mean as 0 and standard deviation as 1. It can be seen below as,

	carat	cut	color	clarity	depth	table	x	y	z	price
22114	-0.983742	-0.826474	-0.819083	-0.034626	0.537882	1.194109	-1.176262	-1.159264	-1.121661	-0.923160
2275	-1.070676	0.981336	0.941779	-0.034626	-0.453630	-1.127570	-1.229747	-1.275960	-1.280335	-0.834576
19183	-0.636006	0.981336	-0.232129	-0.641330	0.620508	-0.198899	-0.570093	-0.611693	-0.530240	-0.720311
5030	0.668006	-1.730378	0.941779	-1.248033	1.281516	-0.663234	0.713557	0.761724	0.883399	0.094835
25414	0.494138	0.077431	0.354825	-1.248033	-0.536256	2.122780	0.722471	0.680935	0.638176	0.092527

1.3. Encode the data (having string values) for Modelling. Data Split: Split the data into test and train (70:30). Apply Linear regression. Performance Metrics: Check the performance of Predictions on Train and Test sets using Rsquare, RMSE.

- Converting the categorical/object variables using Label Encoding in the order as given for each column as data pre-processing step.

```
CUT :  5
4    10805
3     6886
2     6027
1     2435
0      780
Name: cut, dtype: int64
```

```
COLOR :  7
3    5653
5    4916
4    4723
2    4095
6    3341
1    2765
0    1440
Name: color, dtype: int64
```

```
CLARITY :  8
2    6565
3    6093
1    4564
4    4087
5    2530
6    1839
7     891
0     364
Name: clarity, dtype: int64
```

- Here, we split the data as X (with all columns except the target column) and Y (target column) used for the model prediction.
- The Linear regression used is Ordinary least squares Linear Regression.
- It fits a linear model with coefficients w = (w1, ..., wp)to minimize the residual sum of squares between the observed targets in the dataset, and the targets predicted by the linear approximation.
- Applying Linear Regression on the scaled data.
- We can view the summary of the regression results applied on the data as below,

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.931
Model:                            OLS   Adj. R-squared:                  0.931
Method:                 Least Squares   F-statistic:                 2.838e+04
Date:                Sat, 05 Dec 2020   Prob (F-statistic):               0.00
Time:                        09:05:58   Log-Likelihood:                -1507.5
No. Observations:               18853   AIC:                             3035.
Df Residuals:                   18843   BIC:                             3113.
Df Model:                           9
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept  -1.017e-16      0.002  -5.33e-14      1.000      -0.004       0.004
carat          1.1794      0.011    107.756      0.000       1.158       1.201
cut            0.0352      0.002     15.013      0.000       0.031       0.040
color          0.1368      0.002     67.738      0.000       0.133       0.141
clarity        0.2094      0.002     98.747      0.000       0.205       0.214
depth          0.0102      0.004      2.659      0.008       0.003       0.018
table         -0.0080      0.002     -3.297      0.001      -0.013      -0.003
x             -0.4028      0.040    -10.083      0.000      -0.481      -0.325
y              0.4548      0.039     11.633      0.000       0.378       0.531
z             -0.1785      0.027     -6.529      0.000      -0.232      -0.125
==============================================================================
Omnibus:                     2744.045   Durbin-Watson:                   1.989
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             9116.198
Skew:                           0.738   Prob(JB):                         0.00
Kurtosis:                       6.070   Cond. No.                         56.7
==============================================================================

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

- We can evaluate the model by applying the performance metrics on the model data.
- The Mean square error and Root Mean Square Error of the train data and test are as below,

```
Mean Square Error of Train data set: 0.06870339619004881
Mean Square Error of Test data set: 0.06865988474722368
Root Mean Square Error of Train data set: 0.2621133269981685
Root Mean Square Error of Test data set: 0.26203031264955523
```

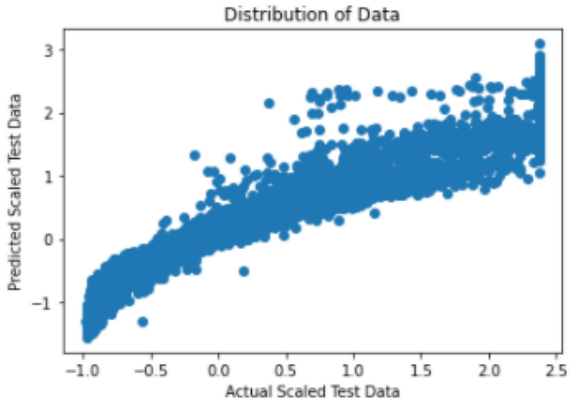
- The R-Squared is the Coefficient of determination which evaluates the scatter of data points around the fitted regression line and recognizes the percentage of variation of the dependent variable.

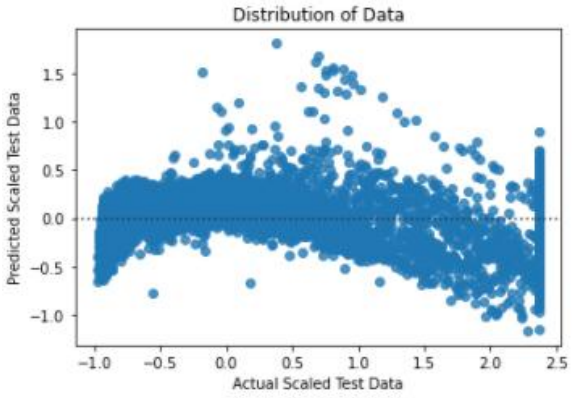
```
The R-Squared of the model : 0.9312966038099512
```

- Adjusted R Square of the model is preferred metrics than R-Squared as it removes the statistical fluke of the model.

```
The Adjusted R-Squared of the model : 0.9312637889415273
```

- We can visualize them in scatterplot or regression plot(residplot).





- Comparing the MSE, RMSE and R-squared value for non-scaled and scaled on the train and test data.

	LinearReg Train before Scaling	LinearReg Test before Scaling	LinearReg Train after Scaling	LinearReg Test after Scaling
RSquared	0.93	0.93	0.93	0.93
MSE	825170.55	828447.68	0.07	0.07
RMSE	908.39	910.19	0.26	0.26

- Comparing the R-Squared and Adjusted R-Squared value for non-scaled and scaled on the train and test data.

	LinearReg before Scaling	LinearReg after Scaling
RSquared	0.93	0.93
Adj RSquared	0.93	0.93

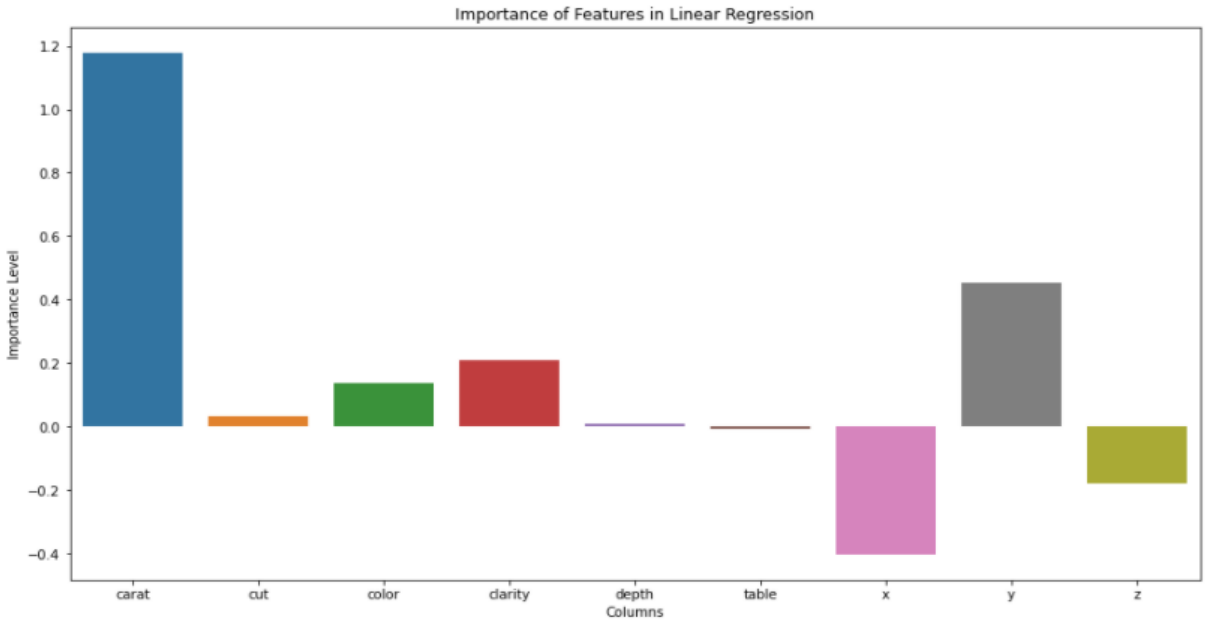
1.4. Inference: Basis on these predictions, what are the business insights and recommendations.

