Project - Predictive Modelling,

Name: Ajay Patwari

Problem 1: Linear Regression

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.

Below I've used the head function to show the top 5 rows and 10 columns.

In [5]:
```
df.head()
```

Out[5]:

	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

In [233]:
```
df.info()
```

```
<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
```

Below I've used the describe function which shows the mean, standard deviation, median, minimum value and the maximum value.

```
In [234]: df.describe(include='all').T
```

Out[234]:

	count	unique	top	freq	mean	std	min	25%	50%	75%	max
carat	26967	NaN	NaN	NaN	0.798375	0.477745	0.2	0.4	0.7	1.05	4.5
cut	26967	5	Ideal	10816	NaN	NaN	NaN	NaN	NaN	NaN	NaN
color	26967	7	G	5661	NaN	NaN	NaN	NaN	NaN	NaN	NaN
clarity	26967	8	SI1	6571	NaN	NaN	NaN	NaN	NaN	NaN	NaN
depth	26270	NaN	NaN	NaN	61.7451	1.41286	50.8	61	61.8	62.5	73.6
table	26967	NaN	NaN	NaN	57.4561	2.23207	49	56	57	59	79
x	26967	NaN	NaN	NaN	5.72985	1.12852	0	4.71	5.69	6.55	10.23
y	26967	NaN	NaN	NaN	5.73357	1.16606	0	4.71	5.71	6.54	58.9
z	26967	NaN	NaN	NaN	3.53806	0.720624	0	2.9	3.52	4.04	31.8
price	26967	NaN	NaN	NaN	3939.52	4024.86	326	945	2375	5360	18818

Check for Null Values

```
In [235]: df.isnull().sum()
```

Out[235]:

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

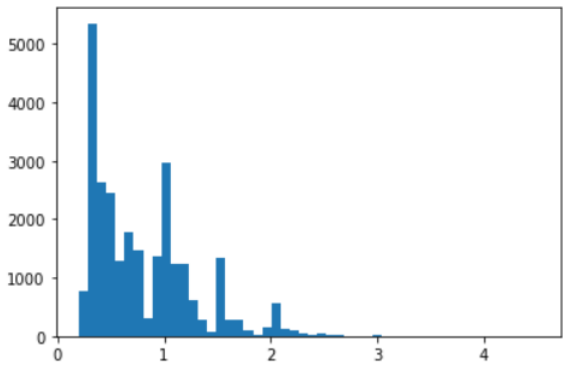
I have checked the null values and we can see that the column depth has 697 null values.

% Null values

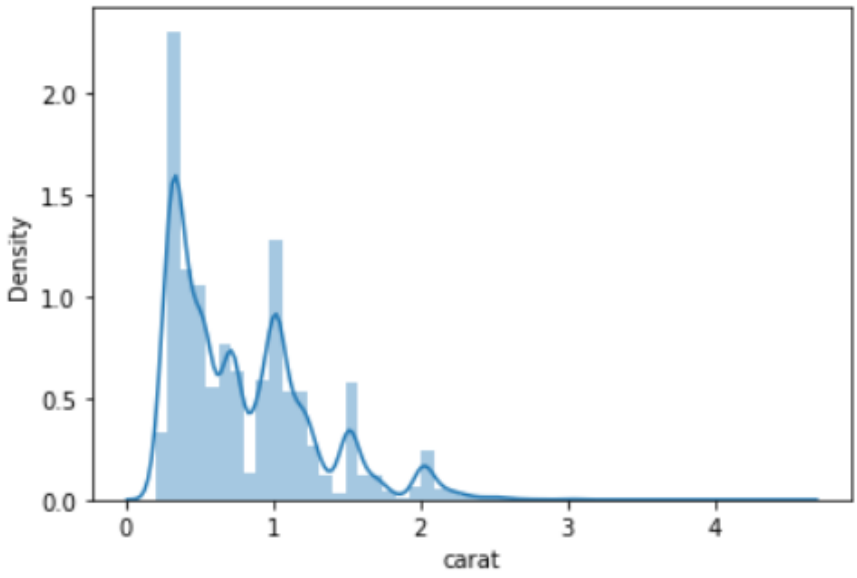
```
In [132]: df.isnull().sum()/df.isnull().sum().sum()*100

Out[132]: Unnamed: 0      0.0
          carat           0.0
          cut             0.0
          color           0.0
          clarity         0.0
          depth         100.0
          table           0.0
          x               0.0
          y               0.0
          z               0.0
          price           0.0
          dtype: float64
```

I've used the shape function and the output says we have 26967 rows and 10 columns.

I have done the univariate analysis on two of the columns. I have done it on Carat and Price. There are no missing values on Carat and Price

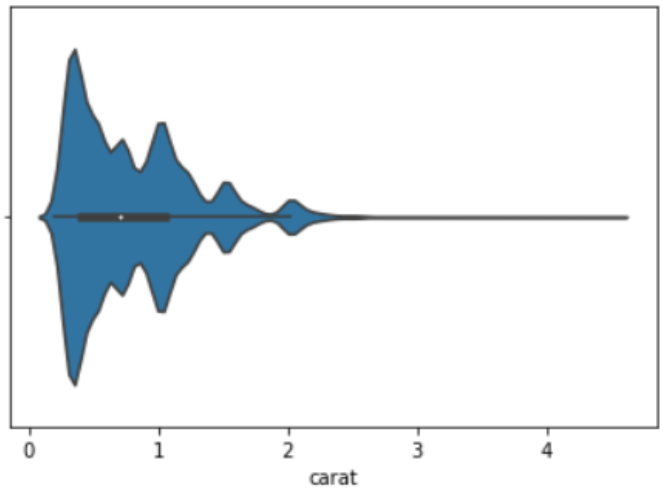


Above is the Histogram Plot for Carat column.

