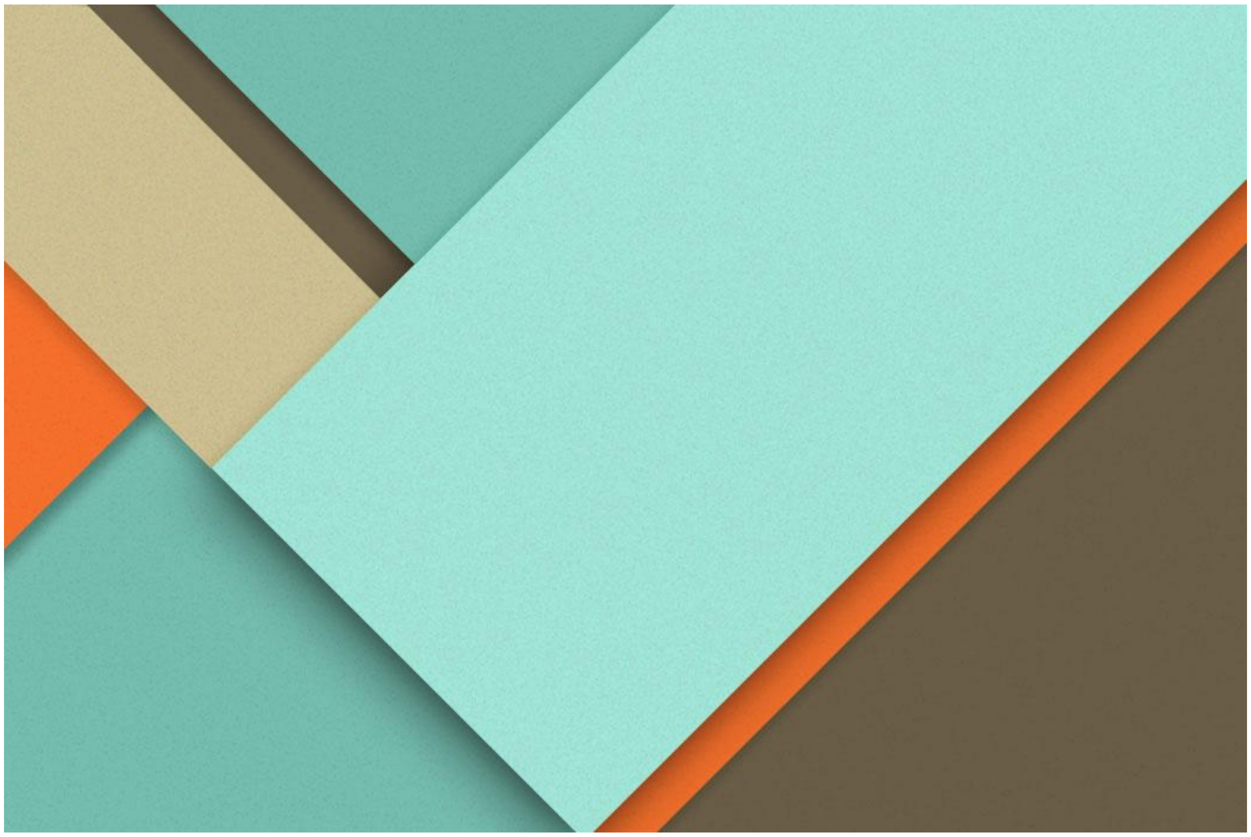


Predictive Modeling

Pradeep Kumar Mishra
PGP-DSBA Online
Jun_B_21
Date: 28:Nov:2021

Content View

List of Tables :

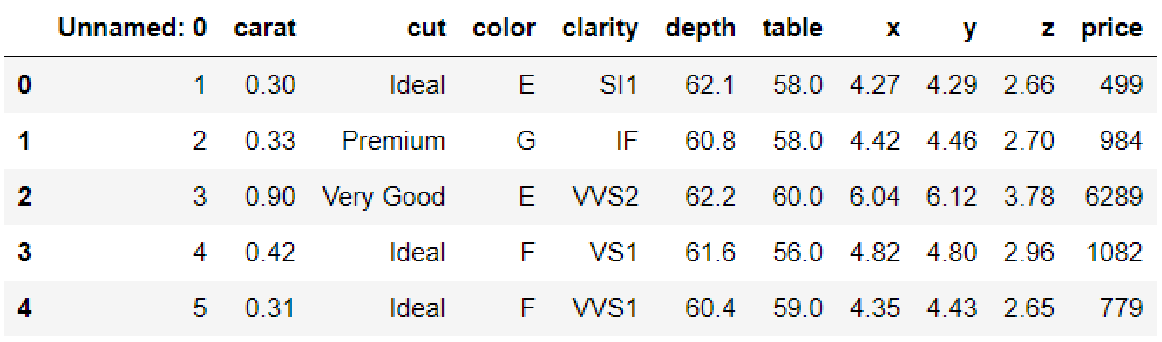
List of Figures :

Problem Statement - 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 basis 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, duplicate values). Perform Univariate and Bivariate Analysis.

Sample of the dataset



	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

Table-01Dataset Sample

```
<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
```

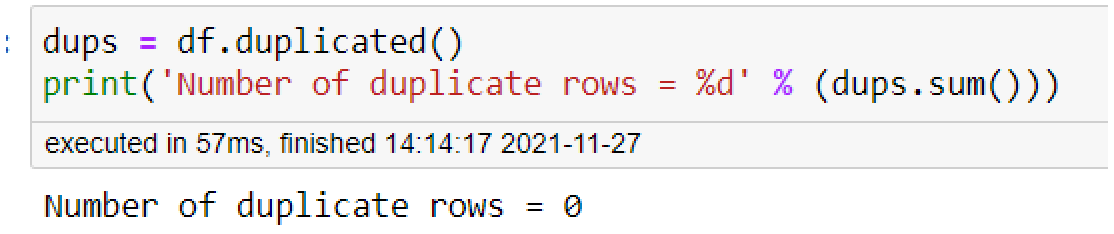
- From the above results we can see that there is some missing value present in the dataset.
- There are a total of 26967 rows and 10 columns in the dataset.
- 6 variables out of 10 is float64 , 1 variable is int64 and 3 variables out 10 are object data types.

missing values :

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

From the above result we can see that there are 697 values missing in the depth variable.

Duplicate values :



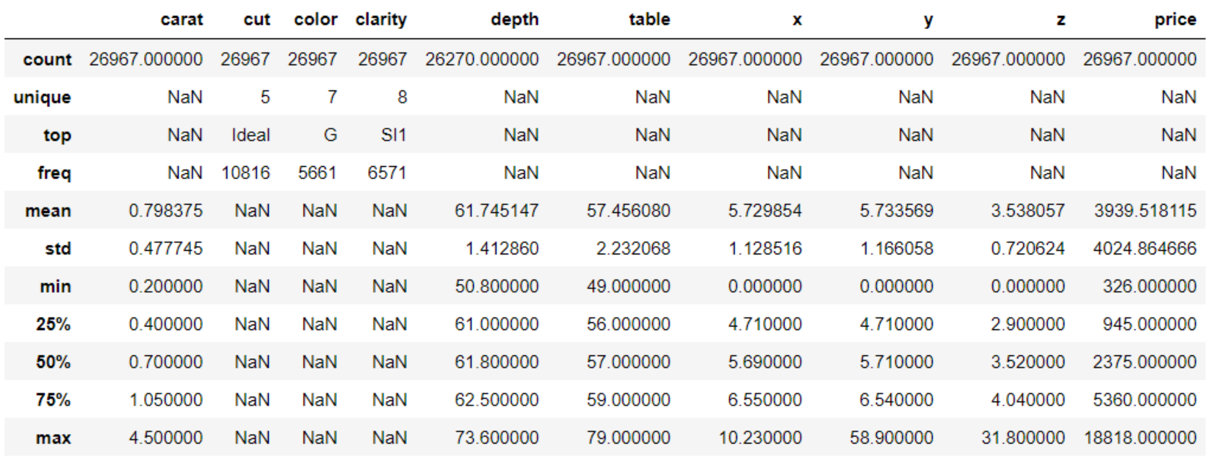
```
dups = df.duplicated()
print('Number of duplicate rows = %d' % (dups.sum()))
```

executed in 57ms, finished 14:14:17 2021-11-27

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

There is no duplicate value in the datasets.

Describe datasets :



	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

Table-02 Describe the data

```
CUT :  5
Fair           781
Good          2441
Very Good     6030
Premium       6899
Ideal        10816
Name: cut, dtype: int64


COLOR :  7
J    1443
I    2771
D    3344
H    4102
F    4729
E    4917
G    5661
Name: color, dtype: int64


CLARITY :  8
I1       365
IF       894
VVS1    1839
VVS2    2531
VS1     4093
SI2     4575
VS2     6099
SI1     6571
Name: clarity, dtype: int64
```

- We have 3 object variables and the remaining are numerical variables.
- In the cut variable there are 5 unique values, most of the data is Ideal and their frequency is 10816.
- In the color variable there are 7 unique values, most of the color is G and their frequency is 5661.
- In the clarity variable there are 8 unique values, most of the data is SI1 and their frequency is 6571.

- Some of the numerical variables have mean and median are almost equal and some of the variables have skewness.

Univariate / Bivariate analysis :

Distplot :

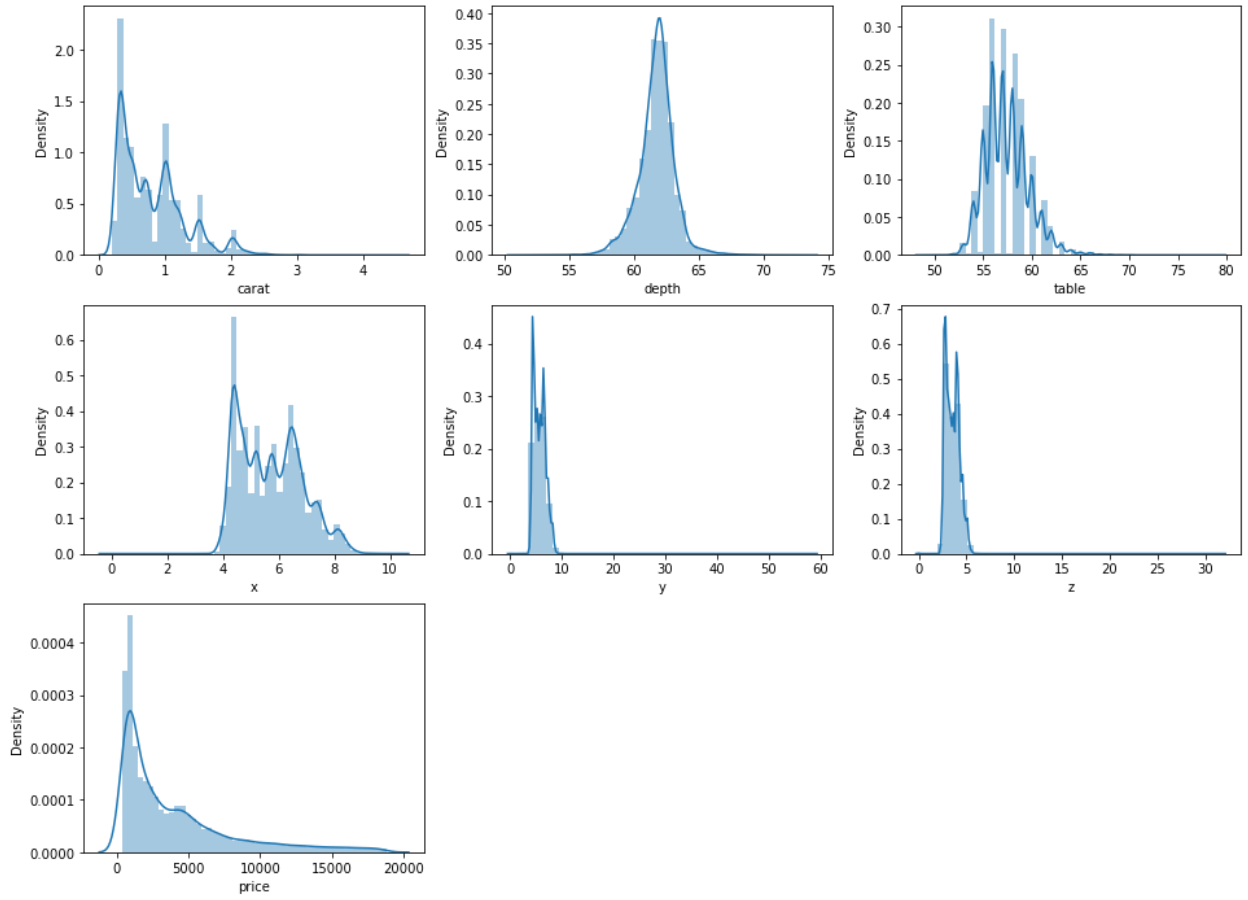


Figure-01

Boxplot :

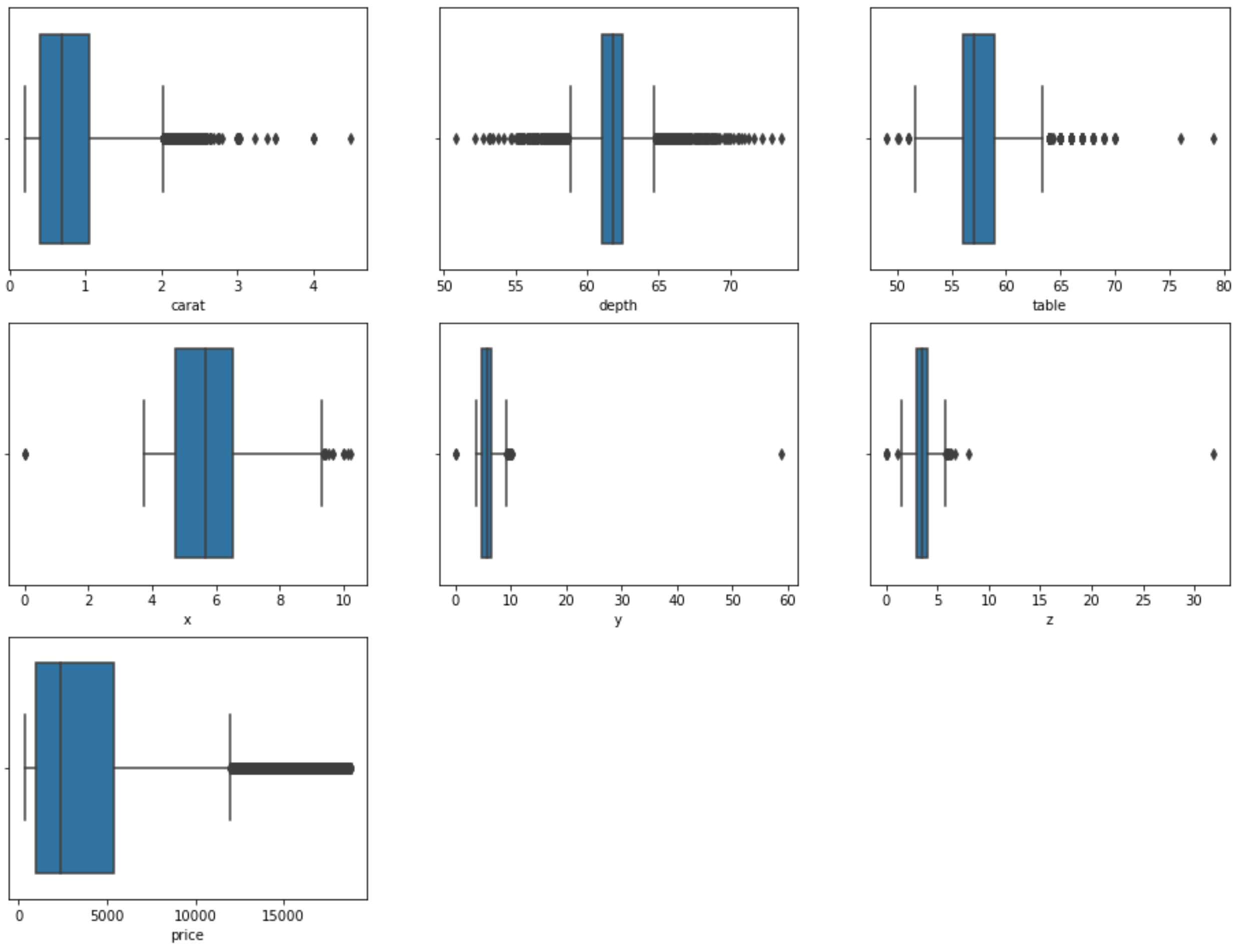


Figure-02

```
carat Skewed:  1.1165
depth Skewed:  -0.0286
table Skewed:  0.7658
x Skewed:  0.388
y Skewed:  3.8502
z Skewed:  2.5683
price Skewed:  1.6185
```

Carat feature :

- Distribution of carat variable seems to right skewed = 1.1165
- In the distplot it seems there are multiple peak points.
- There are large no. of outliers.