Business Insights:

- While we consider all the given columns without scaling the data and built the linear regression model, we get the high RMSE and R-Squared.
- While we take scaled data and apply linear regression model on them, we get better RMSE and R-Squared.
- Checking the Important columns for this dataset, based on the coefficient of the columns in the descending order after executing the linear regression model on the data.

	Imp
carat	1.179354
y	0.454847
clarity	0.209365
color	0.136790
cut	0.035210
depth	0.010218
table	-0.007999
z	-0.178478
x	-0.402821

- We can that the columns carat, y, clarity, color, cut (in the same order) are highly important in predicting the price of the stones.
- The above list of columns from the top can be used to distinguish between higher profitable stones and lower profitable stones as the company can have better profit share.



- The final Linear Regression equation is

price = (-0.0) * Intercept + (1.18) * carat + (0.04) * cut + (0.14) * color + (0.21) * clarity + (0.01) * depth + (-0.01) * table + (-0.4) * x + (0.45) * y + (-0.18) * z

- When carat increases by 1 unit, price increases by 1.18 units, keeping all other predictors constant.
- Similarly, when clarity increases by 1 unit, price increases by 0.21 units, keeping all other predictors constant.
- There are also some negative co-efficient values, for instance, table has its corresponding co-efficient as -0.01. This implies that the price decreases by 0.01 units, keeping all other predictors constant.

<u>Recommendations:</u>

- We can further improve the model evaluation by checking for the Multi Collinearity condition and reducing them if present.
- The Variance Inflation Factor (VIF) measures how much variance of an estimated regression coefficient increases if the predictors are correlated, also called checking the Multi Collinearity among the independent variables.
- The model whose VIF has values between 5 and 10 is good, values that exceed 10 are regarded as Multi collinearity and those to be handled.

Before Scaling

```
carat ---> 120.15655831946196
cut ---> 9.893839309850376
color ---> 5.54434536823118
clarity ---> 5.539258792510921
depth ---> 1169.3313100189714
table ---> 847.8147336575138
x ---> 9108.8272499798
y ---> 8081.202958845331
z ---> 3017.46990843189
```

After Scaling

```
carat ---> 32.73431042162108
cut ---> 1.51397276833375
color ---> 1.1244314196314955
clarity ---> 1.2645882948891463
depth ---> 4.1803071087980905
table ---> 1.6302940892747588
x ---> 361.12992505065074
y ---> 345.9113861602963
z ---> 205.30986601493032
```

- Checking the VIF for the predicted data with all the given columns, we get the below values.
- Multi collinearity does not affect the prediction or interpretability (RMSE or R-Squared value) but the explainability of the model.
- One way to reduced multi collinearity is removing the highly correlated independent variables.
- We try to reduce the multi collinearity by removing the highly correlated columns one by one in the order and re running the entire linear regression model as over again.
- We get the following VIF values for the predicted test data as below,

After dropping X

```
carat ---> 31.217474547261826
cut ---> 1.5061402501126566
color ---> 1.1242354405100066
clarity ---> 1.2513362090898155
depth ---> 3.8945439293363835
table ---> 1.604578365216172
y ---> 183.19086245774272
z ---> 180.60553551441245
```

After dropping X and Y

```
carat ---> 29.540404574658258
cut ---> 1.5043324754970284
color ---> 1.1234762111275292
clarity ---> 1.2510750833008915
depth ---> 1.443259519626005
table ---> 1.6011207872861142
z ---> 30.026315894745313
```

After dropping X, Y and Z

```
carat ---> 1.295262979648412
cut ---> 1.4904881308096702
color ---> 1.116609825133409
clarity ---> 1.1914965431450795
depth ---> 1.3187649255257503
table ---> 1.5739422281720268
```

- The R-Squared and Adjusted R-Squared values remain the same even after dropping the highly correlated columns, can be seen below.

	LR before Scaling	LR after Scaling	After Dropping X	After Dropping X,Y	After Dropping X,Y,Depth
RSquared	0.93	0.93	0.93	0.93	0.93
Adj RSquared	0.93	0.93	0.93	0.93	0.93

2. Logistic Regression and LDA

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.

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.

- The data is read from the *'Holiday_Package'* dataset. We can check the columns as below.