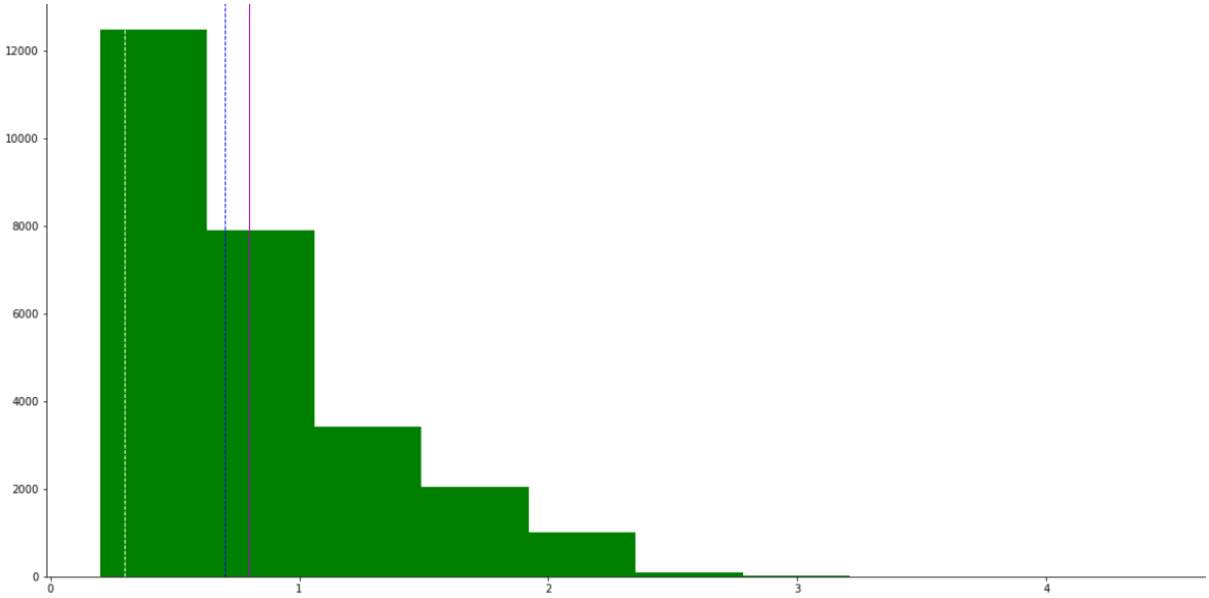


Above is the distribution plot for the carat column.

```
Out[141]: <AxesSubplot:xlabel='carat'>
```



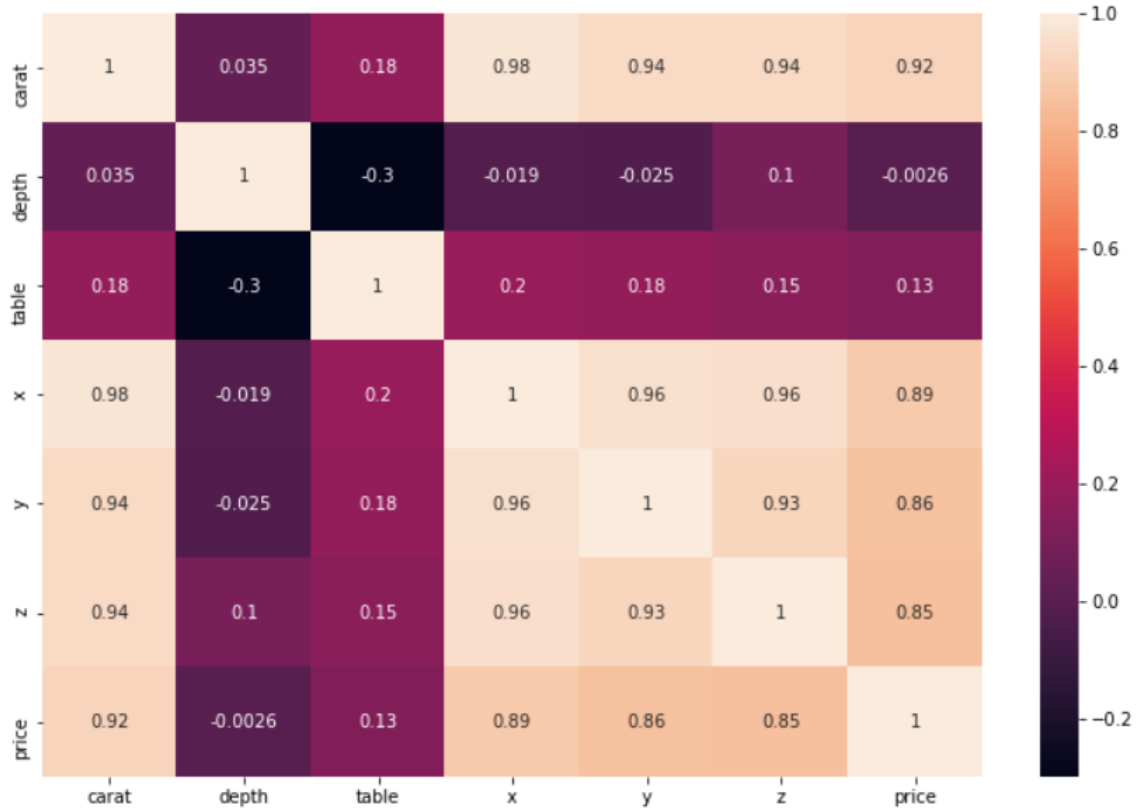
Violin Plot for carat column is shown in the above figure.



The above is the Histogram Plot for the Carat column.