Depth feature :

- Distribution of depth variable seems to be left skewed = -0.0289 it is very less no. so we can say depth data is normally distributed.
- There are large no. of outliers in both the direction left and right.

Table feature :

- Distribution of table variable seems to be positively skewed =0.7658
- Boxplot of the table has outliers.

X (Length of the cubic zirconia in mm) feature :

- Distribution of the X (Length of the cubic zirconia in mm) seems positively skewed= 0.388.
- There are outliers.

Y (width of the cubic zirconia in mm) feature:

- Distribution of the Y (width of the cubic zirconia in mm) seems positively skewed= 3.8502.
- There are outliers.

Z (Height of the cubic zirconia in mm) feature :

- Distribution of the Z (Height of the cubic zirconia in mm) seems positively skewed= 2.5683.
- There are outliers.

Price feature :

- Distribution of the price feature seems positively skewed= 1.6185.
- There are outliers.

Countplot :

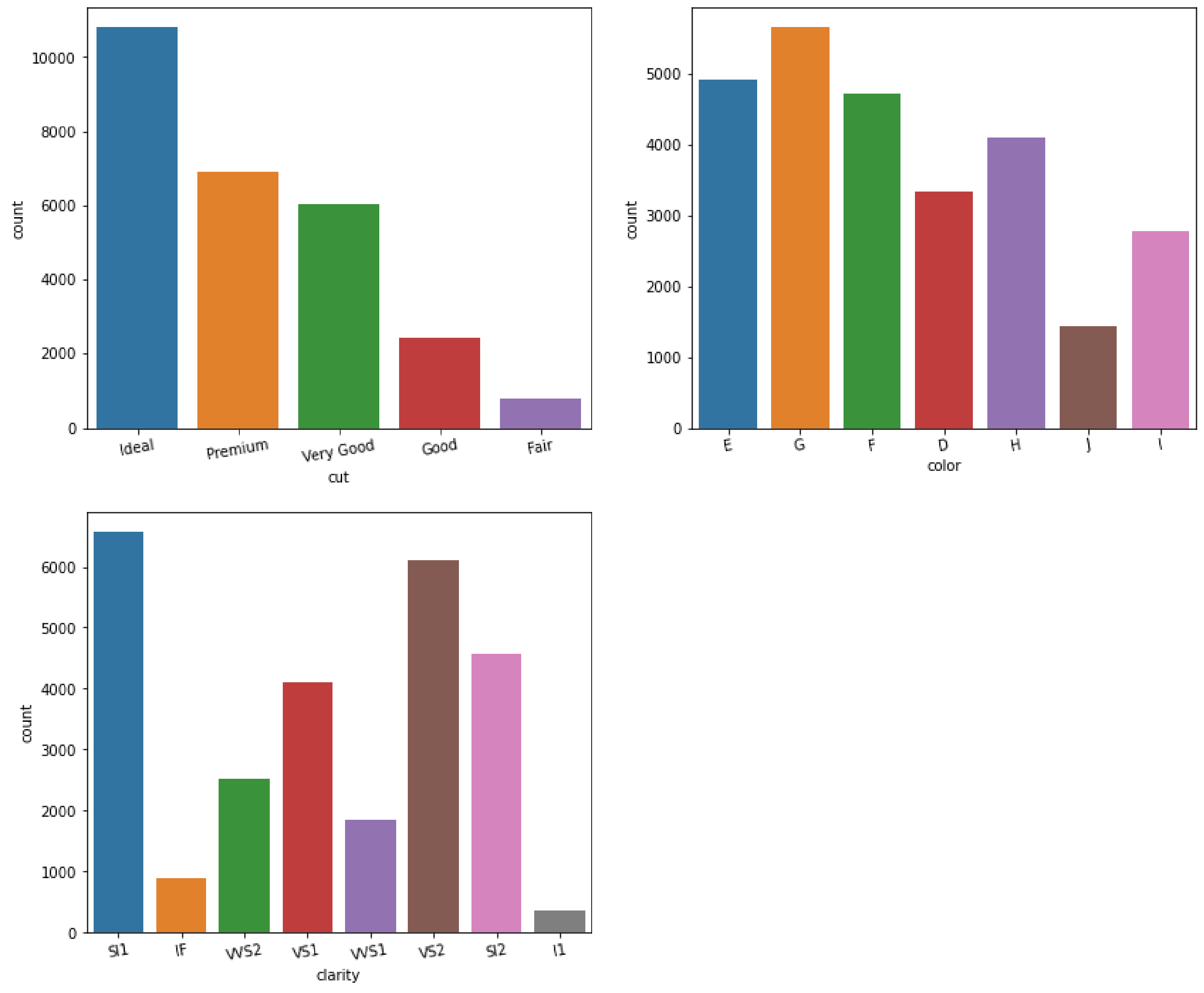


Figure-03

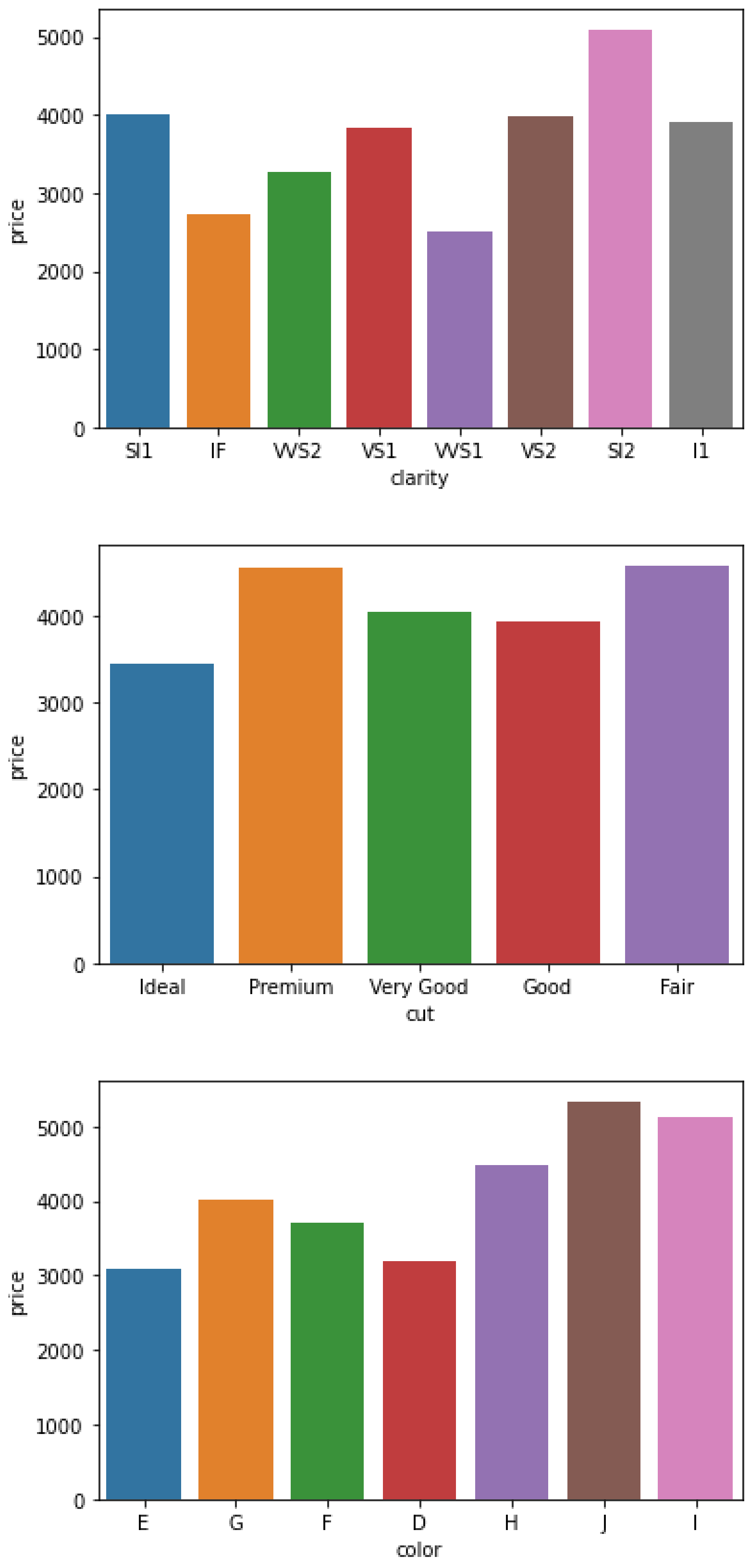

price
5000
4000
3000
2000
1000
0
SI1
IF
VVS2
VS1
VVS1
VS2
SI2
I1
clarity
price
4000
3000
2000
1000
0
Ideal
Premium
Very Good
Good
Fair
cut
price
5000
4000
3000
2000
1000
0
E
G
F
D
H
J
I
color

Figure-04

Cut feature :

- Ideal cut is the most preferred cut for the diamond.
- Ideal cut price is less compared to others.

Color feature :

- G color seems to be the most preferred color.
- J color price is the most expensive.

Clarity feature :

- SI1 seems to be the most preferred clarity.

Pairplot :

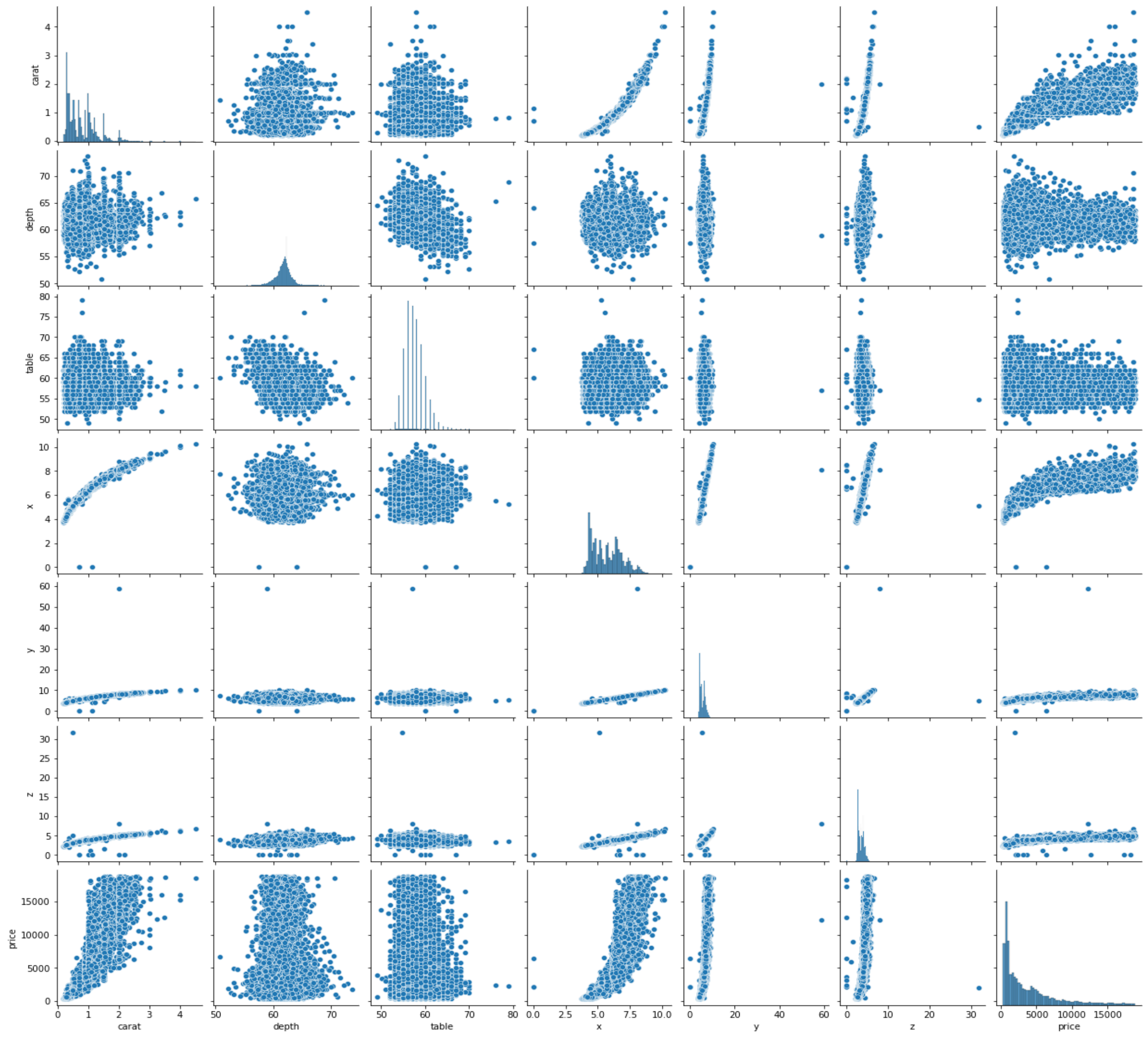
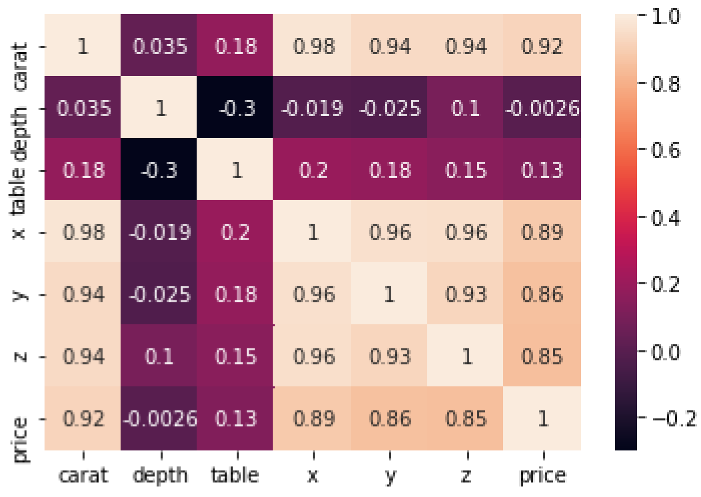


Figure -05

Heatmap :



	carat	depth	table	x	y	z	price
carat	1	0.035	0.18	0.98	0.94	0.94	0.92
depth	0.035	1	-0.3	-0.019	-0.025	0.1	-0.0026
table	0.18	-0.3	1	0.2	0.18	0.15	0.13
x	0.98	-0.019	0.2	1	0.96	0.96	0.89
y	0.94	-0.025	0.18	0.96	1	0.93	0.86
z	0.94	0.1	0.15	0.96	0.93	1	0.85
price	0.92	-0.0026	0.13	0.89	0.86	0.85	1

Figure-06

After seeing the pairplot and heatmap we can say that there is presence of multicollinearity in the datasets.