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

- The shape of the dataset is

```
(872, 7)
```

- The information of the dataset can be read as

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 872 entries, 0 to 871
Data columns (total 7 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Holliday_Package   872 non-null    object
 1   Salary             872 non-null    int64
 2   age                872 non-null    int64
 3   educ               872 non-null    int64
 4   no_young_children  872 non-null    int64
 5   no_older_children  872 non-null    int64
 6   foreign            872 non-null    object
dtypes: int64(5), object(2)
memory usage: 47.8+ KB
```

- The summary of the data description is as seen below,

	Holliday_Package	Salary	age	educ	no_young_children	no_older_children	foreign
count	872	872.000000	872.000000	872.000000	872.000000	872.000000	872
unique	2	NaN	NaN	NaN	NaN	NaN	2
top	no	NaN	NaN	NaN	NaN	NaN	no
freq	471	NaN	NaN	NaN	NaN	NaN	656
mean	NaN	47729.172018	39.955275	9.307339	0.311927	0.982798	NaN
std	NaN	23418.668531	10.551675	3.036259	0.612870	1.086786	NaN
min	NaN	1322.000000	20.000000	1.000000	0.000000	0.000000	NaN
25%	NaN	35324.000000	32.000000	8.000000	0.000000	0.000000	NaN
50%	NaN	41903.500000	39.000000	9.000000	0.000000	1.000000	NaN
75%	NaN	53469.500000	48.000000	12.000000	0.000000	2.000000	NaN
max	NaN	236961.000000	62.000000	21.000000	3.000000	6.000000	NaN

- The null values are checked and we find no null values present in that dataset.

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

- The duplicates in the dataset are checked and we find no duplicates present in them.

Holliday_Package	Salary	age	educ	no_young_children	no_older_children	foreign

- Checking the categorical columns in the dataset.

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

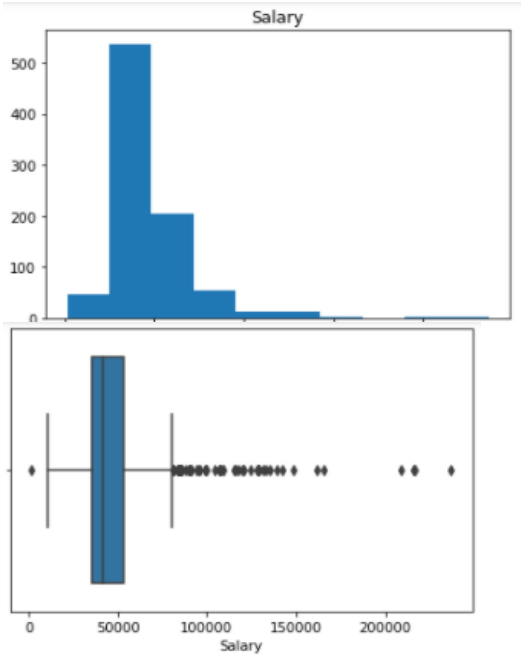
```
FOREIGN :  2
yes    216
no     656
Name: foreign, dtype: int64
```

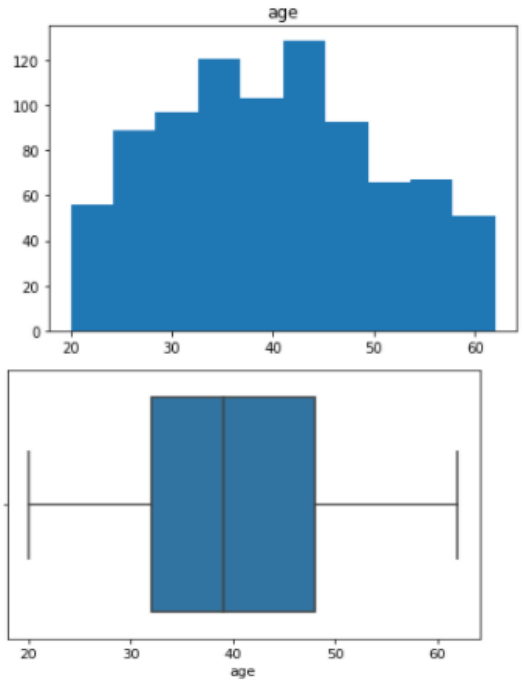
```
NO_OLDER_CHILDREN :  7
6      2
5      2
4     14
3     55
1    198
2    208
0    393
Name: no_older_children, dtype: int64
```

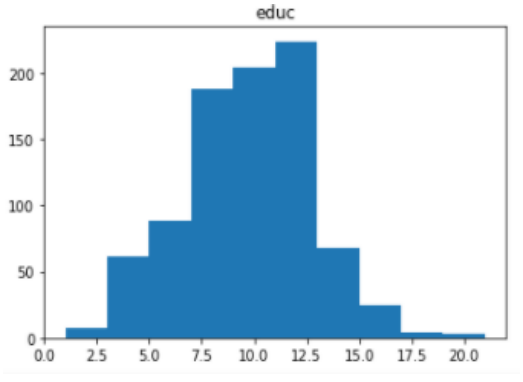
```
NO_YOUNG_CHILDREN :  4
3      5
2     55
1    147
0    665
Name: no_young_children, dtype: int64.
```

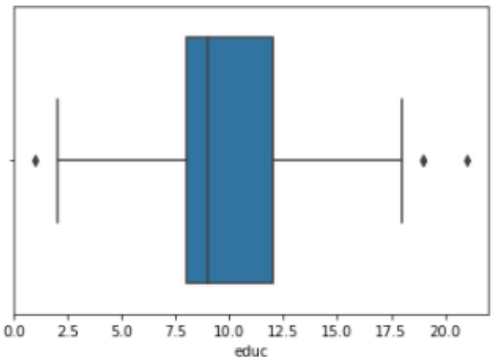
UNIVARIATE ANALYSIS:

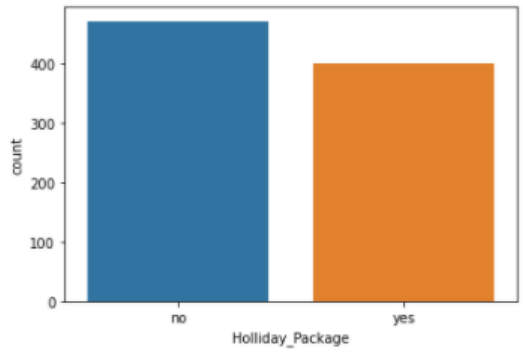
- Plotting the histogram to know the distribution of the numerical columns using histplot.
- Plotting the boxplot from sns package to check the outliers in the columns.
- The salary column is right skewed.
- The age column has symmetric distribution.
- The educ column is slightly skewed.