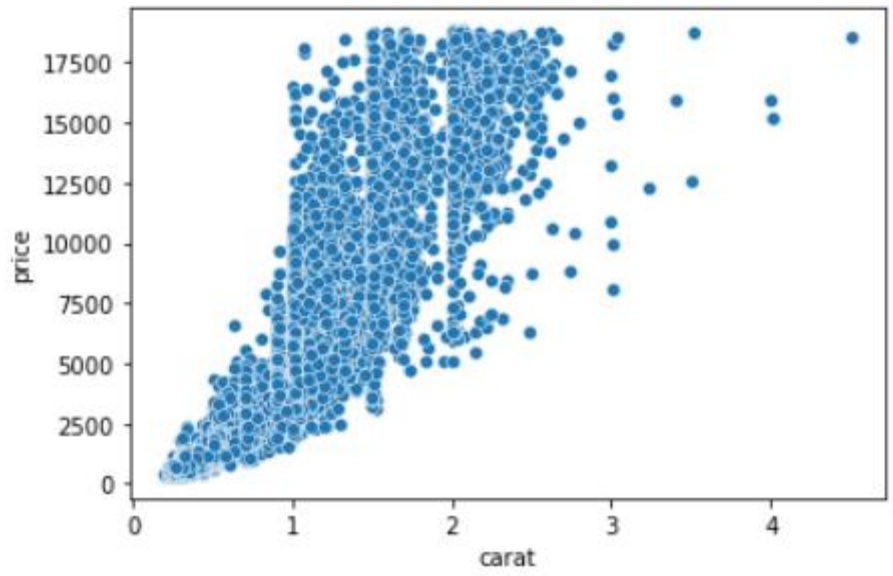
And the same univariate analysis is done on other column Price as well and it can be found in the coding part.

Then coming to Bivariate analysis, below I will show the heatmap which will corelation between all the columns.



If you see columns price and carat they have corelation of about 0.92 which is highly corelated. Any values which are close to 1 are highly related to each other. If you look at price and depth they are least related because we can see that the value is -0.0026

I've drawn the pair plot which is similar to the heat map but the presentation of the relation is different. You can see that the in the code attached.



Above is the scatter plot between price and carat.

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?

We have earlier seen that null values were present in "depth" column. I have imputed the null values using the average. After null values imputation we can see that the null values are zero.

Yes, scaling for the given data is mandatory because the given data are all of different values. This means that there are discrepancies with the data and we must make sure that the data is scaled and made sure that all the values are on the same scale. Hence, I've used the Standard Scaler method and done the scaling part for the data.

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

Encoding of the data has been done and the data split is also done in the 70:30 ratio and I have also applied the linear regression algorithm. You can find it in the code attached.

Performance metrics are also done using Rsquare and RMSE. You can find the process and coding part in the code.

R square on training data is 0.9203 which means 92% of the variation in the price is explained by the predictors in the model for train set and R square on test data 0.923 which is almost 92.3% without much variation between training and testing data.

When it comes to RMSE, the root-mean-square deviation (RMSD) or root-mean-square error (RMSE) is a frequently used measure of the differences between values (sample or population values) predicted by a model or an estimator and the values observed. RMSD is the square root of the average of squared errors.

RMSE on training data 0.281

RMSE on testing data 0.279

Below I've attached the final summary using the formula.api function which explains complete p values, t values and also the R square, F score.

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

From the above analysis, I can say that one of the main factor for the pricing of the stone has to be the carat. The carat weight is one of the important variable for predicting the price. The next important variable on predicting the price is the variable X which is the length of the stone in mm. We can clearly say from the analysis that X, Y, Z are also all important variable in predicting the price. The length, width and the height of the stone plays a crucial role in predicting the price.

Hence I would say that the variables "carat", "X","Y","Z" are the most important variables in predicting the prices.

Problem 2: Logistic Regression and LDA

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.

Below is the screenshot of the dataset after reading the data. It shows the first 5 rows and all the columns present in the data.

```
In [5]: df.head()
```

Out[5]:

	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 data consists of 7 columns and 872 rows. The names of the columns are Holliday Package, Salary, age, educ, no young children, no older children and foreign.

```
In [6]: df.info()
```

```
<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
```

Above screenshots gives us an idea of the datatypes present in our data. We can see 2 columns are of object data type and other are integer datatype.

```
In [6]: df.describe(include='all').T
```

Out[6]:

	count	unique	top	freq	mean	std	min	25%	50%	75%	max
Holliday_Package	872	2	no	471	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Salary	872	NaN	NaN	NaN	47729.2	23418.7	1322	35324	41903.5	53469.5	236961
age	872	NaN	NaN	NaN	39.9553	10.5517	20	32	39	48	62
educ	872	NaN	NaN	NaN	9.30734	3.03626	1	8	9	12	21
no_young_children	872	NaN	NaN	NaN	0.311927	0.61287	0	0	0	0	3
no_older_children	872	NaN	NaN	NaN	0.982798	1.08679	0	0	1	2	6
foreign	872	2	no	656	NaN	NaN	NaN	NaN	NaN	NaN	NaN

I have used the describe function which gives us the descriptive stats of the data present. It gives us the Max values, Min values, Standard Deviation, Mean and median for the data present.

Like I said earlier 2 columns are of Object datatype and therefore we do not get any descriptive stats for those columns.

Check for Null Values

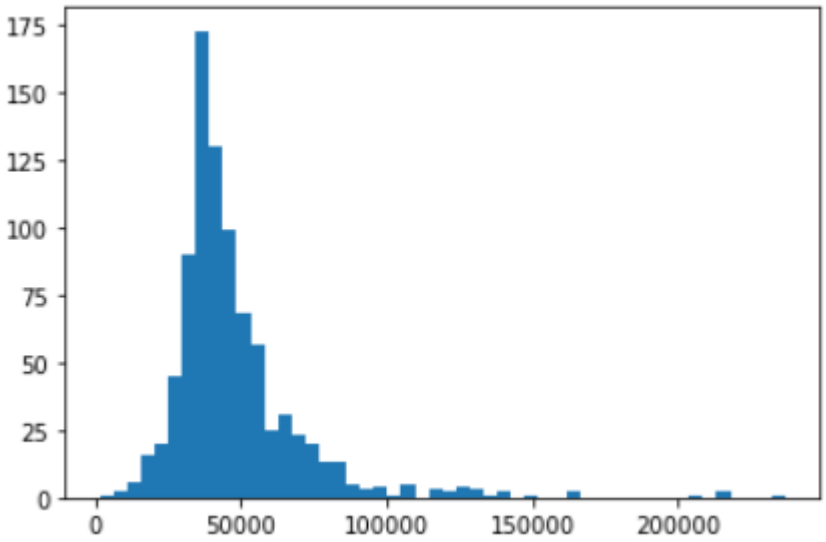
```
In [8]: df.isnull().sum()
```

Out[8]:

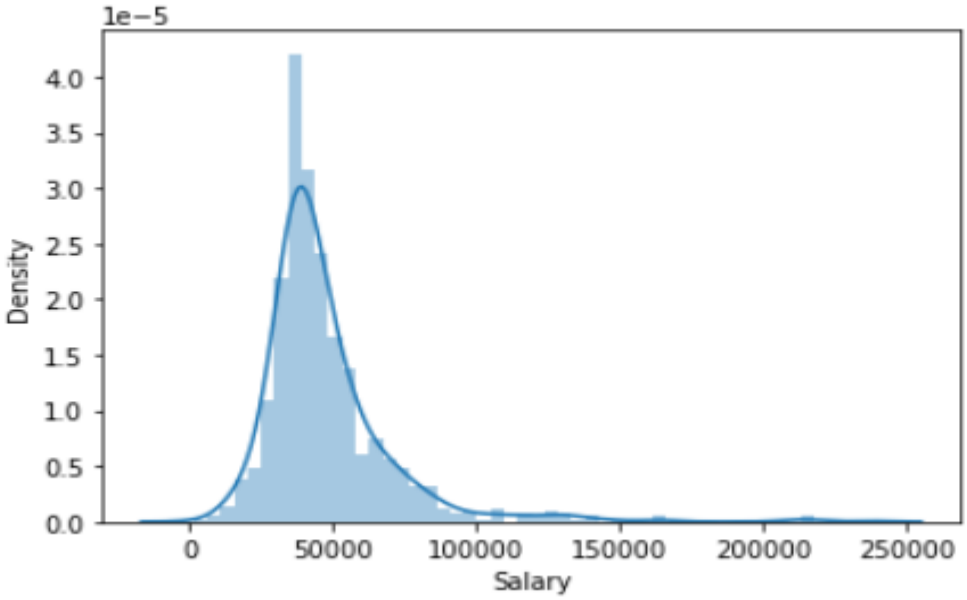
```
Holliday_Package     0
Salary               0
age                  0
educ                 0
no_young_children    0
no_older_children    0
foreign              0
dtype: int64
```

I've checked for the null values and as you can see from the screenshot there are no null values present in the given data.

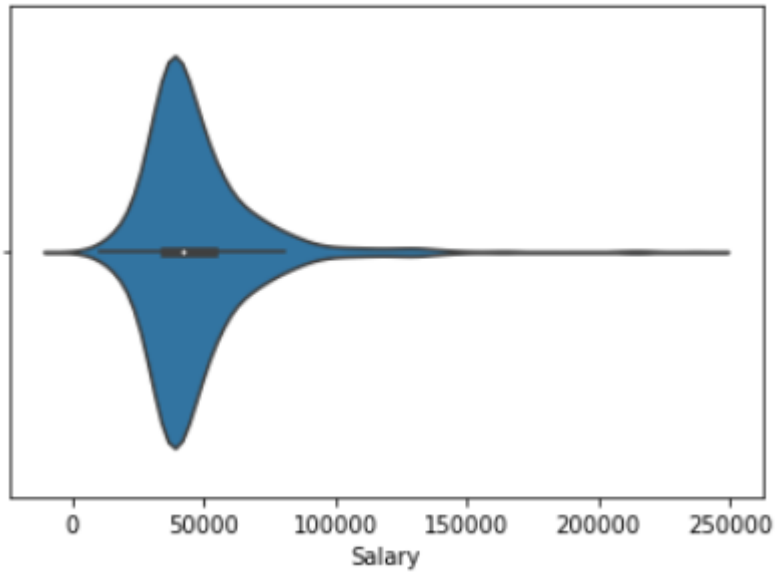
Univariate analysis has to be done on a single variable present in the data. Hence I've taken the Salary column to do the univariate analysis.



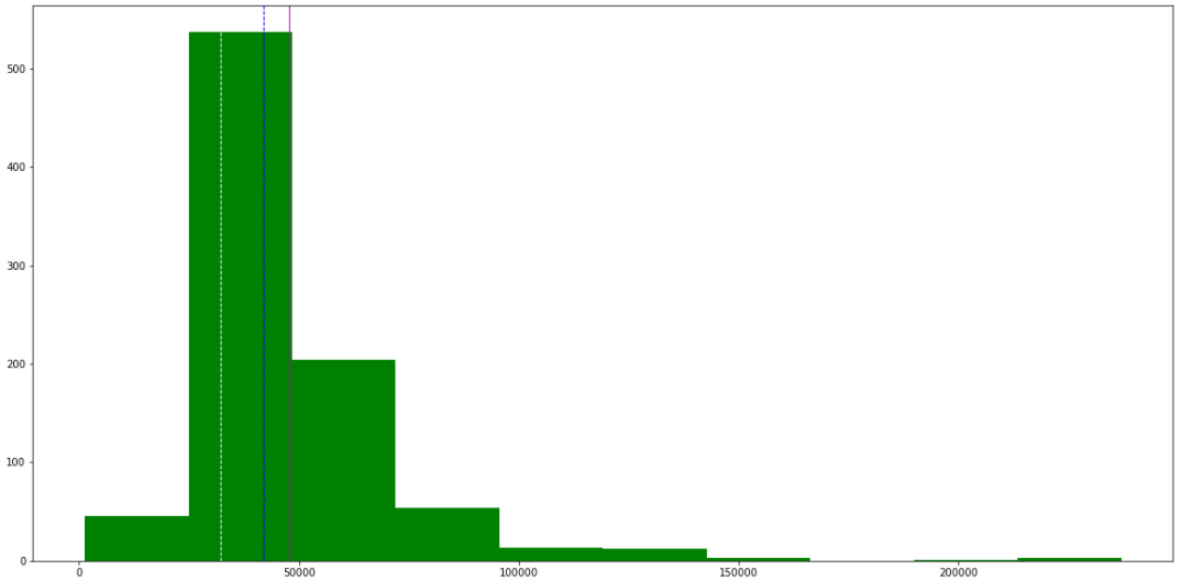
The above is the Histogram Plot for the column Salary.