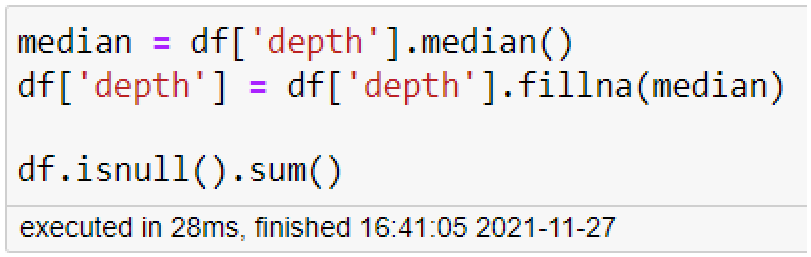
1.2 Impute null values if present, also check for the values which are equal to zero. Do they have any meaning or do we need to change them or drop them? Check for the possibility of combining the sub levels of a ordinal variables and take actions accordingly. Explain why you are combining these sub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

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

```
carat       0.000000
cut         0.000000
color       0.000000
clarity     0.000000
depth       0.025846
table       0.000000
x           0.000000
y           0.000000
z           0.000000
price       0.000000
dtype: float64
```

There are null values in depth variables and this is less than 5% so we can fill them using median imputation.

After the median imputation we see below there are no null values now.



```
median = df['depth'].median()
df['depth'] = df['depth'].fillna(median)

df.isnull().sum()
```

executed in 28ms, finished 16:41:05 2021-11-27

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

Checking if there is value that is “0” :

```
df.all()
```

```
executed in 17ms, finished 16:41:15 2021-11-27
```

```
carat         True
cut           True
color         True
clarity       True
depth         True
table         True
x            False
y            False
z            False
price         True
dtype: bool
```

	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
6215	0.71	Good	F	SI2	64.1	60.0	0.00	0.00	0.0	2130
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

Table-03

- I found that there is some row containing zero that has an X,Y,Z feature and has 0 values.
- We have to drop the zero value containing row.

Remove outliers :

- As we know there are outliers in the datasets so we have to impute the outliers.

- After the imputation of outliers we have seen there is now outliers now.

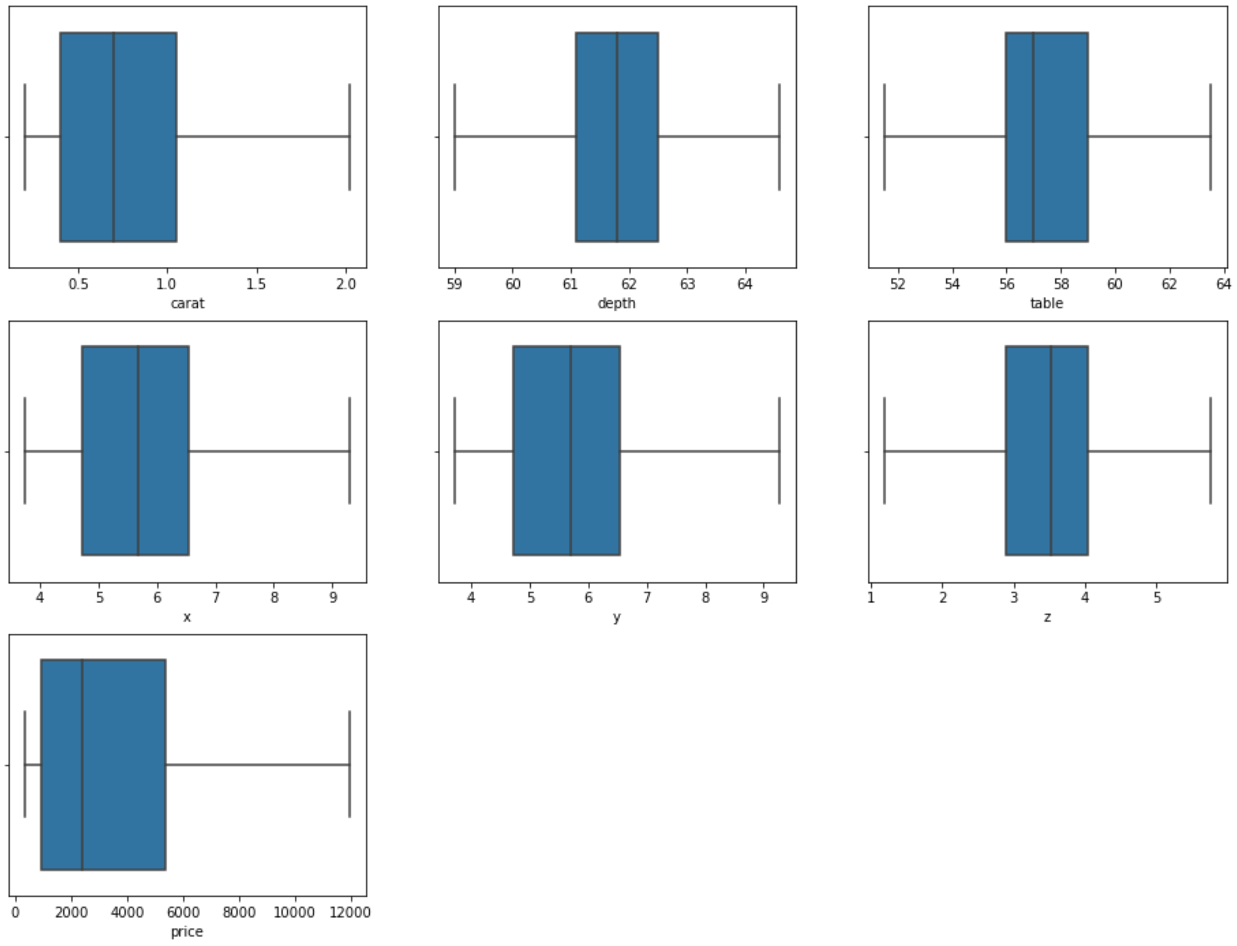
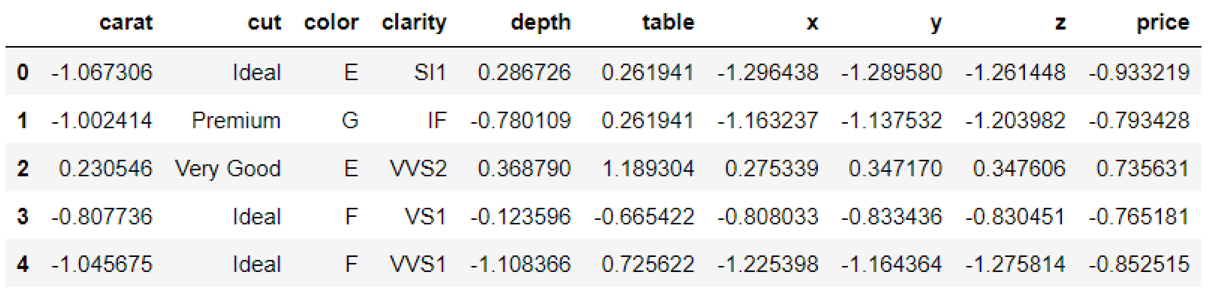


Figure-07

Scaling :

- Scaling is required for the linear regression because it reduces the multicollinearity.
- I use StandardScaler for scaling the datasets.



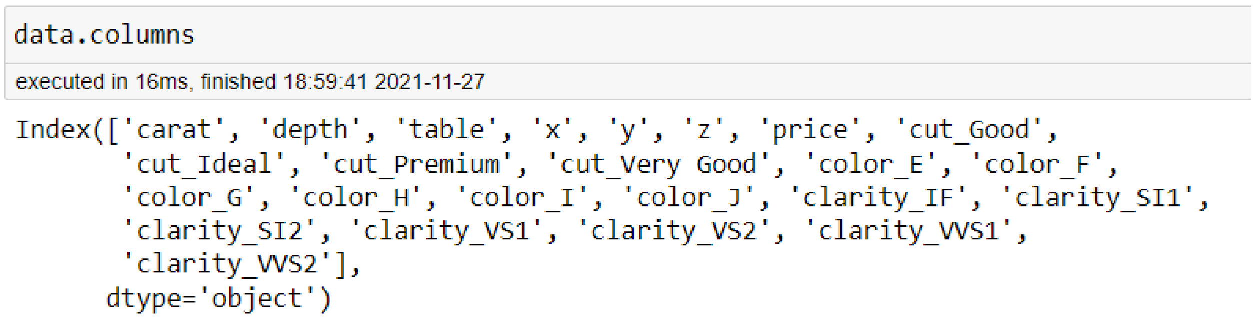
	carat	cut	color	clarity	depth	table	x	y	z	price
0	-1.067306	Ideal	E	SI1	0.286726	0.261941	-1.296438	-1.289580	-1.261448	-0.933219
1	-1.002414	Premium	G	IF	-0.780109	0.261941	-1.163237	-1.137532	-1.203982	-0.793428
2	0.230546	Very Good	E	VVS2	0.368790	1.189304	0.275339	0.347170	0.347606	0.735631
3	-0.807736	Ideal	F	VS1	-0.123596	-0.665422	-0.808033	-0.833436	-0.830451	-0.765181
4	-1.045675	Ideal	F	VVS1	-1.108366	0.725622	-1.225398	-1.164364	-1.275814	-0.852515

Table-04

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 string variables :

After encoding the datasets then shape is 5X24

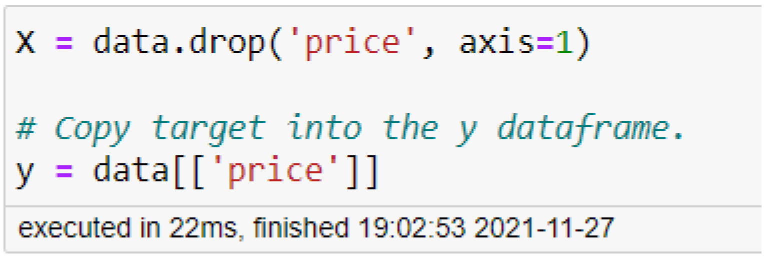


```
data.columns
```

executed in 16ms, finished 18:59:41 2021-11-27

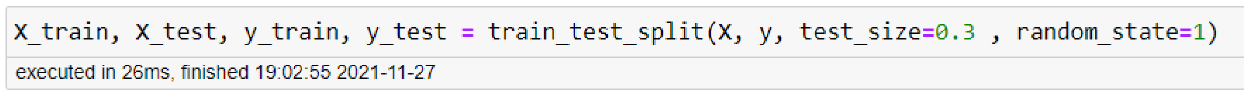
```
Index(['carat', 'depth', 'table', 'x', 'y', 'z', 'price', 'cut_Good',
       'cut_Ideal', 'cut_Premium', 'cut_Very Good', 'color_E', 'color_F',
       'color_G', 'color_H', 'color_I', 'color_J', 'clarity_IF', 'clarity_SI1',
       'clarity_SI2', 'clarity_VS1', 'clarity_VS2', 'clarity_VVS1',
       'clarity_VVS2'],
      dtype='object')
```

Dropping target variables for the model and split the data 70:30 ratio for the train and test respectively.



```
X = data.drop('price', axis=1)

# Copy target into the y dataframe.
y = data[['price']]
```

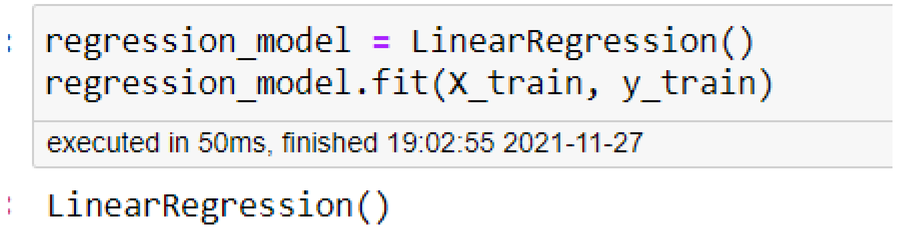
executed in 22ms, finished 19:02:53 2021-11-27



```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3 , random_state=1)
```

executed in 26ms, finished 19:02:55 2021-11-27

Linear Regression Model :

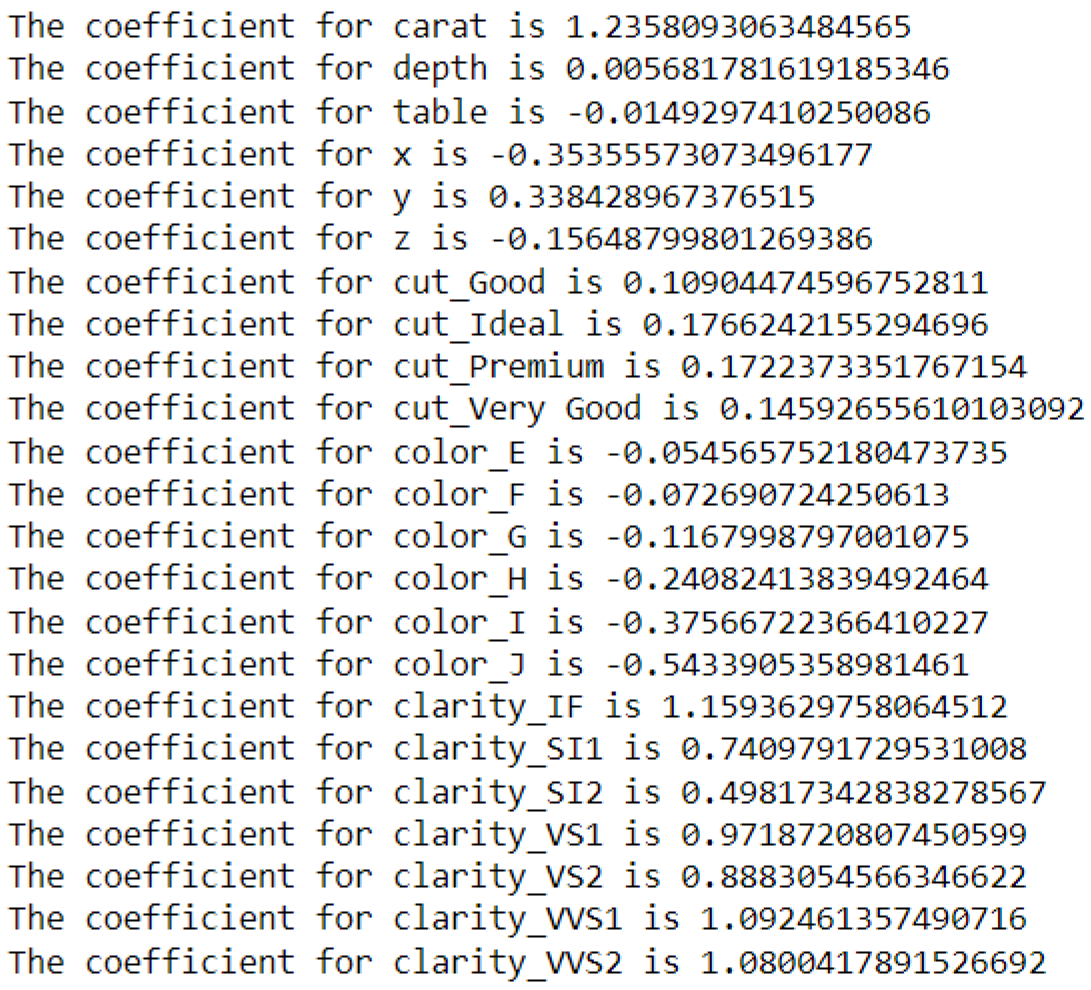


```
regression_model = LinearRegression()
regression_model.fit(X_train, y_train)
```