- The right skewed (or positively skewed) distribution has large occurrence in the left side and few in the right side. Here, mean is greater than the median.
- The left skewed (or negatively skewed) distribution has large occurrence in the right side and few in the left side. Here, mean is less than the median
- The symmetric distribution is the bell-shaped or normal distribution.

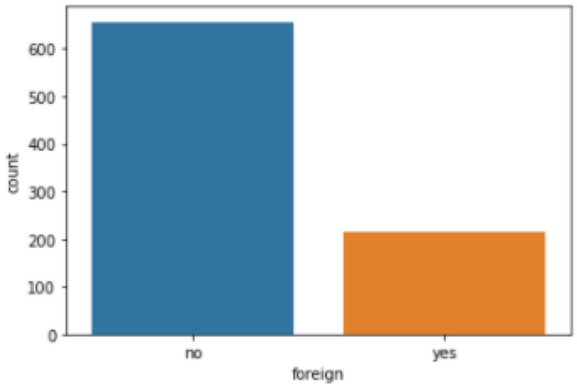

Salary
500
400
300
200
100
0
0
50000
100000
150000
200000
Salary

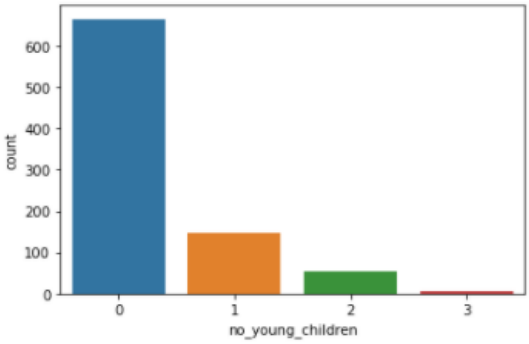

age
120
100
80
60
40
20
0
20
30
40
50
60
20
30
40
50
60
age

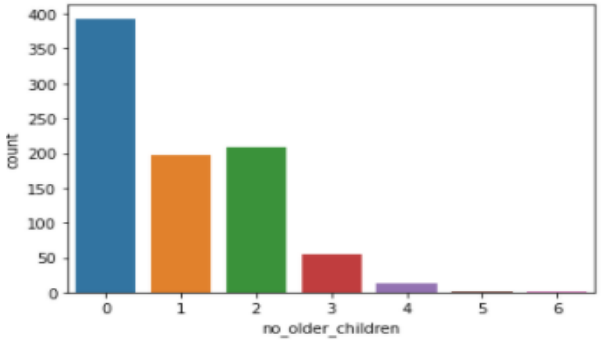

educ
200
150
100
50
0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0


0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
educ





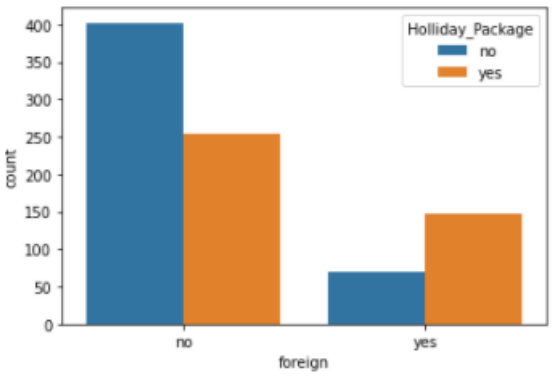




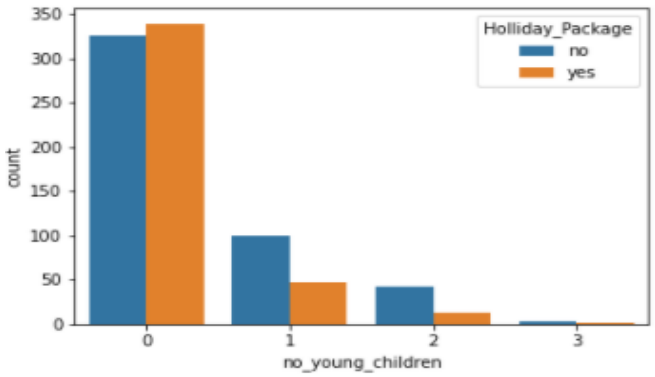
BIVARIATE ANALYSIS:

- We use crosstab that results the frequency table of the factors, used for checking relation between 2 categorical columns.
- Here we use it between the target column and other categorical column.
- We can get the same using the countplot (from sns package).

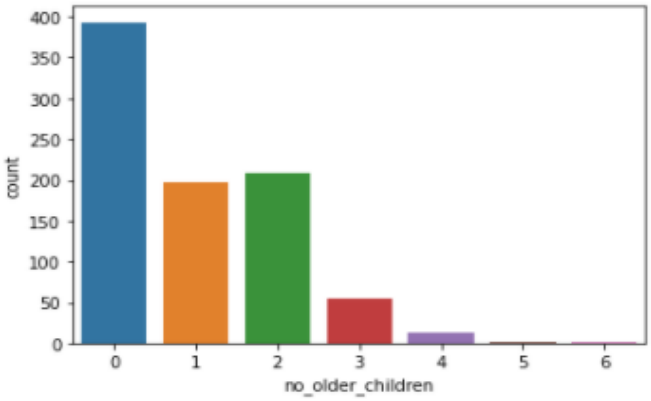
foreign	no	yes
Holliday_Package		
no	402	69
yes	254	147



no_young_children	0	1	2	3
Holliday_Package				
no	326	100	42	3
yes	339	47	13	2



no_older_children	0	1	2	3	4	5	6
Holliday_Package							
no	231	102	102	27	7	2	0
yes	162	96	106	28	7	0	2

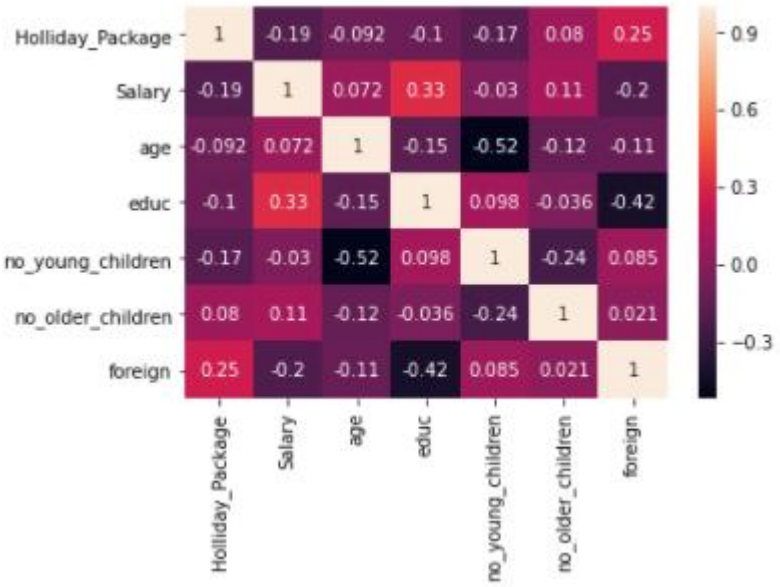


MULTI VARIATE:

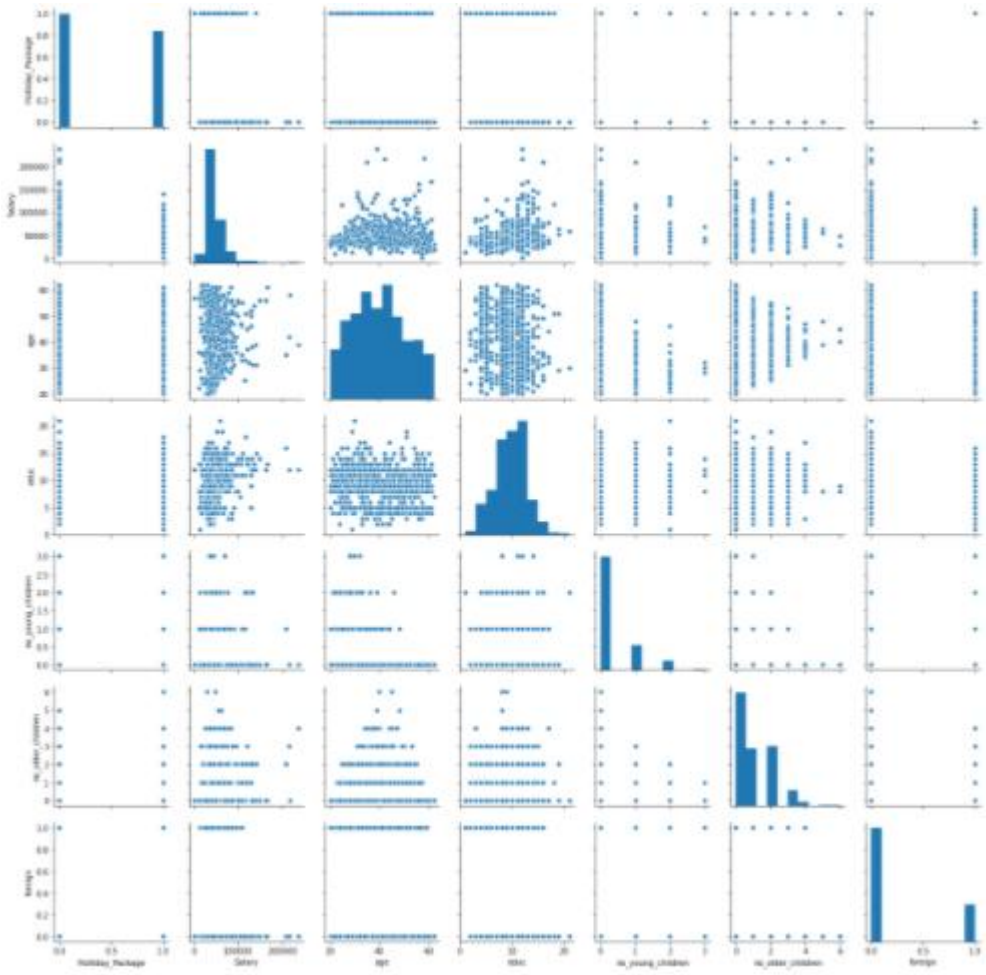
- The correlation across all the numerical or continuous variables can be found using corr() function in a matrix form.

	Holliday_Package	Salary	age	educ	no_young_children	no_older_children	foreign
Holliday_Package	1.000000	-0.185694	-0.092311	-0.102552	-0.173115	0.080286	0.254096
Salary	-0.185694	1.000000	0.071709	0.326540	-0.029664	0.113772	-0.201043
age	-0.092311	0.071709	1.000000	-0.149294	-0.519093	-0.116205	-0.107148
educ	-0.102552	0.326540	-0.149294	1.000000	0.098350	-0.036321	-0.419678
no_young_children	-0.173115	-0.029664	-0.519093	0.098350	1.000000	-0.238428	0.085111
no_older_children	0.080286	0.113772	-0.116205	-0.036321	-0.238428	1.000000	0.021317
foreign	0.254096	-0.201043	-0.107148	-0.419678	0.085111	0.021317	1.000000

- To indicate and visualize the clusters within the data using the heatmap (from sns package).



- The pairplot (from sns package) is used for visualizing the pair wise relationship across entire dataframe.



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).

- We encode the object type columns to categorical columns using the pandas categorical function.