The above figure gives us the distribution plot and you can see how the data is distributed using the distribution plot.

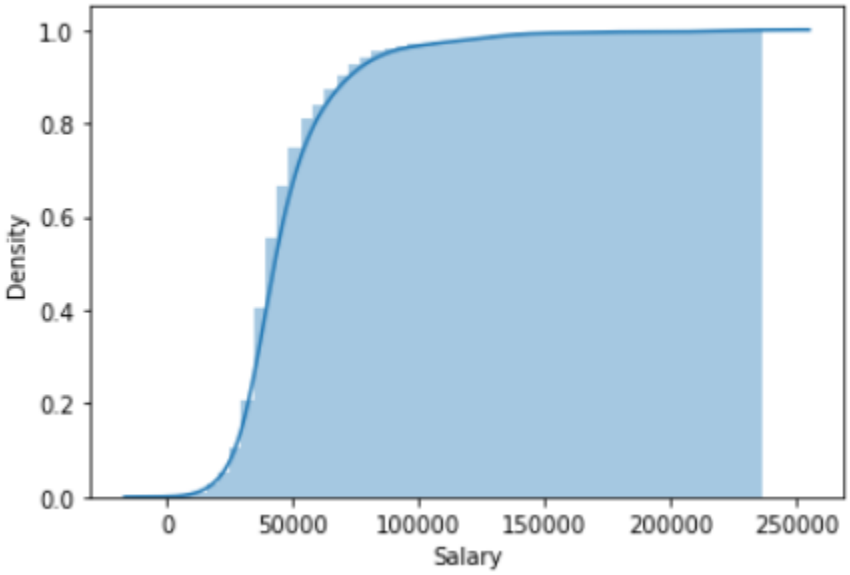


The above figure gives a violin plot.



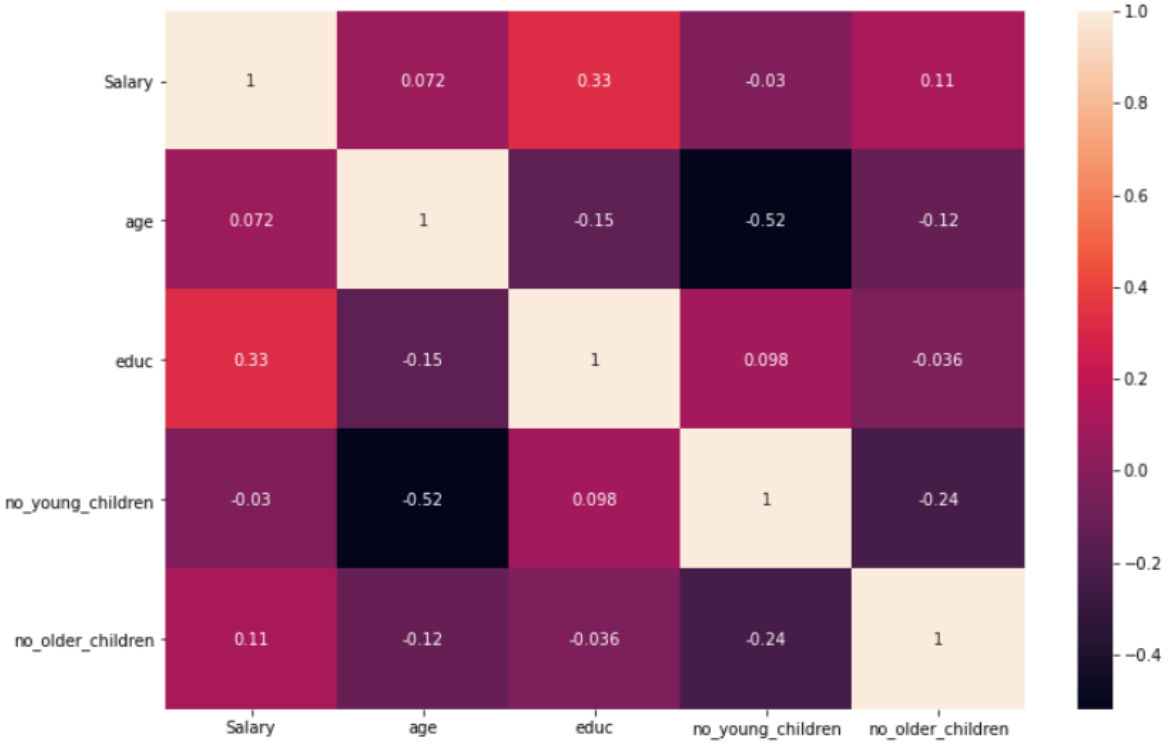
A frequency distribution shows how often each different value in a set of data occurs. A histogram is the most commonly used graph to show frequency distributions. It looks very much like a bar chart, but there are important differences between them.

The above is the Histogram for the salary column.