executed in 50ms, finished 19:02:55 2021-11-27

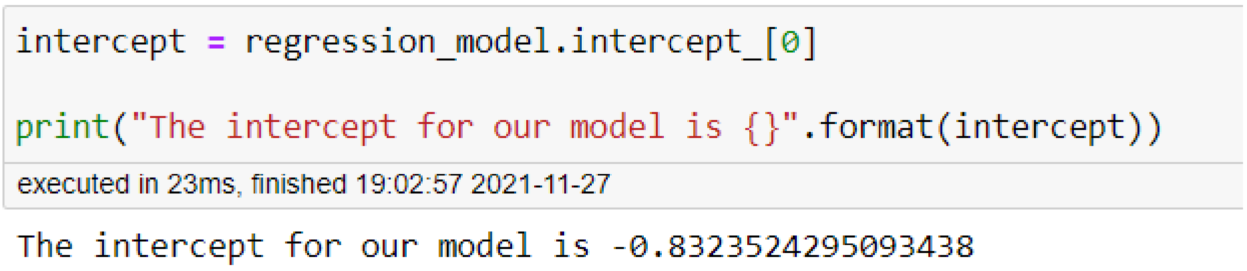
```
LinearRegression()
```

Coefficient of variables :



```
The coefficient for carat is 1.2358093063484565
The coefficient for depth is 0.005681781619185346
The coefficient for table is -0.014929741025008 6
The coefficient for x is -0.3535573073496177
The coefficient for y is 0.338428967376515
The coefficient for z is -0.15648799801269386
The coefficient for cut_Good is 0.10904474596752811
The coefficient for cut_Ideal is 0.1766242155294696
The coefficient for cut_Premium is 0.1722373351767154
The coefficient for cut_Very Good is 0.14592655610103092
The coefficient for color_E is -0.05456575218047373 5
The coefficient for color_F is -0.07269072425061 3
The coefficient for color_G is -0.1167998797001075
The coefficient for color_H is -0.24082413839492464
The coefficient for color_I is -0.3756672236641022 7
The coefficient for color_J is -0.5433905358981461
The coefficient for clarity_IF is 1.1593629758064512
The coefficient for clarity_SI1 is 0.7409791729531008
The coefficient for clarity_SI2 is 0.49817342838278567
The coefficient for clarity_VS1 is 0.9718720807450599
The coefficient for clarity_VS2 is 0.8883054566346622
The coefficient for clarity_VVS1 is 1.092461357490716
The coefficient for clarity_VVS2 is 1.0800417891526692
```

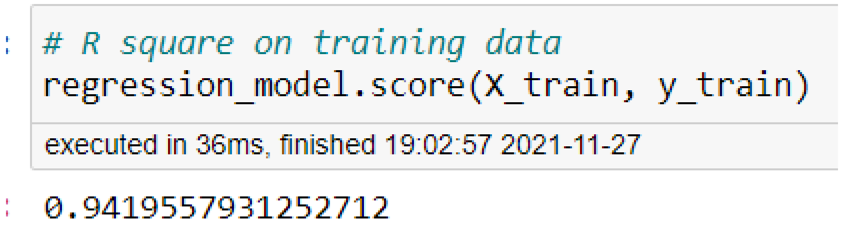
Intercept of model :



```
intercept = regression_model.intercept_[0]

print("The intercept for our model is {}".format(intercept))
```

executed in 23ms, finished 19:02:57 2021-11-27

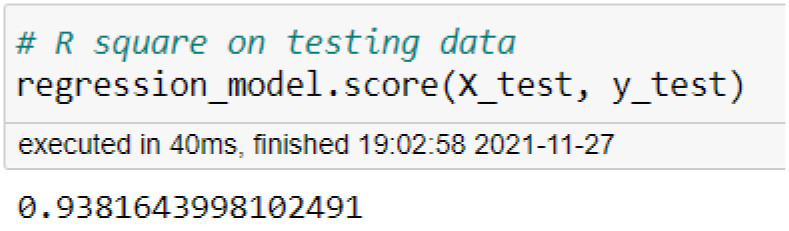
```
The intercept for our model is -0.8323524295093438
```



```
# R square on training data
regression_model.score(X_train, y_train)
```

executed in 36ms, finished 19:02:57 2021-11-27

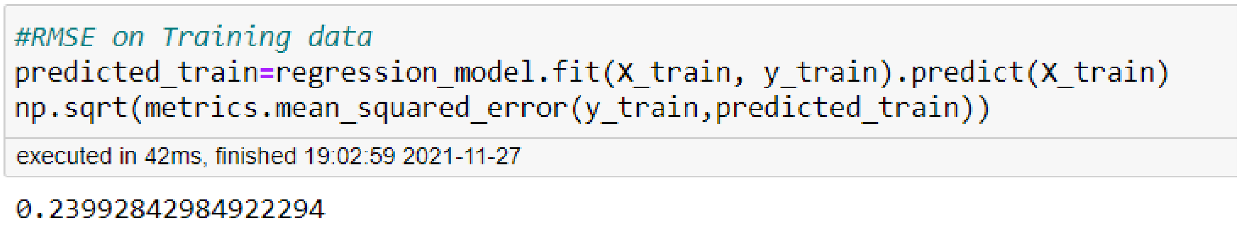
```
0.9419557931252712
```



```
# R square on testing data
regression_model.score(X_test, y_test)
```

executed in 40ms, finished 19:02:58 2021-11-27

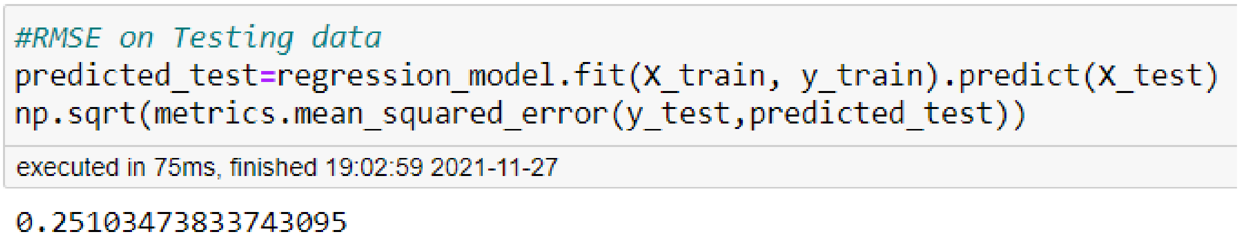
```
0.9381643998102491
```



```
#RMSE on Training data
predicted_train=regression_model.fit(X_train, y_train).predict(X_train)
np.sqrt(metrics.mean_squared_error(y_train,predicted_train))
```

executed in 42ms, finished 19:02:59 2021-11-27

```
0.23992842984922294
```



```
#RMSE on Testing data
predicted_test=regression_model.fit(X_train, y_train).predict(X_test)
np.sqrt(metrics.mean_squared_error(y_test,predicted_test))
```

executed in 75ms, finished 19:02:59 2021-11-27

```
0.25103473833743095
```

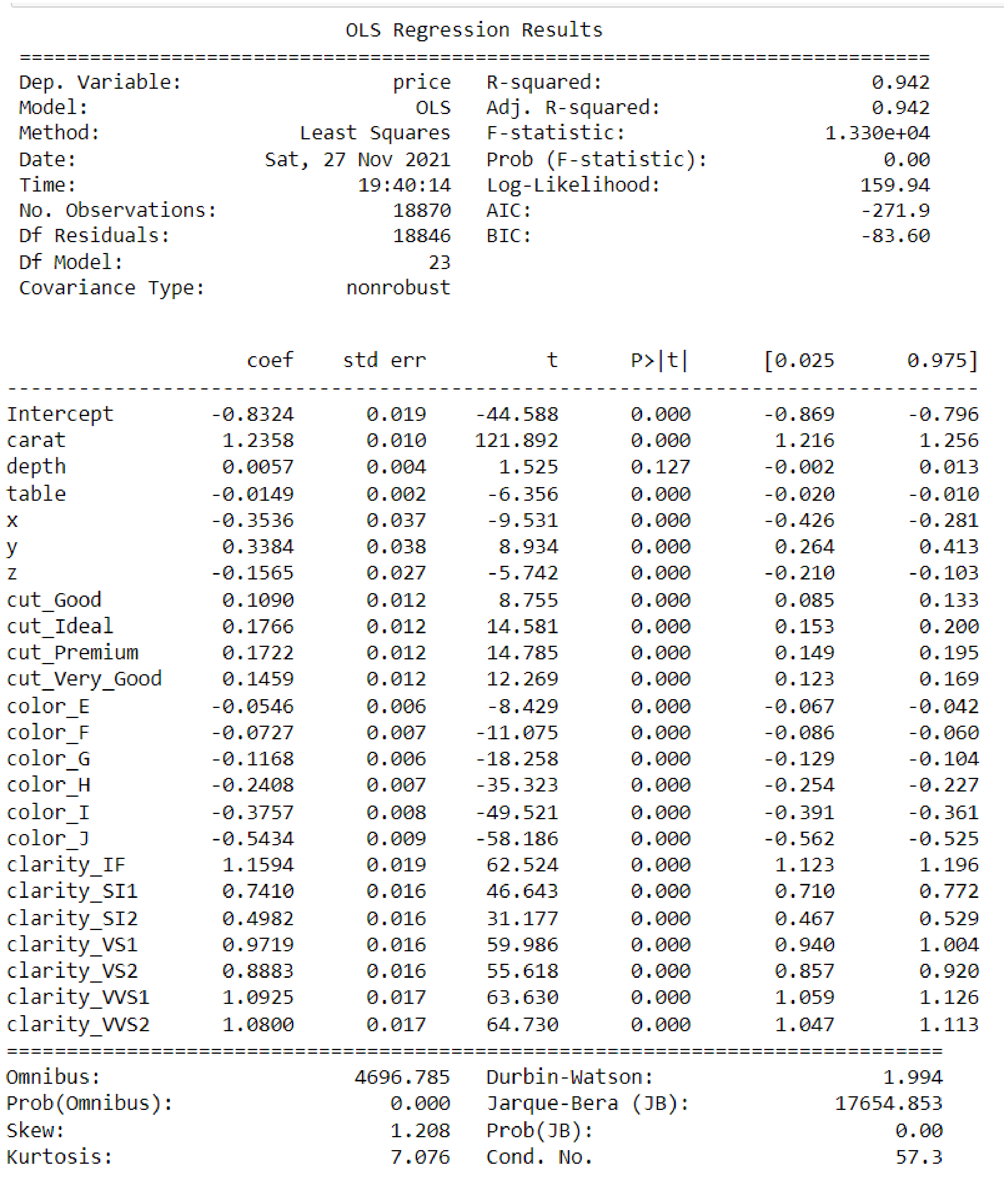
VIF (variance inflation factor) :

```
carat ---> 33.35287649550623
depth ---> 4.574003842337532
table ---> 1.772022611198982
x ---> 463.94494858728734
y ---> 463.08309600508517
z ---> 238.6002431605187
cut_Good ---> 3.6104961328079184
cut_Ideal ---> 14.347409690217939
cut_Premium ---> 8.623207030351887
cut_Very Good ---> 7.852218650260111
color_E ---> 2.3710537954581707
```

We can see in VIF there is multicollinearity in the datasets from the state models. We can understand which feature is not more important for the model. We can remove those variables . Ideal value of VIF is less than 10%.

Using Statsmodel library :

```
Intercept        -0.832352
carat             1.235809
depth             0.005682
table            -0.014930
x                -0.353556
y                 0.338429
z                -0.156488
cut_Good          0.109045
cut_Ideal         0.176624
cut_Premium       0.172237
cut_Very_Good     0.145927
color_E          -0.054566
color_F          -0.072691
color_G          -0.116800
color_H          -0.240824
color_I          -0.375667
color_J          -0.543391
clarity_IF        1.159363
clarity_SI1       0.740979
clarity_SI2       0.498173
clarity_VS1       0.971872
clarity_VS2       0.888305
clarity_VVS1      1.092461
clarity_VVS2      1.080042
dtype: float64
```



```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.942
Model:                            OLS   Adj. R-squared:                  0.942
Method:                 Least Squares   F-statistic:                 1.330e+04
Date:                Sat, 27 Nov 2021   Prob (F-statistic):               0.00
Time:                        19:40:14   Log-Likelihood:                 159.94
No. Observations:               18870   AIC:                            -271.9
Df Residuals:                   18846   BIC:                            -83.60
Df Model:                          23
Covariance Type:            nonrobust

                   coef    std err          t      P>|t|      [0.025      0.975]
---------------------------------------------------------------------------------
Intercept       -0.8324      0.019    -44.588      0.000      -0.869      -0.796
carat            1.2358      0.010    121.892      0.000       1.216       1.256
depth            0.0057      0.004      1.525      0.127      -0.002       0.013
table           -0.0149      0.002     -6.356      0.000      -0.020      -0.010
x               -0.3536      0.037     -9.531      0.000      -0.426      -0.281
y                0.3384      0.038      8.934      0.000       0.264       0.413
z               -0.1565      0.027     -5.742      0.000      -0.210      -0.103
cut_Good         0.1090      0.012      8.755      0.000       0.085       0.133
cut_Ideal        0.1766      0.012     14.581      0.000       0.153       0.200
cut_Premium      0.1722      0.012     14.785      0.000       0.149       0.195
cut_Very_Good    0.1459      0.012     12.269      0.000       0.123       0.169
color_E         -0.0546      0.006     -8.429      0.000      -0.067      -0.042
color_F         -0.0727      0.007    -11.075      0.000      -0.086      -0.060
color_G         -0.1168      0.006    -18.258      0.000      -0.129      -0.104
color_H         -0.2408      0.007    -35.323      0.000      -0.254      -0.227
color_I         -0.3757      0.008    -49.521      0.000      -0.391      -0.361
color_J         -0.5434      0.009    -58.186      0.000      -0.562      -0.525
clarity_IF       1.1594      0.019     62.524      0.000       1.123       1.196
clarity_SI1      0.7410      0.016     46.643      0.000       0.710       0.772
clarity_SI2      0.4982      0.016     31.177      0.000       0.467       0.529
clarity_VS1      0.9719      0.016     59.986      0.000       0.940       1.004
clarity_VS2      0.8883      0.016     55.618      0.000       0.857       0.920
clarity_VVS1     1.0925      0.017     63.630      0.000       1.059       1.126
clarity_VVS2     1.0800      0.017     64.730      0.000       1.047       1.113
==============================================================================
Omnibus:                     4696.785   Durbin-Watson:                   1.994
Prob(Omnibus):                  0.000   Jarque-Bera (JB):            17654.853
Skew:                           1.208   Prob(JB):                         0.00
Kurtosis:                       7.076   Cond. No.                         57.3
```

In this model we found that depth p>|t| is greater than 0.5 so we should remove and again build the model.

```
Intercept        -0.832269
carat             1.236946
table            -0.015612
x                -0.365810
y                 0.315607
z                -0.122330
cut_Good          0.110307
cut_Ideal         0.175305
cut_Premium       0.170878
cut_Very_Good     0.145550
color_E          -0.054635
color_F          -0.072740
color_G          -0.116725
color_H          -0.240702
color_I          -0.375359
color_J          -0.543203
clarity_IF        1.159684
clarity_SI1       0.741936
clarity_SI2       0.498868
clarity_VS1       0.972497
clarity_VS2       0.889069
clarity_VVS1      1.092825
clarity_VVS2      1.080657
dtype: float64
```

```
print(lm2.summary())
```

executed in 41ms, finished 19:56:43 2021-11-27

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.942
Model:                            OLS   Adj. R-squared:                  0.942
Method:                 Least Squares   F-statistic:                 1.390e+04
Date:                Sat, 27 Nov 2021   Prob (F-statistic):               0.00
Time:                        19:56:43   Log-Likelihood:                 158.78
No. Observations:               18870   AIC:                            -271.6
Df Residuals:                   18847   BIC:                            -91.12
Df Model:                          22
Covariance Type:            nonrobust
```