```
0    471
1    401
Name: Holliday_Package, dtype: int64
```

```
0    656
1    216
Name: foreign, dtype: int64
```

- Apart from the categorical converted columns, we have 3 other categorical columns whose data type is int64.

```
0    665
1    147
2     55
3      5
Name: no_young_children, dtype: int64
```

```
0    393
2    208
1    198
3     55
4     14
6      2
5      2
Name: no_older_children, dtype: int64
```

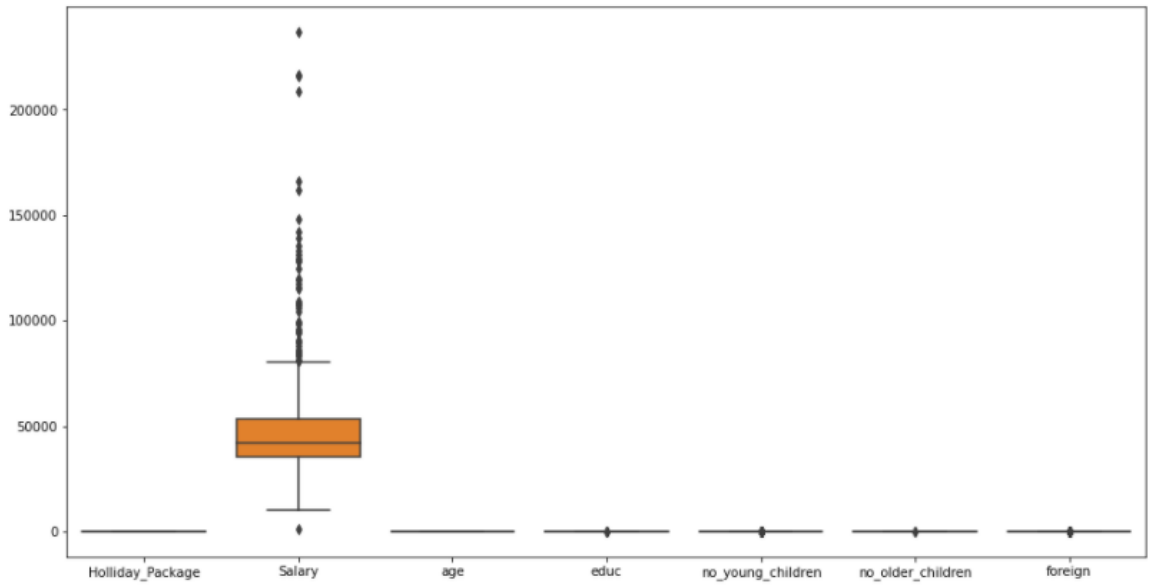
```
8     157
12    124
9     114
11    100
10     90
5      67
4      50
13     43
7      31
14     25
6      21
15     15
3      11
16     10
2       6
17      3
19      2
21      1
18      1
1       1
Name: educ, dtype: int64
```

- As a Data Preprocessing step, we need check for outliers and remove them if present.

```
Shape of the data without outlier treatment : (466, 7)
Shape of the original data  : (872, 7)
```

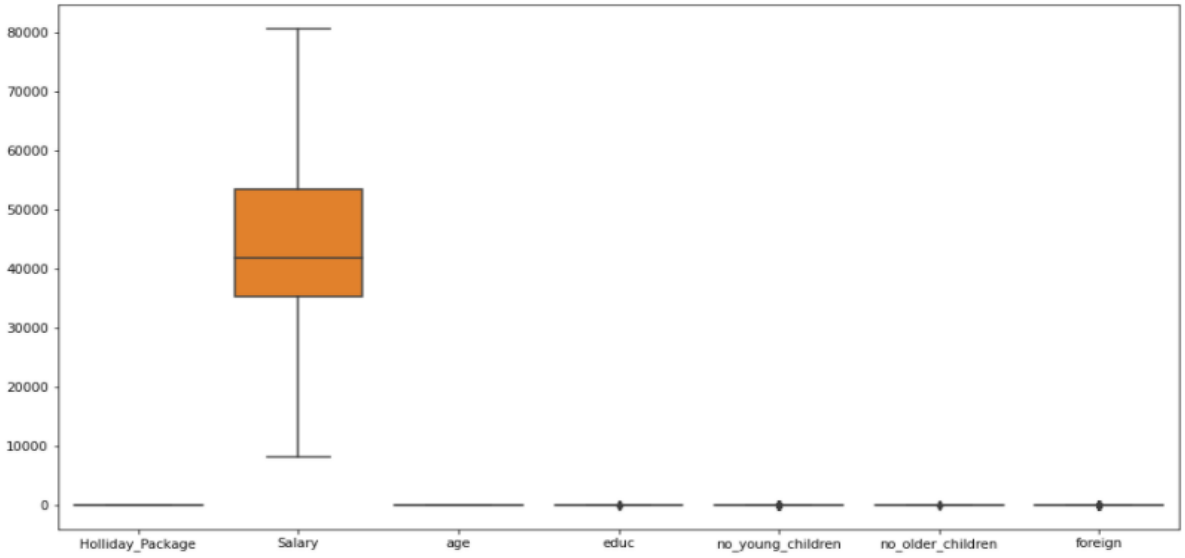
- The Logistic Regression model is affected by the presence of outliers, hence they must be handled.
- We can know the outliers in the data, while we club all the variables together as they depend on each other and project the outliers clearly as seen below.

- We can see that the categorical columns show outliers which are incorrect, so we leave them and handle only the outliers in the numerical columns.



- For the outlier treatment, we use the IQR method for removing the outliers in the salary and age columns as all other columns are categorical.

```
Shape of the data with outlier treatment : (509, 7)
Shape of the original data  : (872, 7)
```



- Grouping all the columns except the target column into separate variable.
- We split the data into train and test data with 70:30 respectively.
- The Logistic Regression model also called Logit is used in the classification problem for both the Binary and Multi class classifier.
- We apply the Logistic Regression model on the data.

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

- We can use the GridSearchCV from sklearn.model_selection package, to try out different set of values for each parameter and get the best set of parameter.
- It uses cross-validation for the number of times to loop through the predefined hyperparameters and fit our Logistic Regression model on the training set.

```
GridSearchCV(cv=5,
             estimator=LogisticRegression(max_iter=10000, n_jobs=2,
                                          random_state=1),
             n_jobs=-1,
             param_grid={'penalty': ['l2', 'none'],
                         'solver': ['saga', 'lbfgs', 'sag', 'liblinear'],
                         'tol': [0.0001, 1e-05, 0.001, 1e-06]},
             scoring='f1')
```

- The hyper parameters are:
    - Solver: used in the optimization problem.
    - Penalty: used to specify the norm used in the penalization.
    - Tol: used to specify the tolerance for the stopping criteria.
    - Max_iter: used for specifying maximum number of iterations taken for solvers to converge.
- We can get the best estimator with the right set of parameters for this data.

```
LogisticRegression(max_iter=10000, n_jobs=2, random_state=1, solver='liblinear',
                   tol=1e-06)
```

- The Linear Discriminant Analysis model is a classifier with a linear decision boundary, generated by fitting class conditional densities to the data and using Bayes' rule.

```
LinearDiscriminantAnalysis(
    *,
    solver='svd',
    shrinkage=None,
    priors=None,
    n_components=None,
    store_covariance=False,
    tol=0.0001,
)
```

- The parameters with their default values are:
    - Solver: svd – Singular Value Decomposition, lsqr – Least Square solution, eigen –Eigen Value decomposition.
    - Shrinkage: none – no shrinkage, auto – automatic shrinkage, give float between 0 and 1.
    - Priors: class prior probabilities as an array.
    - N_components: number of components for dimensionality reduction.

- Store_covariance:
- Tol: absolute threshold.

- We apply the Linear Discriminant Analysis model on the data with default value of parameters.

```
LinearDiscriminantAnalysis()
```

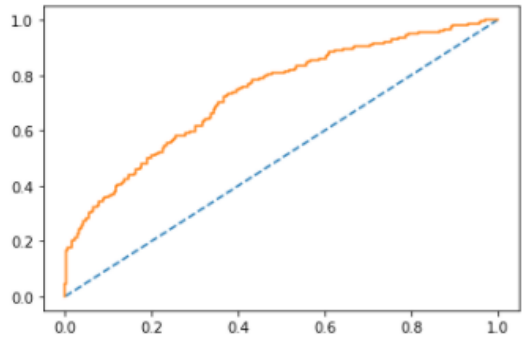
- We can predict the model by using predict function on train or test data.

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.

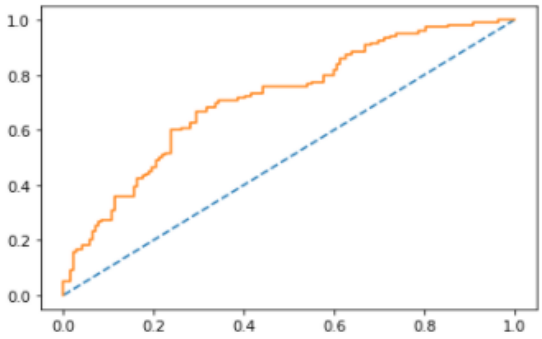
- Applying the performance metrics on the Logistic Regression model with the solver as 'newton-cg' and we get below metrics.