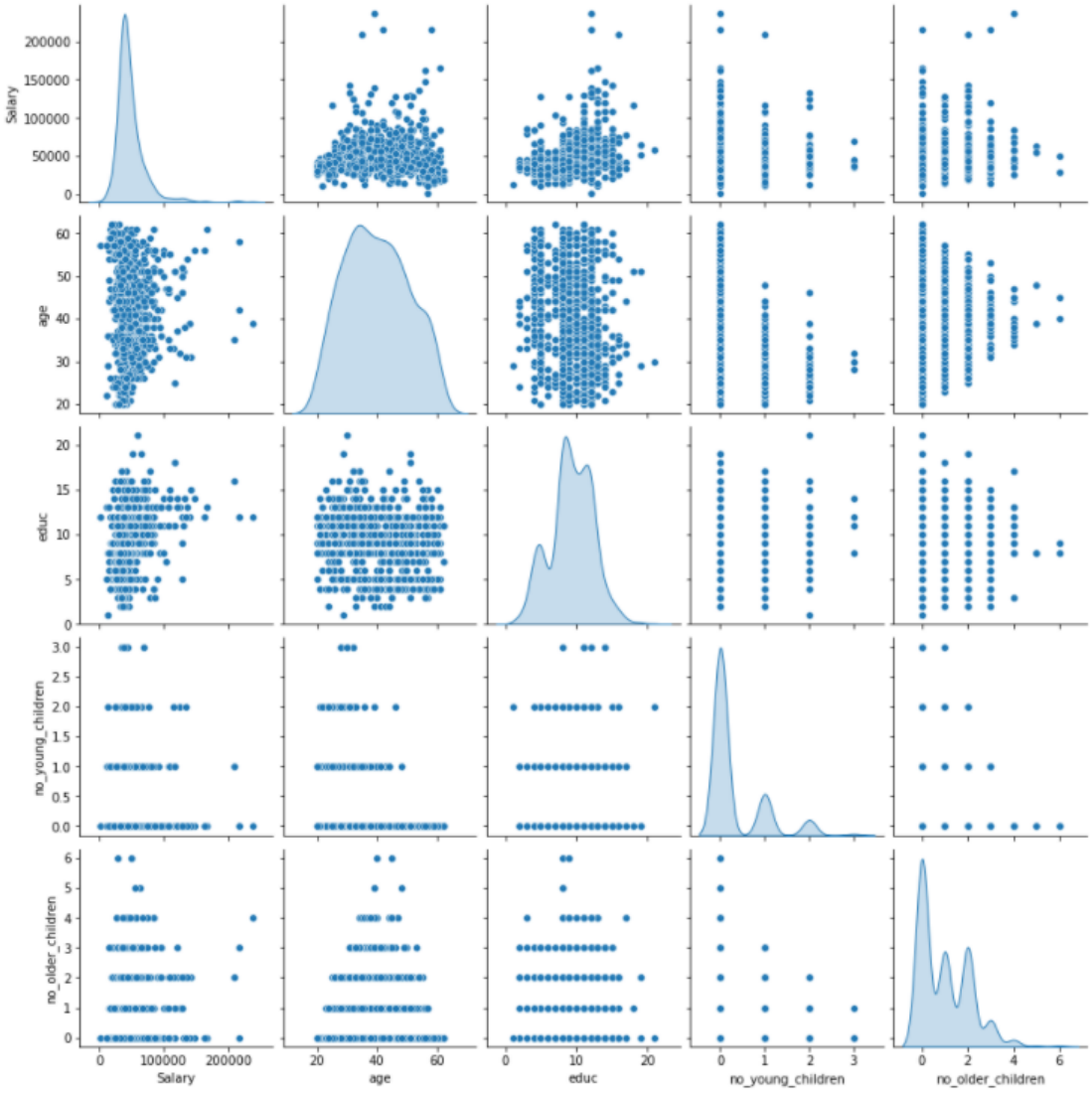


Same Univariate analysis is done on a different column and you can find all the data of analysis in the code.

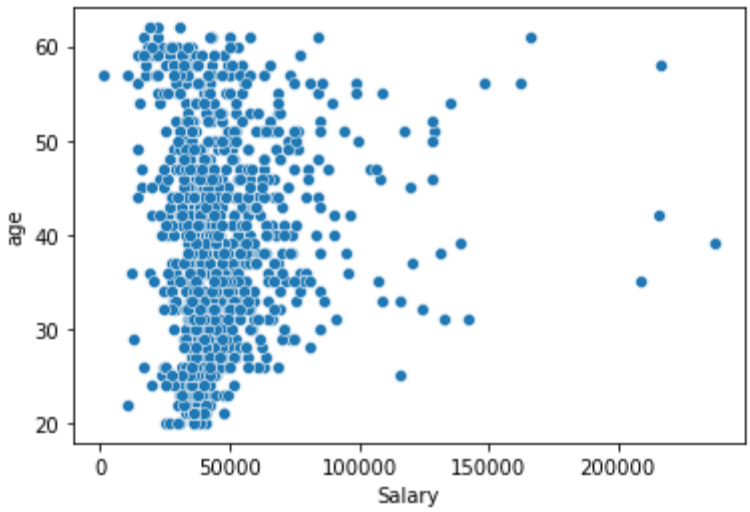
Bivariate analysis:



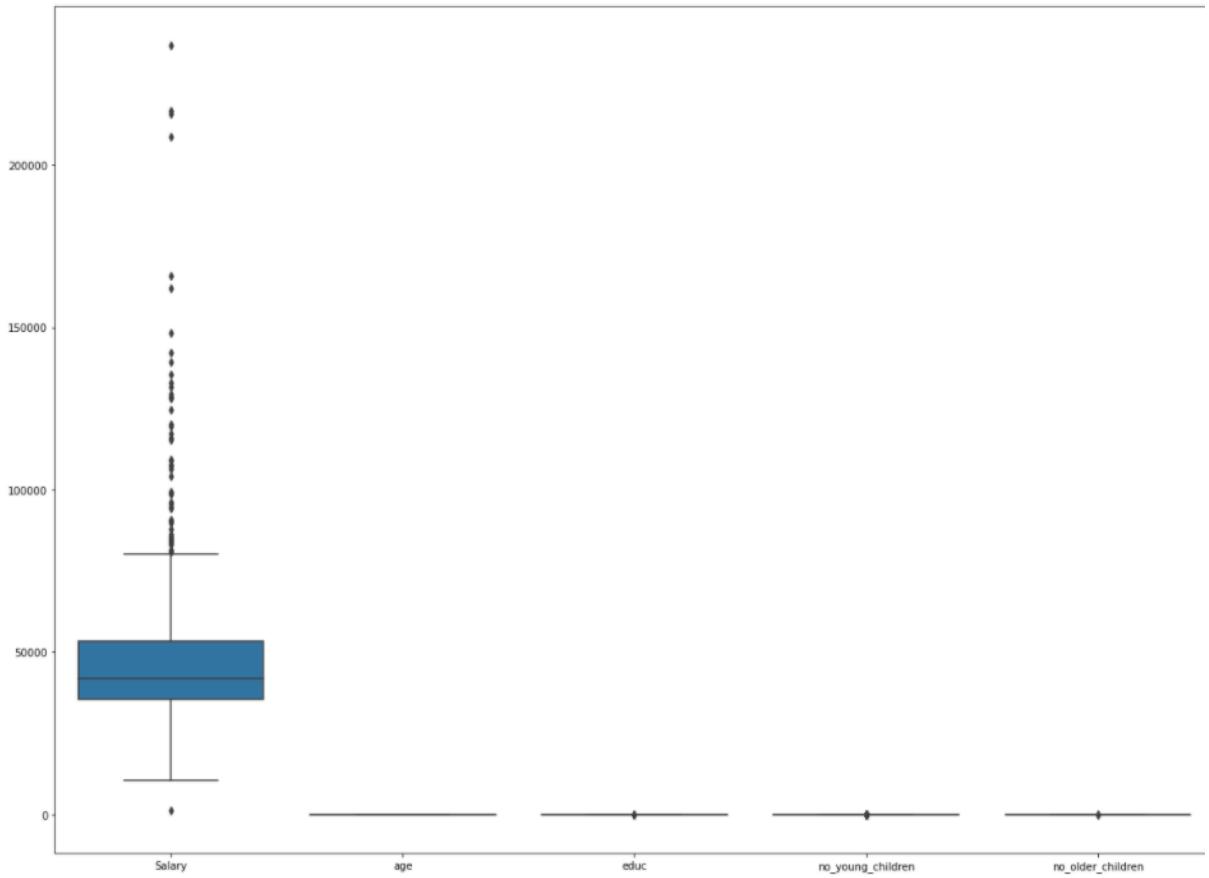
The above figure shows the corelation between all the columns and highly corelated columns will have values closer to 1 and less corelated values will be values far from 1.



The above is a similar corelation plot which shows the relation between the columns.



The scatter plot above gives a figure between age and salary. We can see that most of the data is scatter around 30000 to 50000. However there are few outliers where the salary higher and we can see that on scatter plot.



The box plot gives us an idea about the outliers. We can see that the salary column has outliers present in the data. However it will not effect out analysis because these are genuine outliers.

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).

I have not done the scaling for the data. Encoding of the data has been done and I've split the data into 70:30 ratio and it can all be seen in the coding part.

I've also applied the Logistic regression and Linear discriminant analysis algorithms. The outcome can be seen in the code. I will attach a screenshot of the splitting the data in 70:30 Train and Test below.

```
In [56]: # Copy all the predictor variables into X dataframe
         X = df.drop('Holliday_Package_yes', axis=1)

         # Copy target into the y dataframe.
         y = df[['Holliday_Package_yes']]
```

```
In [57]: # Split X and y into training and test set in 70:30 ratio
         X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30 , random_state=1,stratify=y)
```

```
In [64]: # Fit the Logistic Regression model
         model = LogisticRegression(solver='newton-cg',max_iter=10000,penalty='none',verbose=True,n_jobs=2)
         model.fit(X_train, y_train)

         C:\Users\user\anaconda3\lib\site-packages\sklearn\utils\validation.py:72: DataConversionWarning: A column-vector y was passe
         d when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
           return f(**kwargs)
         [Parallel(n_jobs=2)]: Using backend LokyBackend with 2 concurrent workers.
         [Parallel(n_jobs=2)]: Done   1 out of   1 | elapsed:    0.9s finished

Out[64]: LogisticRegression(max_iter=10000, n_jobs=2, penalty='none', solver='newton-cg',
                            verbose=True)
```

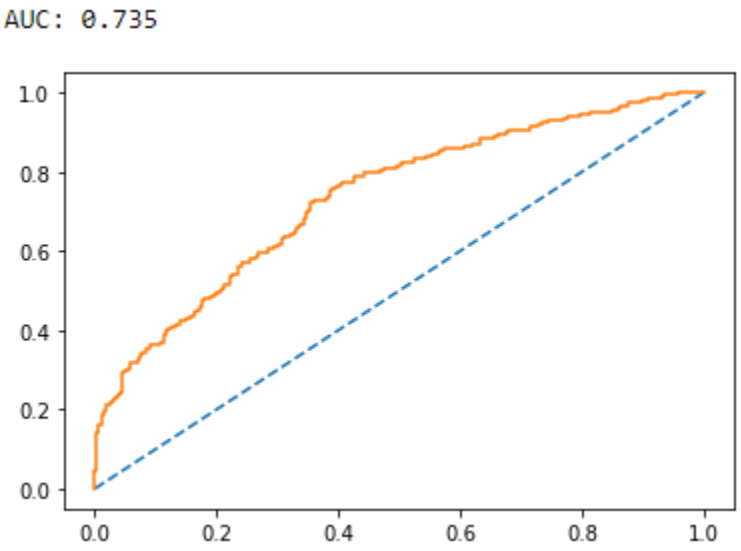
2.3 Performance Metrics: Check the performance of Predictions on Train and Test sets using Accuracy, Confusion Matrix, Plot ROC curve and get ROC_AUC score for each model Final Model: Compare Both the models and write inference which model is best/optimized.