```
--------------------------------------------------------------------------------
                   coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------
Intercept       -0.8323      0.019    -44.583      0.000      -0.869      -0.796
carat            1.2369      0.010    122.331      0.000       1.217       1.257
table           -0.0156      0.002     -6.770      0.000      -0.020      -0.011
x               -0.3658      0.036    -10.101      0.000      -0.437      -0.295
y                0.3156      0.035      9.069      0.000       0.247       0.384
z               -0.1223      0.016     -7.883      0.000      -0.153      -0.092
cut_Good         0.1103      0.012      8.876      0.000       0.086       0.135
cut_Ideal        0.1753      0.012     14.508      0.000       0.152       0.199
cut_Premium      0.1709      0.012     14.711      0.000       0.148       0.194
cut_Very_Good    0.1455      0.012     12.239      0.000       0.122       0.169
color_E         -0.0546      0.006     -8.439      0.000      -0.067      -0.042
color_F         -0.0727      0.007    -11.082      0.000      -0.086      -0.060
color_G         -0.1167      0.006    -18.246      0.000      -0.129      -0.104
color_H         -0.2407      0.007    -35.306      0.000      -0.254      -0.227
color_I         -0.3754      0.008    -49.497      0.000      -0.390      -0.360
color_J         -0.5432      0.009    -58.169      0.000      -0.562      -0.525
clarity_IF       1.1597      0.019     62.544      0.000       1.123       1.196
clarity_SI1      0.7419      0.016     46.738      0.000       0.711       0.773
clarity_SI2      0.4989      0.016     31.232      0.000       0.468       0.530
clarity_VS1      0.9725      0.016     60.042      0.000       0.941       1.004
clarity_VS2      0.8891      0.016     55.691      0.000       0.858       0.920
clarity_VVS1     1.0928      0.017     63.655      0.000       1.059       1.126
clarity_VVS2     1.0807      0.017     64.784      0.000       1.048       1.113
==============================================================================
Omnibus:                     4699.504   Durbin-Watson:                   1.994
Prob(Omnibus):                  0.000   Jarque-Bera (JB):            17704.272
Skew:                           1.208   Prob(JB):                         0.00
Kurtosis:                       7.084   Cond. No.                         56.9
```

Conclusion :

The final Linear Regression equation is :

Log_price = (-0.83) * Intercept + (1.24) * carat + (-0.02) * table + (-0.37) * x + (0.32) * y + (-0.12) * z + (0.11) * cut_Good + (0.18) * cut_Ideal + (0.17) * cut_Premium + (0.15) * cut_Very_Good + (-0.05) * color_E + (-0.07) * color_F + (-0.12) * color_G + (-0.24) * color_H + (-0.38) * color_I + (-0.54) * color_J + (1.16) * clarity_IF + (0.74) * clarity_SI1 + (0.5) * clarity_SI2 + (0.97) * clarity_VS1 + (0.89) * clarity_VS2 + (1.09) * clarity_VVS1 + (1.08) * clarity_VVS2

Remarks : Dropping variables would bring down the multicollinearity level down.

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

Please explain and summarise the various steps performed in this project. There should be proper business interpretation and actionable insights present.

- From the EDA analysis we understand that ideal cuts have more profit to the company.
- Color H, I, J have more profit for the company.
- need to run promotional marketing campaigns or evaluate if we need to tie up with an alternate agency. It will increase sales.
- Using sats model, if we could run the model again we can have P values and coefficients which will give us better understanding of the relationship, so that values more 0.05 we can drop those variables and run again the model for better result.
- The ideal, premium, very good, cut types are those which are bringing profit so that we could use marketing for these to bring us more profits.

Problem 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.

Sample of the dataset:

	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-05 Sample datasets