```
Confusion Matrix of the Logistic Regression for Train data:
[[244  85]
 [118 163]]

Confusion Matrix of the Logistic Regression for Test data:
[[108  34]
 [ 58  62]]

Classification Report of the Logistic Regression for Train data:
              precision    recall  f1-score   support

           0       0.67      0.74      0.71       329
           1       0.66      0.58      0.62       281

    accuracy                           0.67       610
   macro avg       0.67      0.66      0.66       610
weighted avg       0.67      0.67      0.66       610

Classification Report of the Logistic Regression for Test data:
              precision    recall  f1-score   support

           0       0.65      0.76      0.70       142
           1       0.65      0.52      0.57       120

    accuracy                           0.65       262
   macro avg       0.65      0.64      0.64       262
weighted avg       0.65      0.65      0.64       262
```

AUC of the Logistic Regression for Train data: 0.733

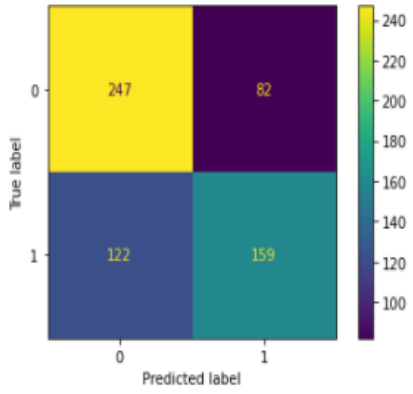


AUC of the Logistic Regression for Test data: 0.715

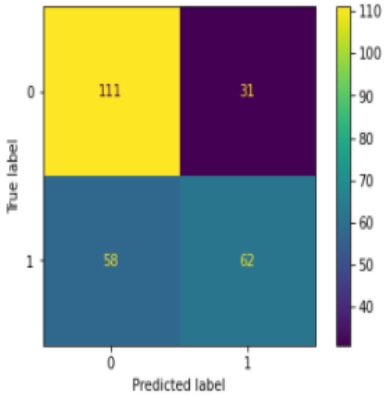


- Checking the metrics for the best parameters from GridSearchCV for the Logistic Regression model.

Confusion Matrix of the Logistic Regression for Train data:



Confusion Matrix of the Logistic Regression for Test data:



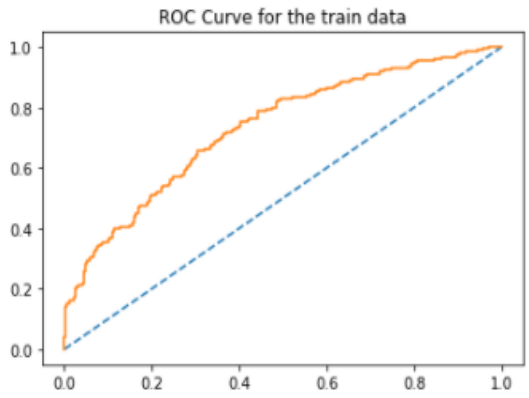
Classification Report of the Logistic Regression for Train data:

	precision	recall	f1-score	support
0	0.67	0.75	0.71	329
1	0.66	0.57	0.61	281
accuracy			0.67	610
macro avg	0.66	0.66	0.66	610
weighted avg	0.66	0.67	0.66	610

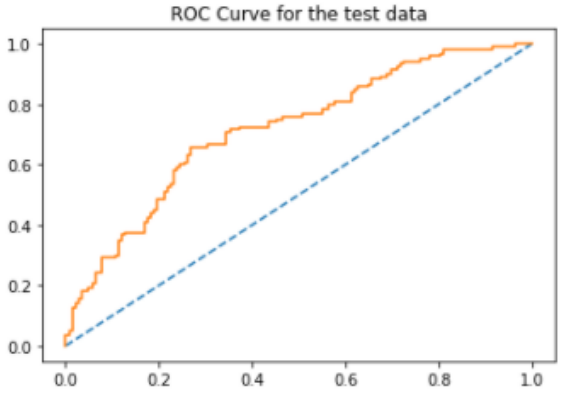
Classification Report of the Logistic Regression for Test data:

	precision	recall	f1-score	support
0	0.66	0.78	0.71	142
1	0.67	0.52	0.58	120
accuracy			0.66	262
macro avg	0.66	0.65	0.65	262
weighted avg	0.66	0.66	0.65	262

AUC of the Logistic Regression for Train data: 0.732

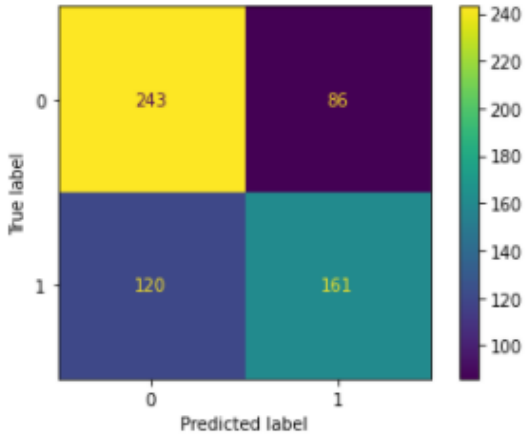


AUC of the Logistic Regression for Test data: 0.716

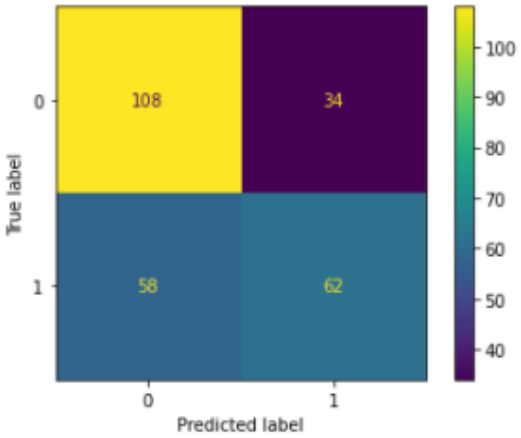


- Checking the performance metrics of the Linear Discriminant Analysis model on the data.

Confusion Matrix of the LDA for Train data:



Confusion Matrix of the LDA for Test data:



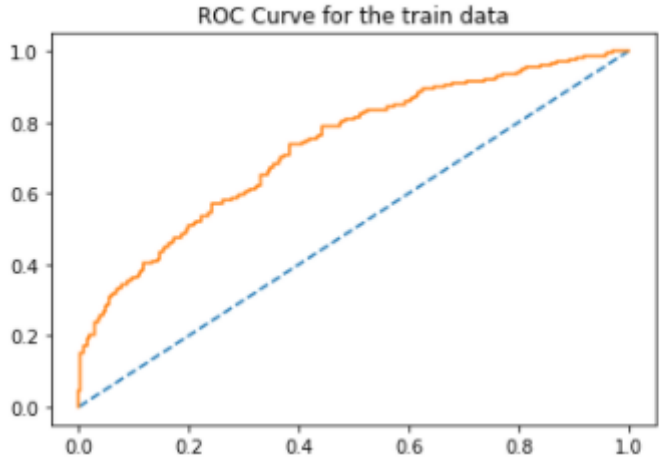
Classification Report of the Linear Discriminant Analysis for Train data:

	precision	recall	f1-score	support
0	0.67	0.74	0.70	329
1	0.65	0.57	0.61	281
accuracy			0.66	610
macro avg	0.66	0.66	0.66	610
weighted avg	0.66	0.66	0.66	610

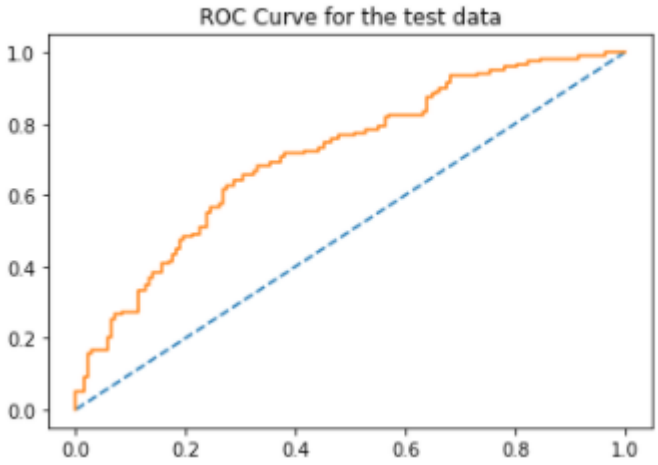
Classification Report of the Linear Discriminant Analysis for Test data:

	precision	recall	f1-score	support
0	0.65	0.76	0.70	142
1	0.65	0.52	0.57	120
accuracy			0.65	262
macro avg	0.65	0.64	0.64	262
weighted avg	0.65	0.65	0.64	262

AUC of the Linear Discriminant Analysis for Train data: 0.731



AUC of the Linear Discriminant Analysis for Test data: 0.714

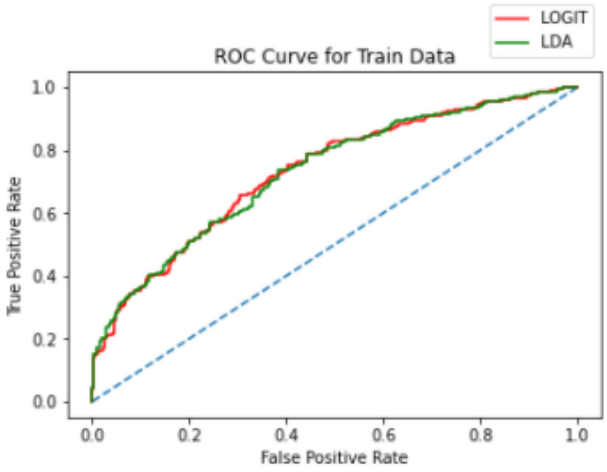


- Comparing the Logistic Regression and Linear Discriminant Analysis models with their performance metrics.

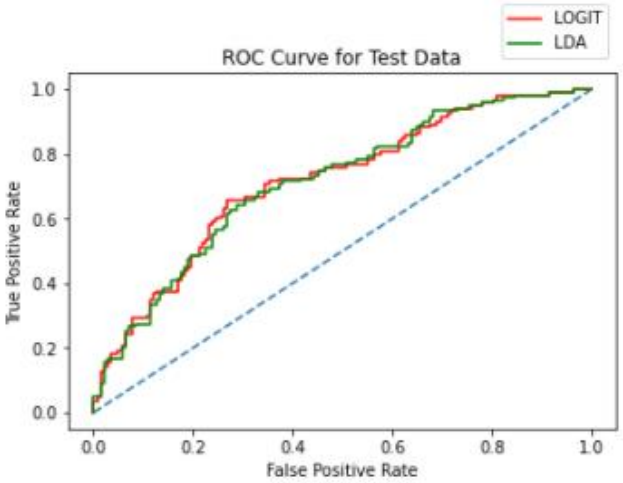
	Logit Train	Logit Test	LDA Train	LDA Test
Accuracy	0.67	0.66	0.66	0.65
AUC Score	0.73	0.72	0.73	0.71
Recall	0.57	0.52	0.57	0.52
Precision	0.66	0.67	0.65	0.65
F1 Score	0.61	0.58	0.61	0.57

- From the above table, we can say that none of the models have been Overfitted and gives better values (not best values though) for their Train and Test data.
- Both the models give similar results for the given problem, but we select the one that is best run or well optimized for the Business Problem.
- We can see that Logit model better than LDA model in terms of Accuracy (by 0.01%), Precision (by 0.01%) and F1 score(by 0.01% test data).
- The Recall seems to have low in both the model.
- But while we try to increase Recall, we decrease the Precision. So try to retain the

Precision with increase in Recall.



- The AUC (Area Under the Curve) is same for Logit and LDA train data with 73%.
- The AUC (Area Under the Curve) is high for Logit, differs by 0.01% than the LDA model.



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

BUSINESS INSIGHTS:

- Checking the Important columns for this dataset, based on the coefficient of the columns in the descending order after executing the linear regression model on the data.

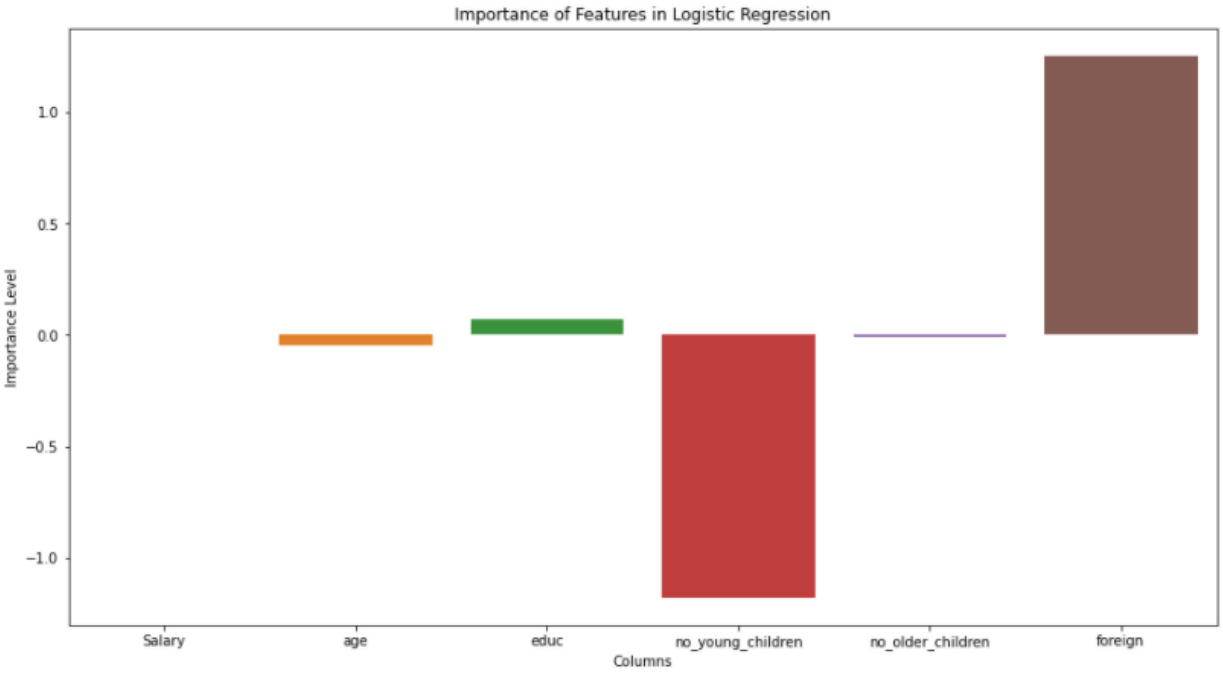
Logistic Regression Model

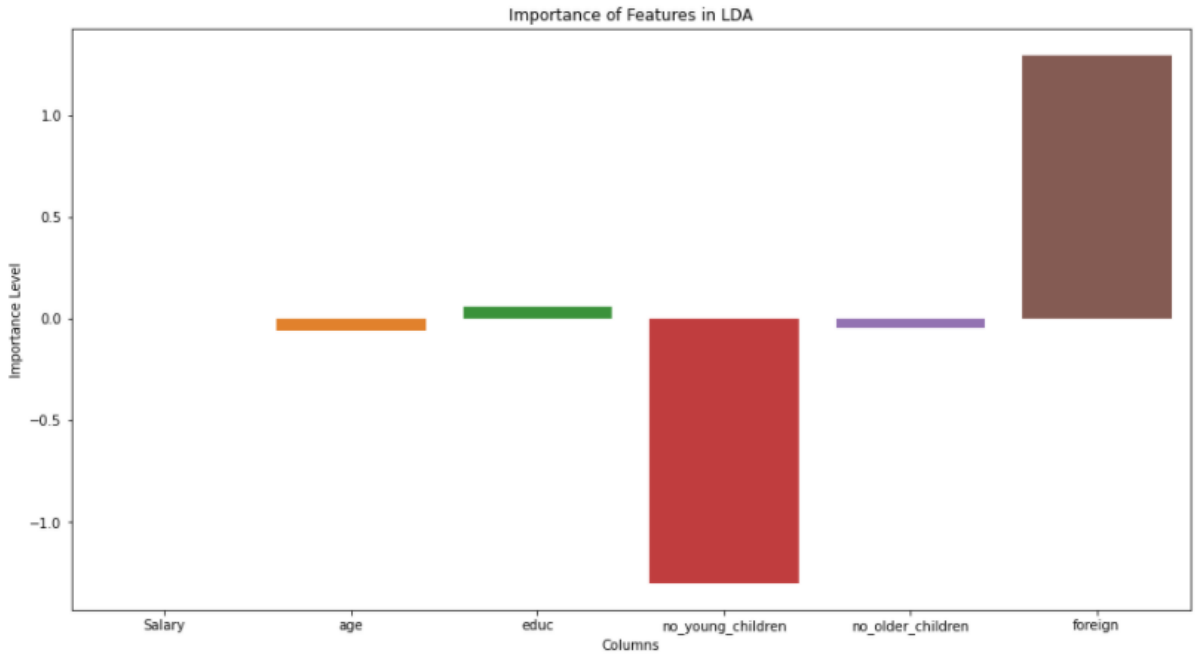
	Imp
foreign	1.256026
educ	0.072284
Salary	-0.000018
no_older_children	-0.008185
age	-0.046505
no_young_children	-1.181626

LDA Model

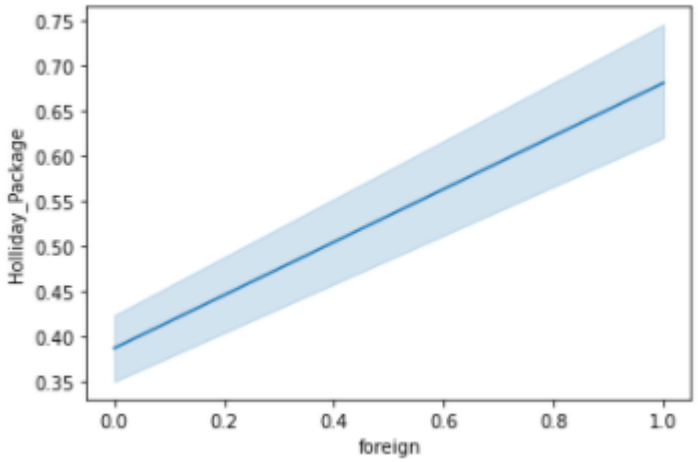
	Imp
foreign	1.295476
educ	0.057748
Salary	-0.000020
no_older_children	-0.046112
age	-0.058829
no_young_children	-1.300687

- In both the Logistic Regression model and Linear Discriminant Analysis model, the order of importance of factors across the columns is same.
- We can see that both the positive and negative correlation of columns with the target column occurs.

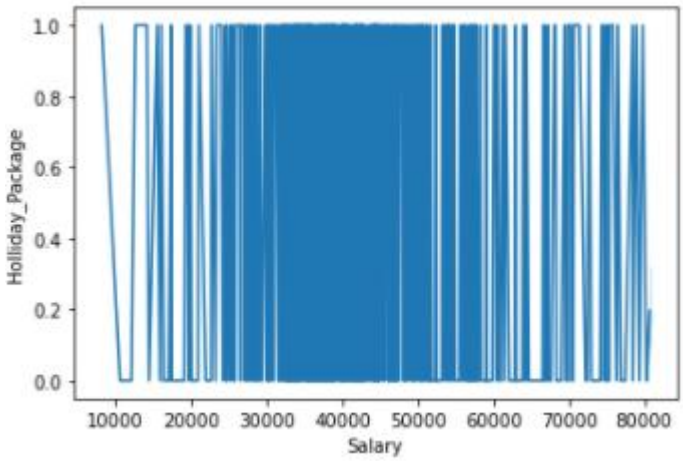