```
<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
```

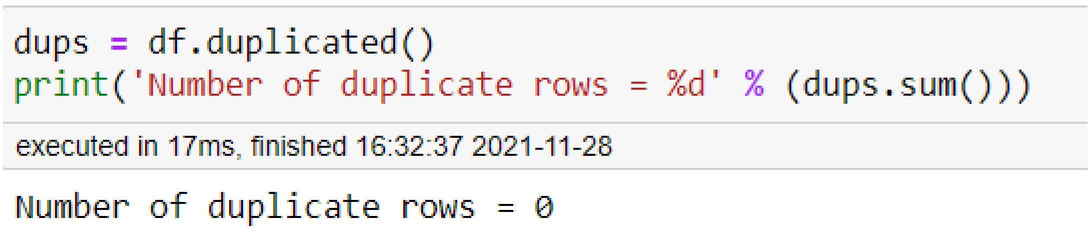
- From the above results we can see that there is no missing value present in the dataset.
- There are a total of 872 rows and 7 columns in the dataset.
- 2 variables out of 7 are objects and 5 variables are int64 data types.

Missing values :

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

From the above result we can see that there is no missing value in the datasets.

Duplicate values :



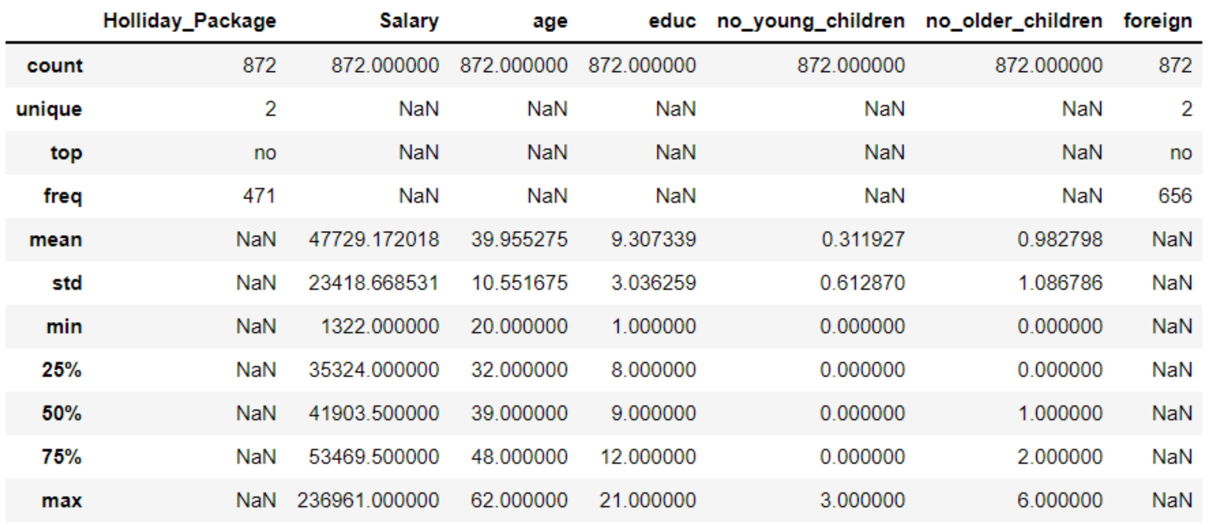
```
dups = df.duplicated()
print('Number of duplicate rows = %d' % (dups.sum()))
```

executed in 17ms, finished 16:32:37 2021-11-28

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

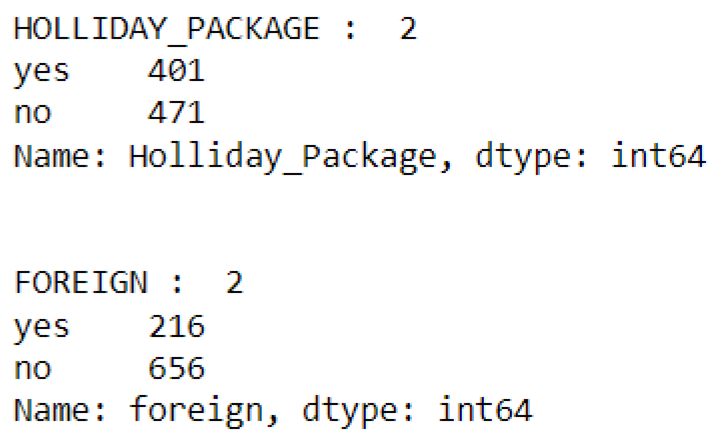
There is no duplicate value in the datasets.

Describe datasets :



	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

Table-06 Describe data



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


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

- We have 2 object variables and the remaining are numerical variables.
- The Holloday_package variable is target variable.
- In the FOREIGH variable most of the data is non-foreigner.

Univariate / Bivariate analysis :

Distplot :

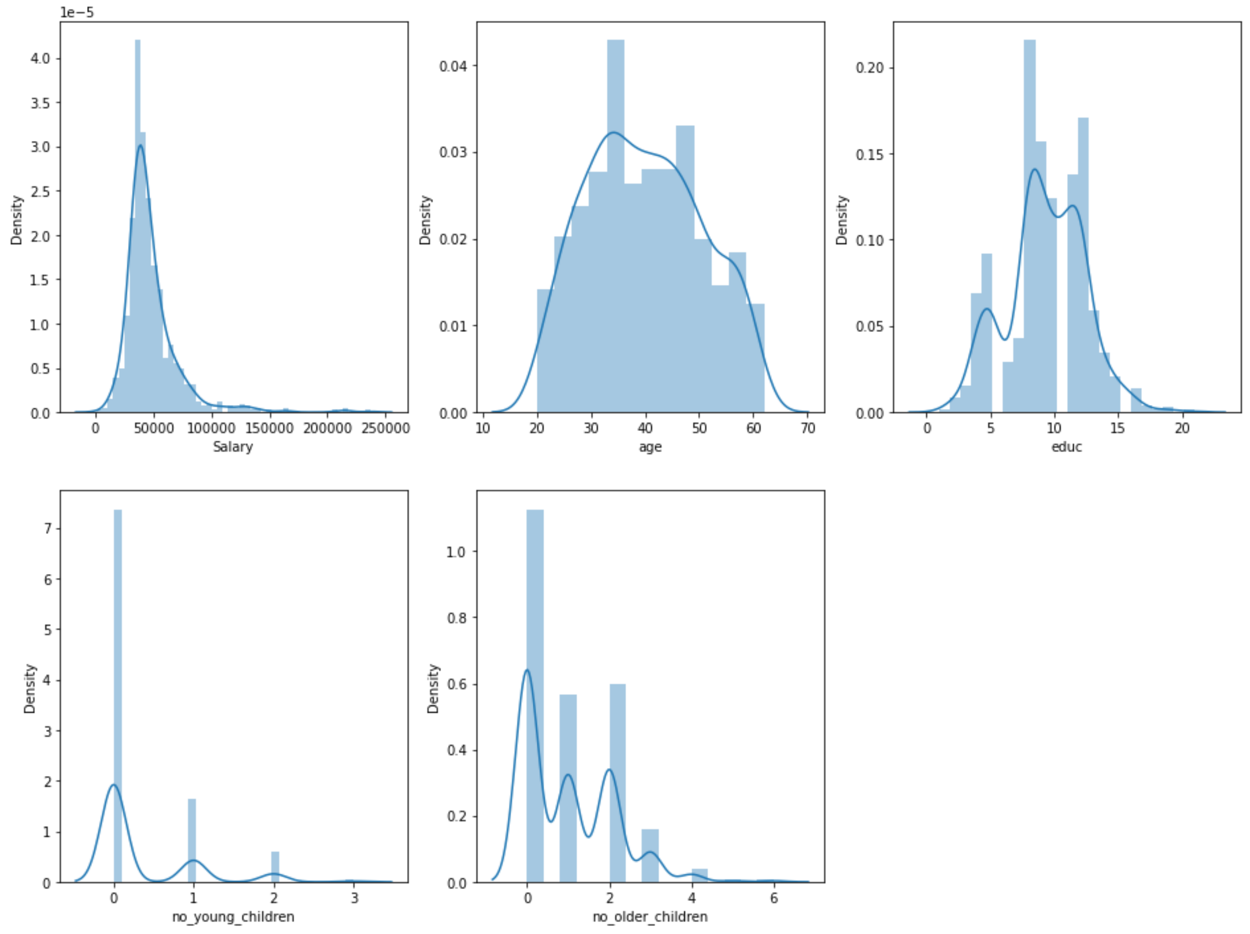


Figure-07

Boxplot :

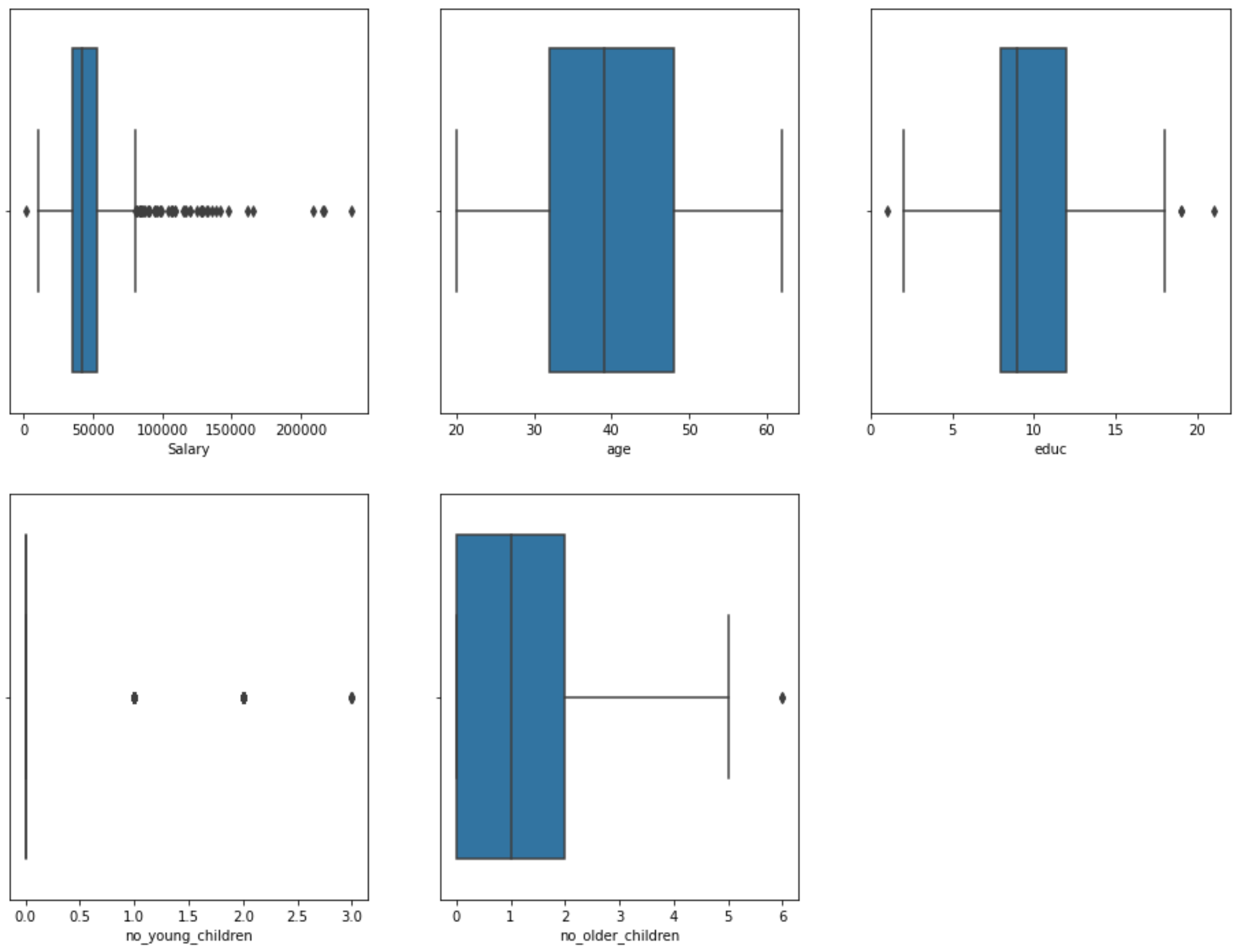


Figure-08

Salary Feature :

- Distribution of salary feature seems right skewed.
- In the distplot there seem to be outliers.

Age feature :

- Distribution of age feature seems normally distributed.
- In the distplot there seems to be no outliers.

Educ feature :

- Distribution of educ features there are multiple peak points.
- There are some outliers in the datasets.

Countplot :

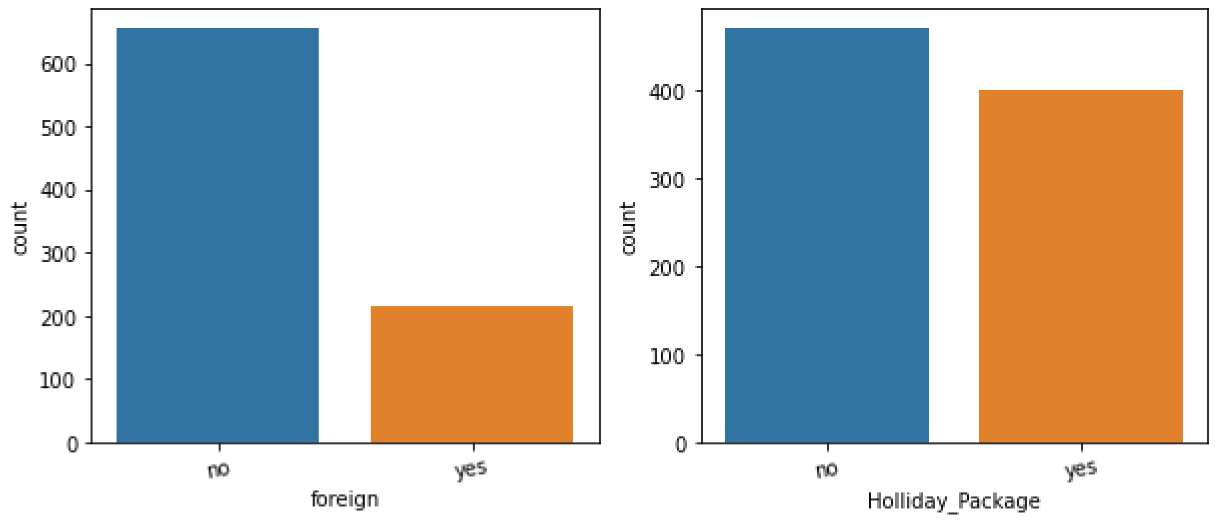


Figure-09

Bivariant :

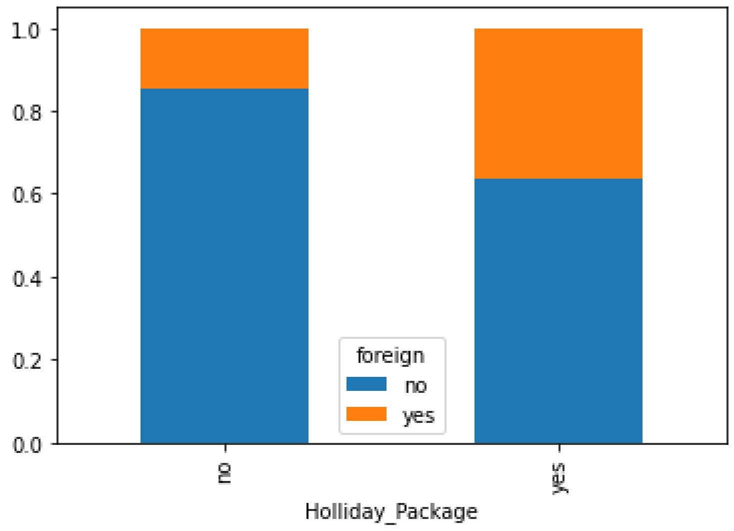


Figure-10

In the above figure we can say that foreigners are more interested in holiday packages.

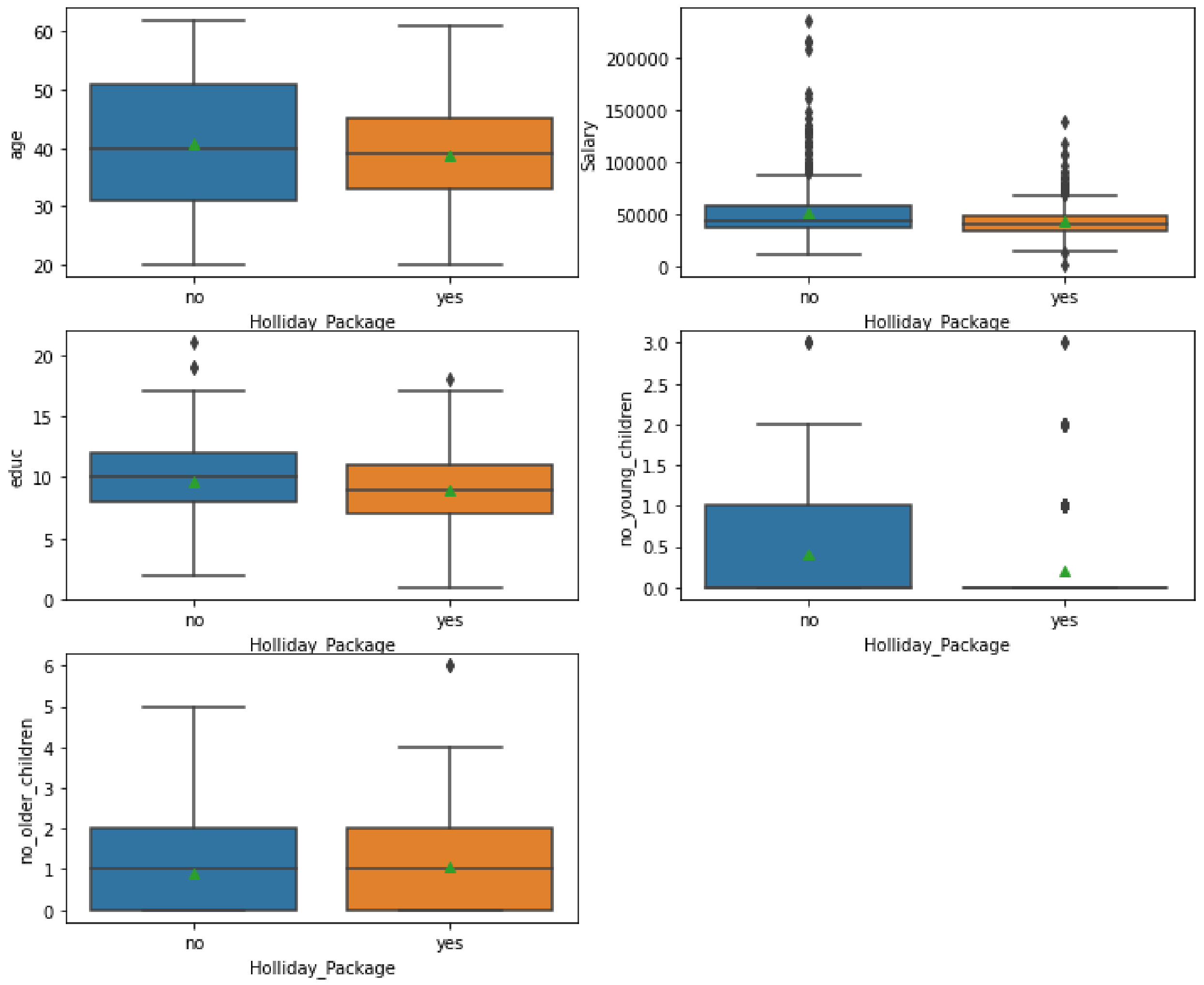

age
Salary
educ
no_young_children
no_older_children
Holliday_Package
no
yes

Figure -11

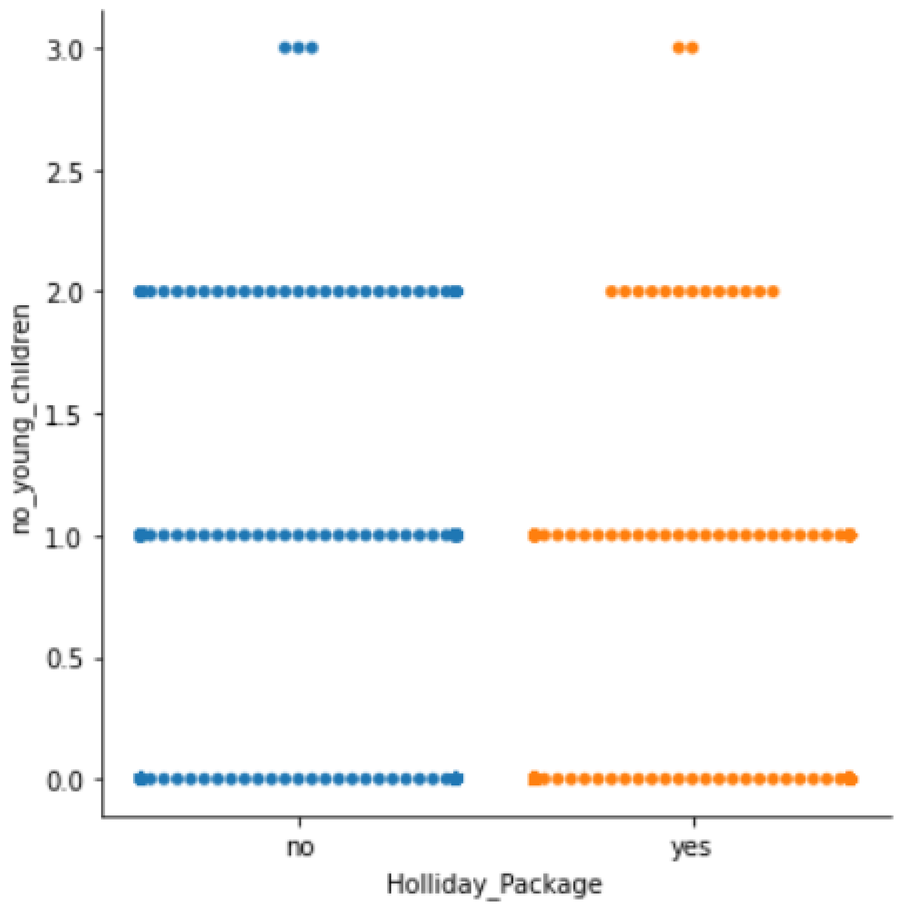


Figure -12

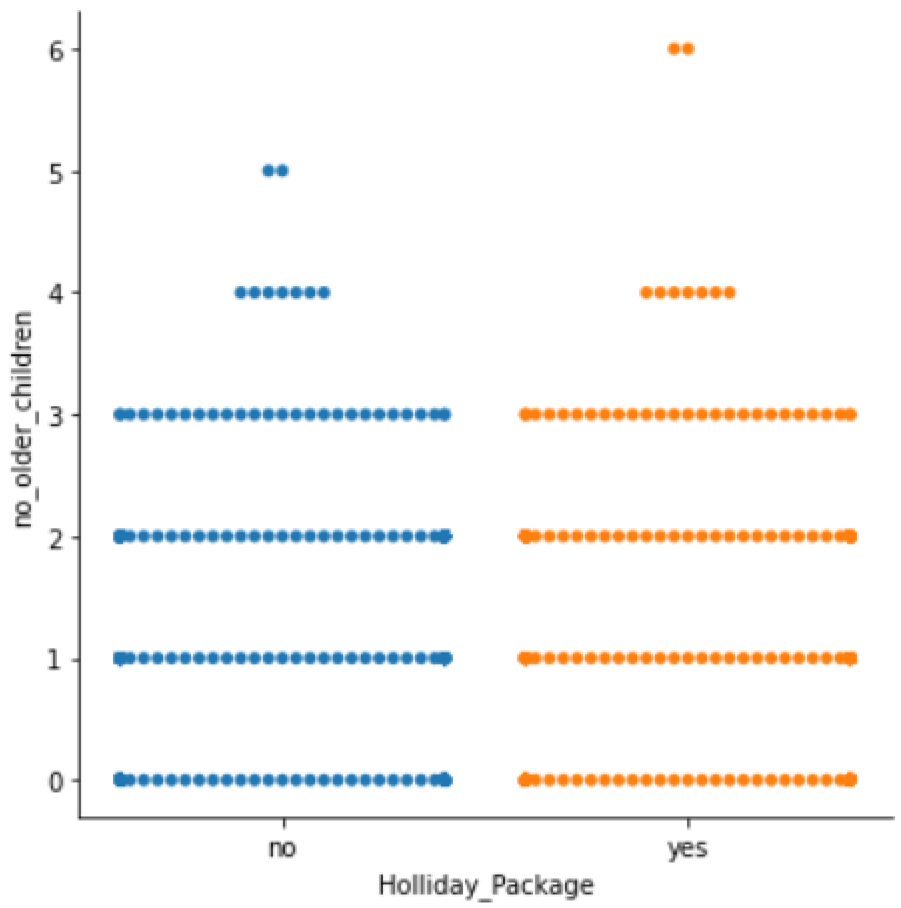


Figure-13

Pairplot :

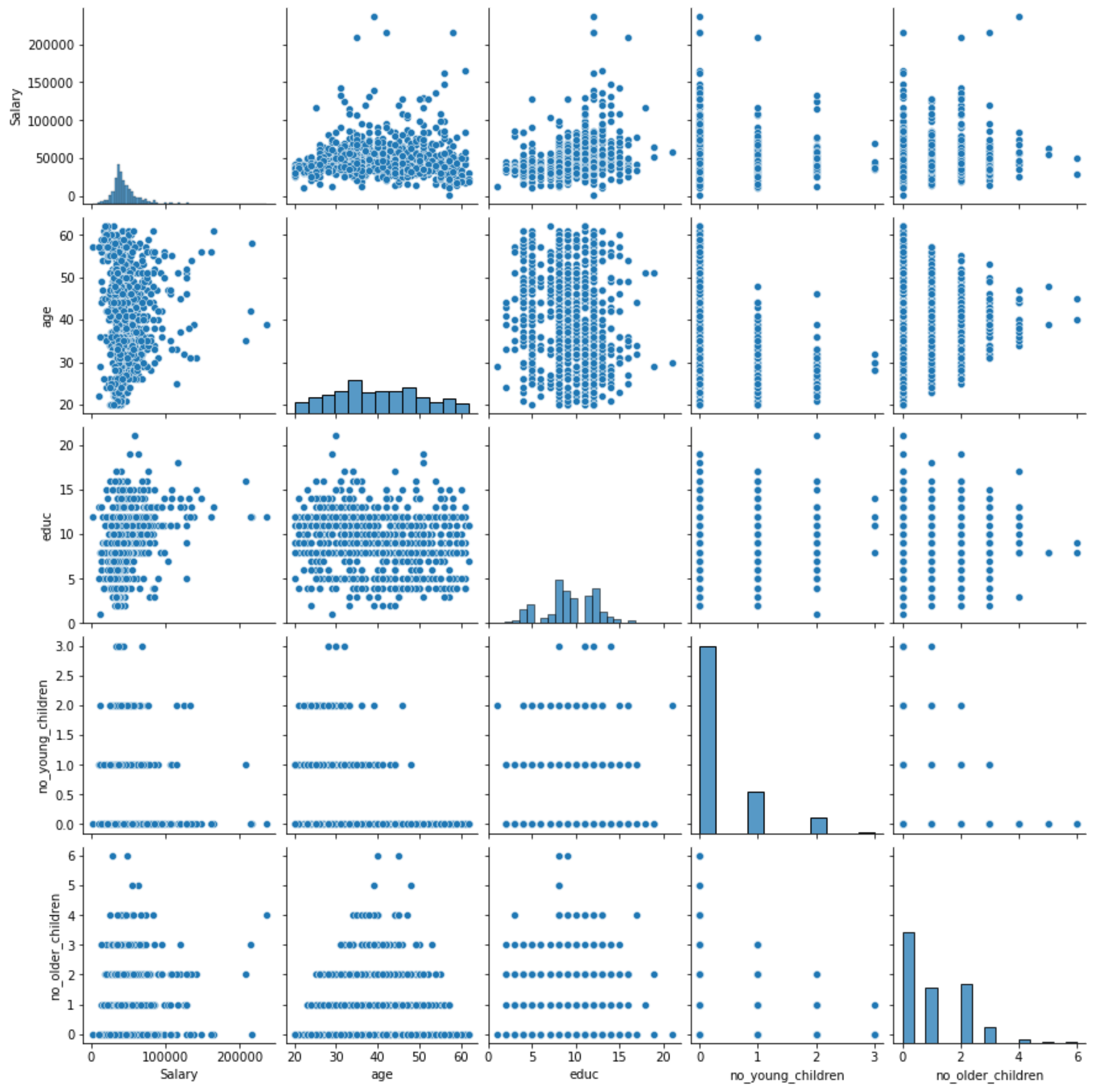


Figure - 14

Heatmap :

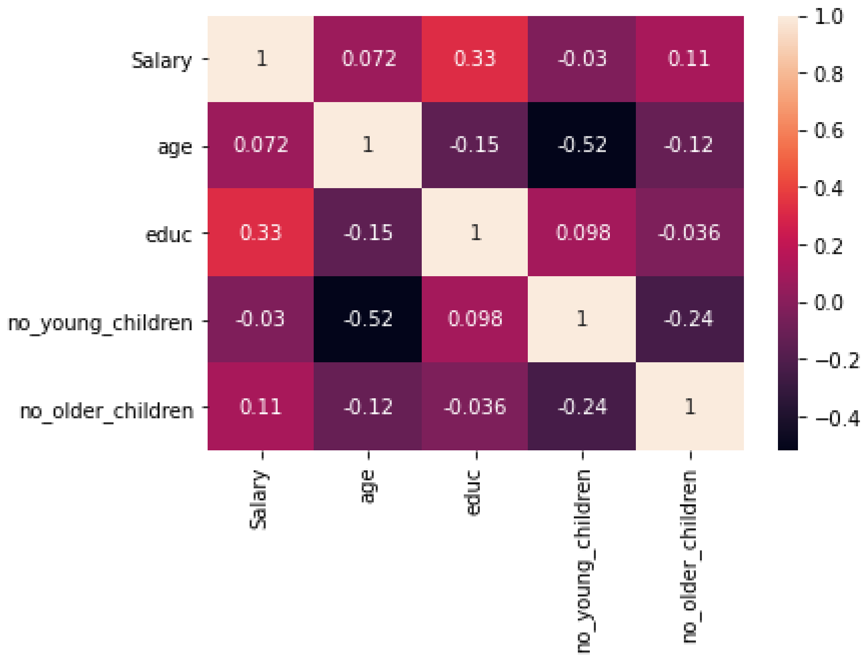
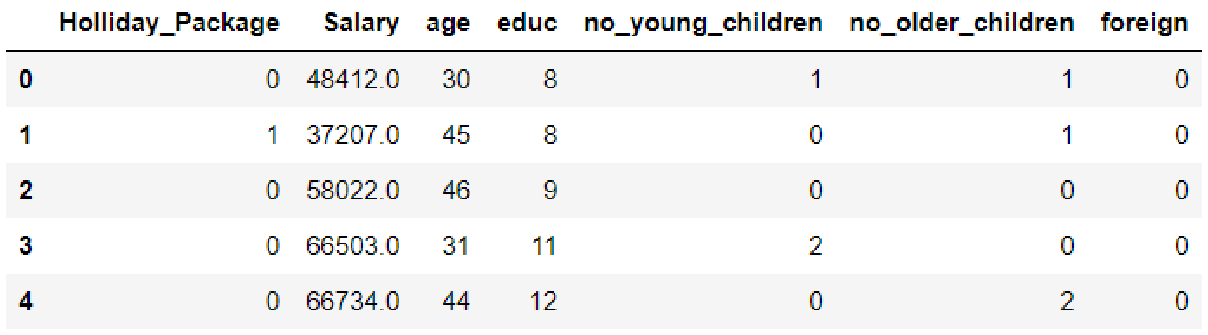


Figure-15

After seeing the pairplot and heatmap we can say that there is presence of multicollinearity in the datasets.

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).

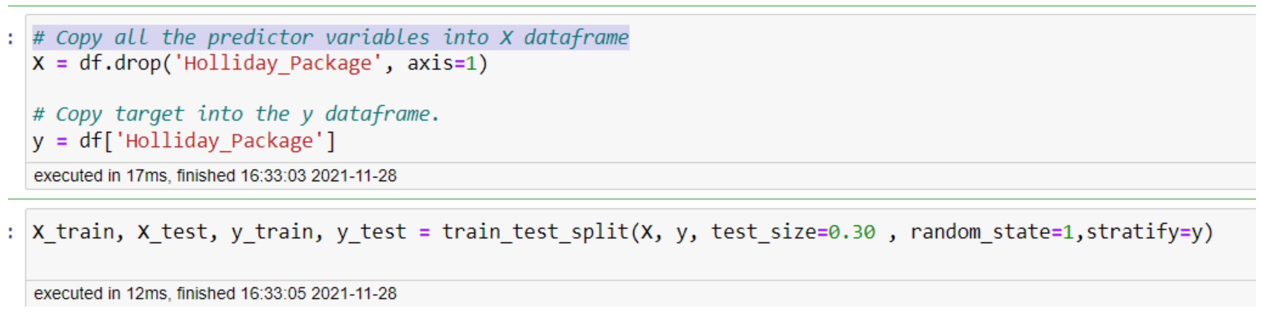
Encode the string variables :



	Holliday_Package	Salary	age	educ	no_young_children	no_older_children	foreign
0	0	48412.0	30	8	1	1	0
1	1	37207.0	45	8	0	1	0
2	0	58022.0	46	9	0	0	0
3	0	66503.0	31	11	2	0	0
4	0	66734.0	44	12	0	2	0

. Table-07

Dropping target variables for the model and split the data 70:30 ratio for the train and test respectively.



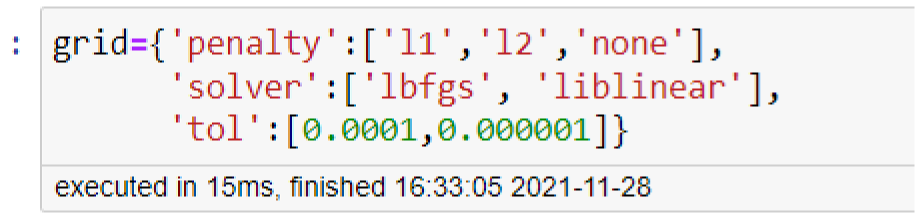
```
: # Copy all the predictor variables into X dataframe
  X = df.drop('Holliday_Package', axis=1)

  # Copy target into the y dataframe.
  y = df['Holliday_Package']
  executed in 17ms, finished 16:33:03 2021-11-28
```

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

  executed in 12ms, finished 16:33:05 2021-11-28
```

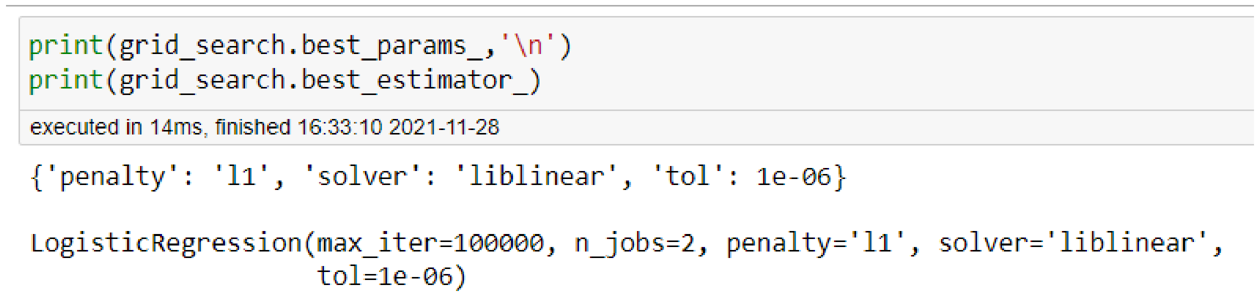
Logistic Regression :

Grid Search Method :

Grid search method is used for logistic regression to find the optimal solving and the parameters.



```
: grid={'penalty':['l1','l2','none'],
        'solver':['lbfgs', 'liblinear'],
        'tol':[0.0001,0.000001]}
  executed in 15ms, finished 16:33:05 2021-11-28
```

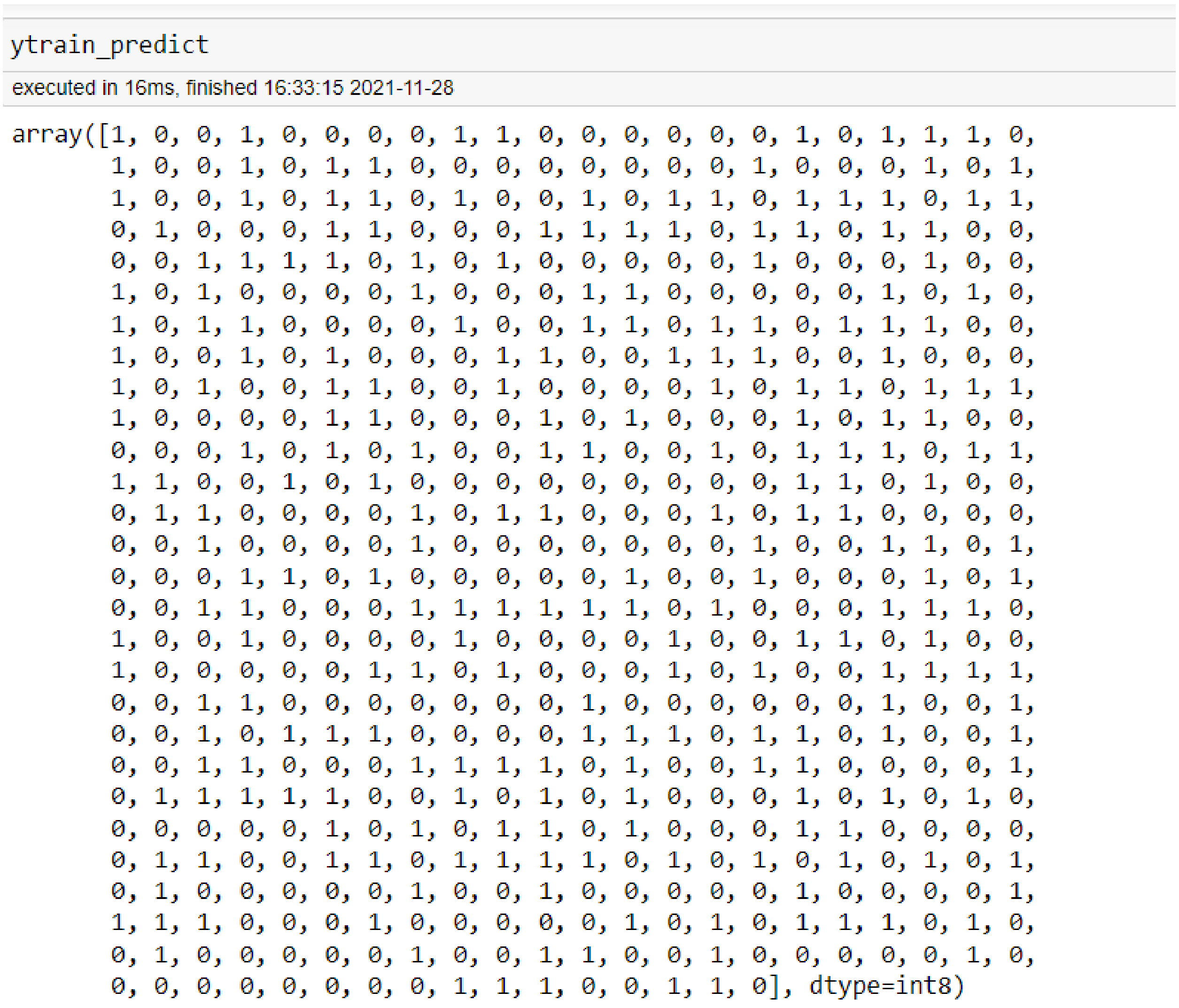


```
print(grid_search.best_params_,'\n')
print(grid_search.best_estimator_)
executed in 14ms, finished 16:33:10 2021-11-28
```

```
{'penalty': 'l1', 'solver': 'liblinear', 'tol': 1e-06}

LogisticRegression(max_iter=100000, n_jobs=2, penalty='l1', solver='liblinear',
                   tol=1e-06)
```

The Grid search method give, solver= liblinear, penalty= l1

Predicting the training data :

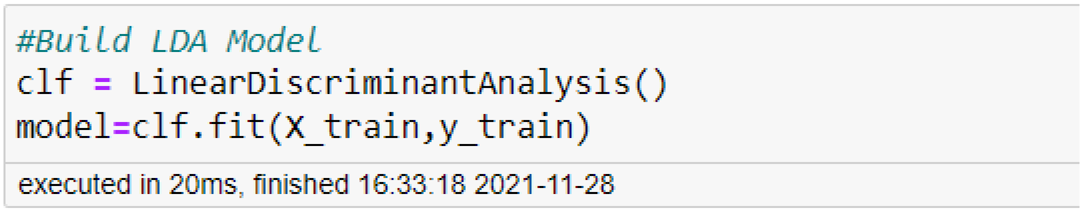


```
ytrain_predict
```

executed in 16ms, finished 16:33:15 2021-11-28

```
array([1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0,
       1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1,
       1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1,
       0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0,
       0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0,
       1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0,
       1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0,
       1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0,
       1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1,
       1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0,
       0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1,
       1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0,
       0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0,
       0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1,
       0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1,
       0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0,
       1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0,
       1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 1,
       0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1,
       0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1,
       0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1,
       0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0,
       0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0,
       0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1,
       0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1,
       1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0,
       0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0], dtype=int8)
```

LDA MODEL :



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

```
executed in 20ms, finished 16:33:18 2021-11-28
```

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.