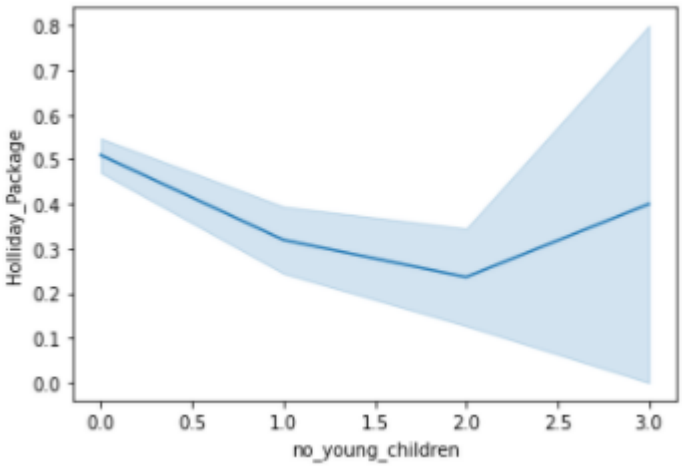
- The column Foreign is more important factor with high correlation with the target column Holliday_Package.



- The Salary column is totally uncorrelated with the target column.



- The column no_young_children is fairly negatively correlated with the target column.



RECOMMENDATIONS:

- We consider the business recommendations based on the Logistic Regression model.
- Using the predicted test data, we can say that out of 262 employees, totally 58 employees opt for the Holiday Package.
- The employees who opt the package, has average salary of around 45000.

	Salary	age	educ	no_young_children	no_older_children	foreign
count	58.000000	58.000000	58.000000	58.000000	58.000000	58.000000
mean	45831.788793	44.068966	9.068966	0.189655	1.155172	0.051724
std	14029.805246	8.303164	2.648151	0.511513	1.472577	0.223404
min	17350.000000	25.000000	2.000000	0.000000	0.000000	0.000000
25%	35850.250000	39.000000	8.000000	0.000000	0.000000	0.000000
50%	44566.000000	44.500000	9.000000	0.000000	1.000000	0.000000
75%	54033.750000	49.000000	11.000000	0.000000	2.000000	0.000000
max	80687.750000	58.000000	16.000000	2.000000	6.000000	1.000000

- The Employee choosing the package has maximum 2 younger children and 6 older children, could be said as the employee having maximum with 6 children.
- The average age of the Employee who opt the package is 44 years old.
- Using the predicted train data, we can say that out of 610 employees, totally 120 employees opt for the Holiday Package.

	Salary	age	educ	no_young_children	no_older_children	foreign
count	120.000000	120.000000	120.000000	120.000000	120.000000	120.000000
mean	49632.845833	41.633333	9.916667	0.375000	0.975000	0.075000
std	16648.114461	9.282615	3.033658	0.710811	1.040867	0.264496
min	8105.750000	22.000000	1.000000	0.000000	0.000000	0.000000
25%	37821.250000	35.000000	8.000000	0.000000	0.000000	0.000000
50%	46087.000000	42.000000	10.000000	0.000000	1.000000	0.000000
75%	58435.750000	47.250000	12.000000	1.000000	2.000000	0.000000
max	80687.750000	61.000000	18.000000	3.000000	4.000000	1.000000

- Here, we can say that the average salary of the employee selecting the package has average Age of 42 years old and average Salary of around 46000.
- The Employee has maximum 7 children whose choosing the package option.
- The Employee who opt the package is a non-foreigner as a majority, according to the data as can see above.
- Some foreigner employee opt the package to travel to their hometowns at this time.