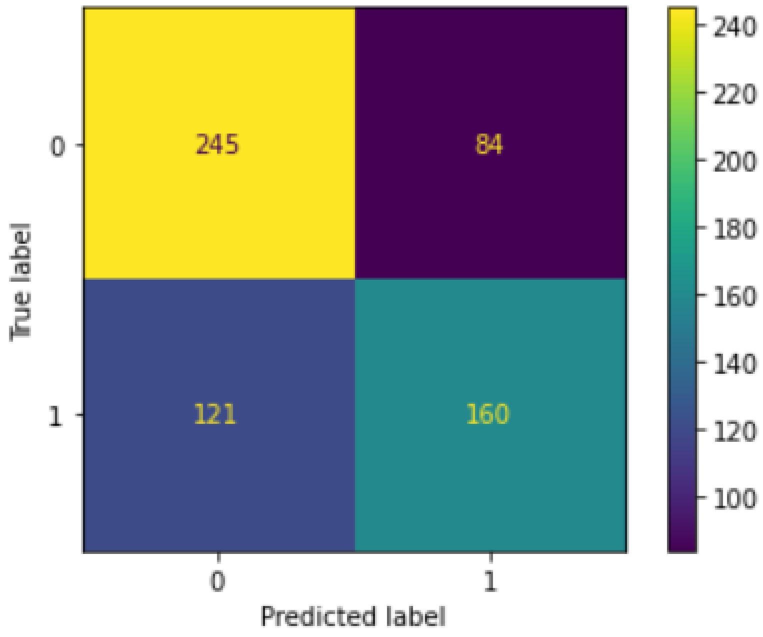
Logistic Regression Performance :

Confusion matrix on training data :

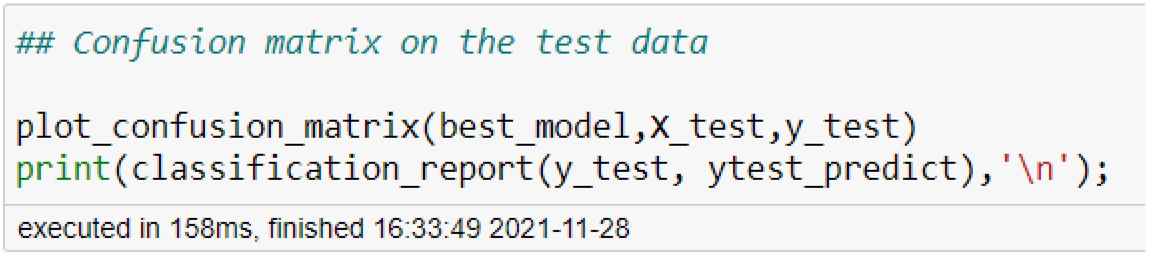
```
## Confusion matrix on the training data

plot_confusion_matrix(best_model,X_train,y_train)
print(classification_report(y_train, ytrain_predict),'\n');
```

executed in 163ms, finished 16:33:49 2021-11-28

```
              precision    recall  f1-score   support

           0       0.67      0.74      0.71       329
           1       0.66      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
```



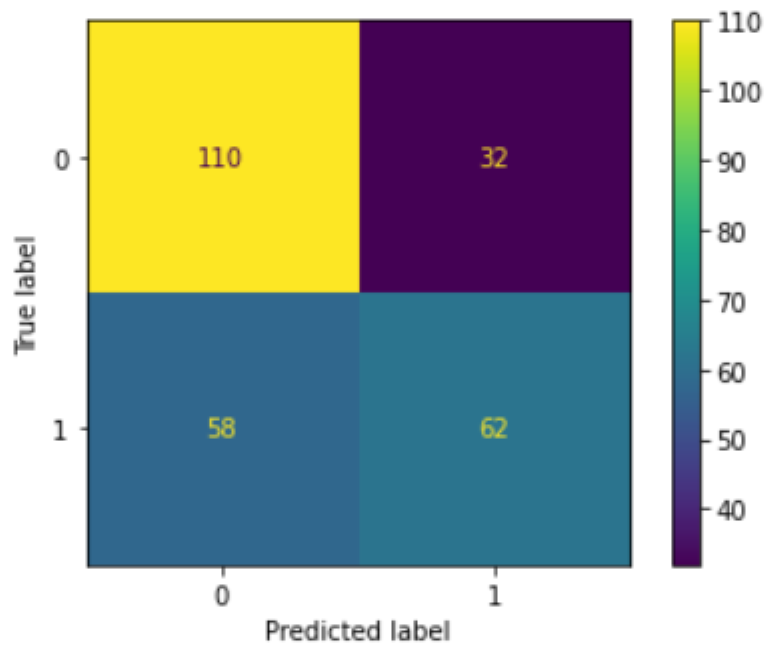
Confusion matrix on testing data :



```
## Confusion matrix on the test data

plot_confusion_matrix(best_model,X_test,y_test)
print(classification_report(y_test, ytest_predict),'\n');
```

executed in 158ms, finished 16:33:49 2021-11-28

```
              precision    recall  f1-score   support

           0       0.65      0.77      0.71       142
           1       0.66      0.52      0.58       120

    accuracy                           0.66       262
   macro avg       0.66      0.65      0.64       262
weighted avg       0.66      0.66      0.65       262
```



Accuracy :

```
# Accuracy - Training Data

lr_train_acc = best_model.score(X_train, y_train)
lr_train_acc
```

executed in 10ms, finished 16:33:50 2021-11-28

```
0.6639344262295082
```

AUC , ROC curve for train data :

```
AUC: 0.733
```

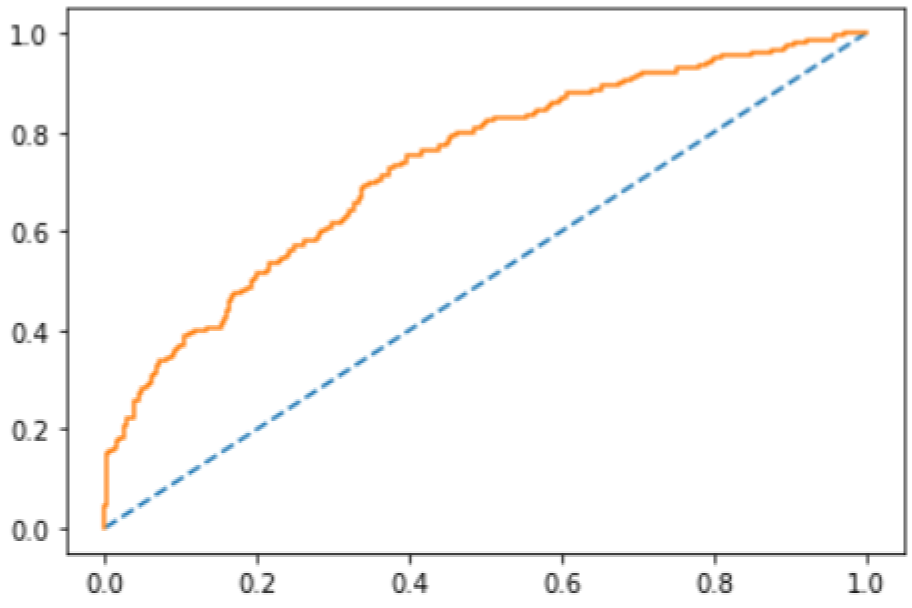


Figure-16

AUC, ROC cure for test data :

```
AUC: 0.716
```

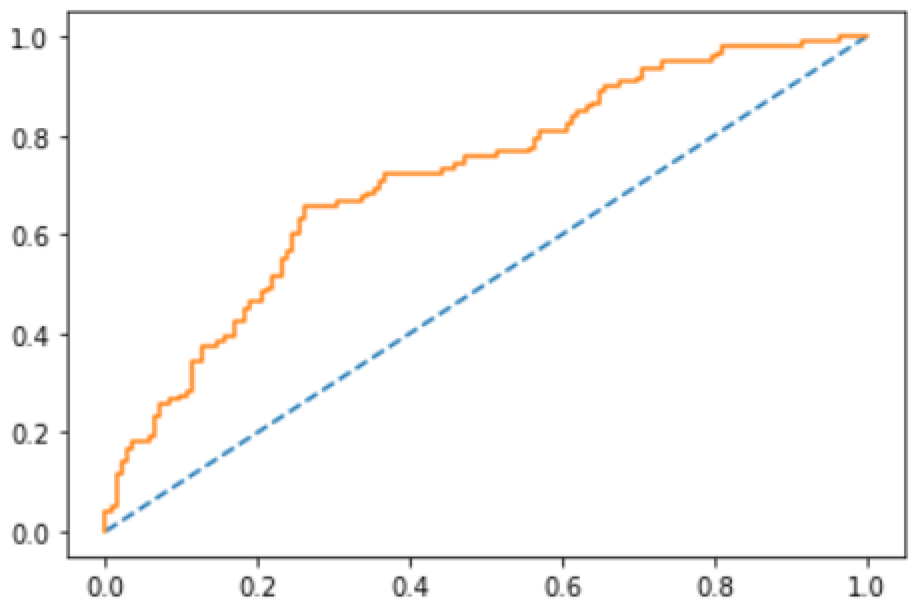


Figure- 17

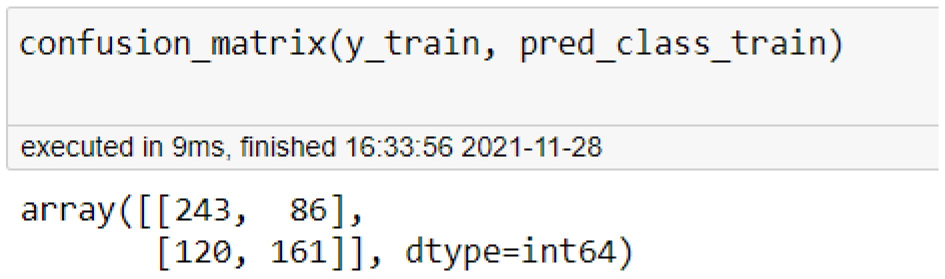
LDA MODEL Performance :

Classification report on training data :

```
print(classification_report(y_train, pred_class_train))
```

executed in 16ms, finished 16:33:55 2021-11-28

```
              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
```

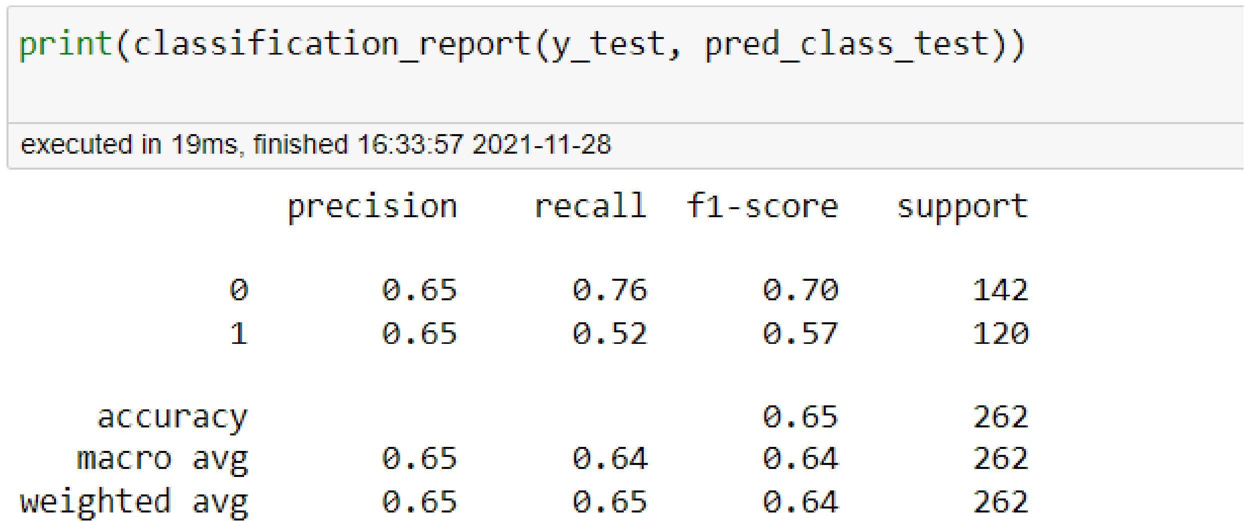


```
confusion_matrix(y_train, pred_class_train)
```

executed in 9ms, finished 16:33:56 2021-11-28

```
array([[243,  86],
       [120, 161]], dtype=int64)
```

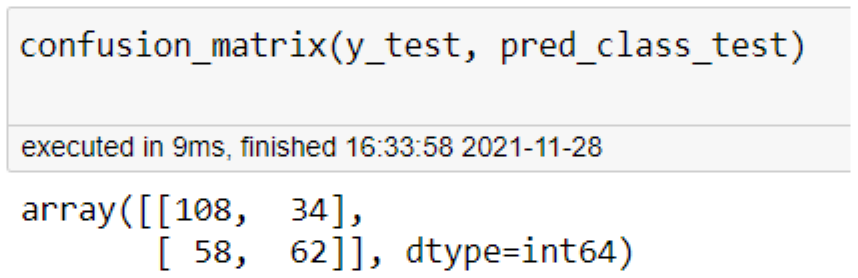
Classification report on testing data :



```
print(classification_report(y_test, pred_class_test))
```

executed in 19ms, finished 16:33:57 2021-11-28

```
              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
```



```
confusion_matrix(y_test, pred_class_test)
```

executed in 9ms, finished 16:33:58 2021-11-28

```
array([[108,  34],
       [ 58,  62]], dtype=int64)
```

AUC and ROC for the training and testing data :

```
AUC for the Training Data: 0.731
AUC for the Test Data: 0.714
```

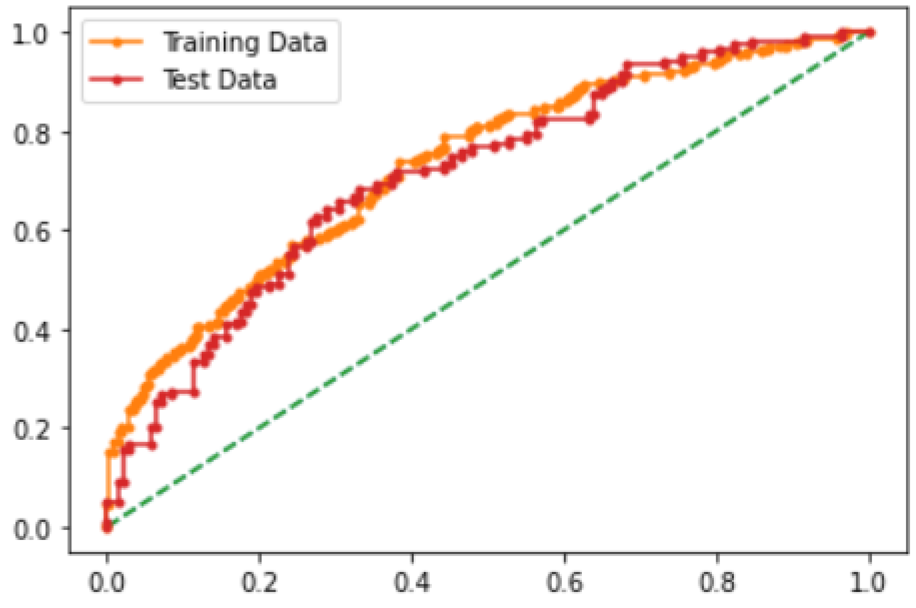


Figure-18

Comparison :

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

Table-08

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

Please explain and summarise the various steps performed in this project. There should be proper business interpretation and actionable insights present.

- We have done predictions for both logistic regression and LDA. Since both results are the same.
- EDA analysis clearly indicates certain people aged above 50 are not interested much for holiday.
- People ranging from 35 to 45 generally opted for holiday.
- The important factor deciding the predictions are salary, age and educ.
- To improve holiday packages above 50 we can provide them religious destination places.
- For people earning more than 150000 we can provide them vacation holiday packages.
- Who have more than a number of older children we can provide packages in holiday vacation places.
- need to run promotional marketing campaigns or evaluate if we need to tie up with an alternate agency. It will increase sales.