Performance of predictions on train and test sets using accuracy are below using Logistic regression:

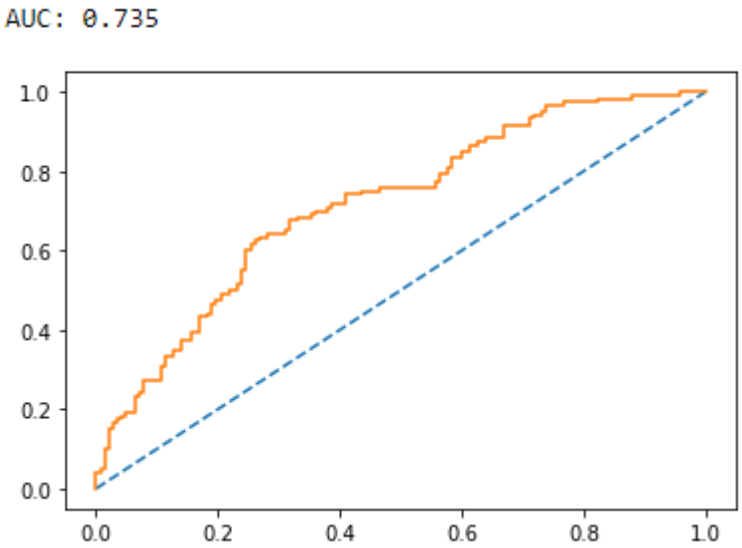
67% accuracy on training data has been observed.

65% accuracy on testing data has been observed.

AUC and ROC for training data is below:



AUC and ROC for testing data is below:

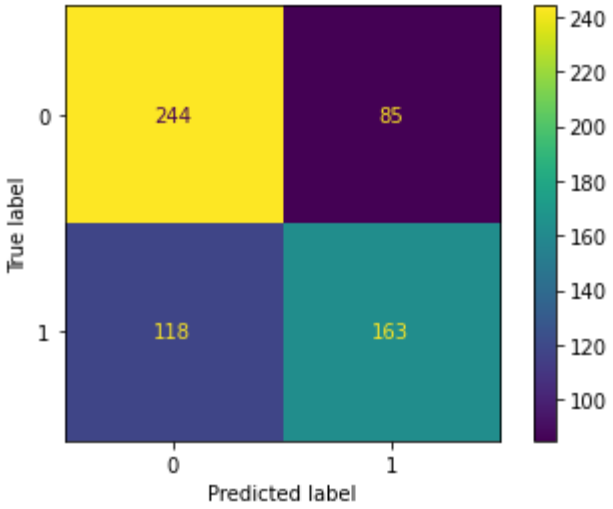


Confusion Matrix on Training Data:

```
confusion_matrix(y_train, ytrain_predict)
```

```
array([[244,  85],
       [118, 163]], dtype=int64)
```

```
plot_confusion_matrix(model,X_train,y_train);
```

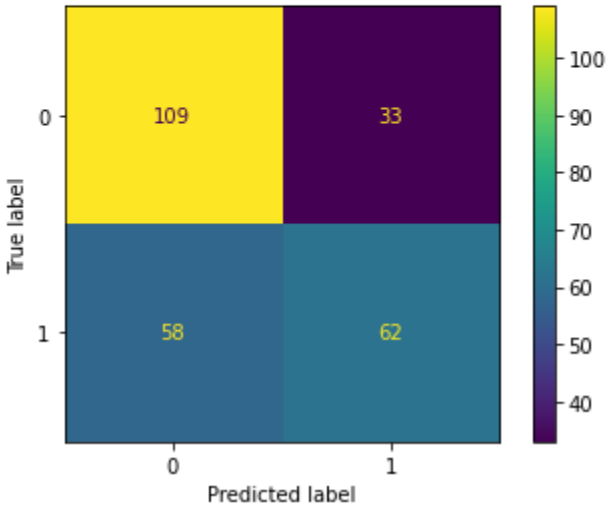


Confusion matrix on Testing data:

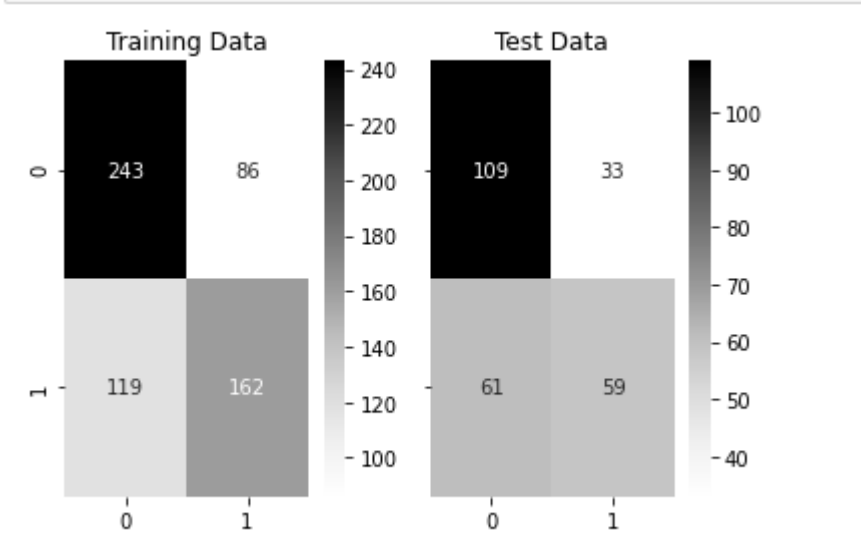
```
confusion_matrix(y_test, ytest_predict)
```

```
array([[109,  33],
       [ 58,  62]], dtype=int64)
```

```
plot_confusion_matrix(model,X_test,y_test);
```



Performance of predictions on train and test sets using accuracy are below using Linear Discriminant analysis:



Above is the confusion matrix for Training and testing data.

Classification Report for training and testing data is below:

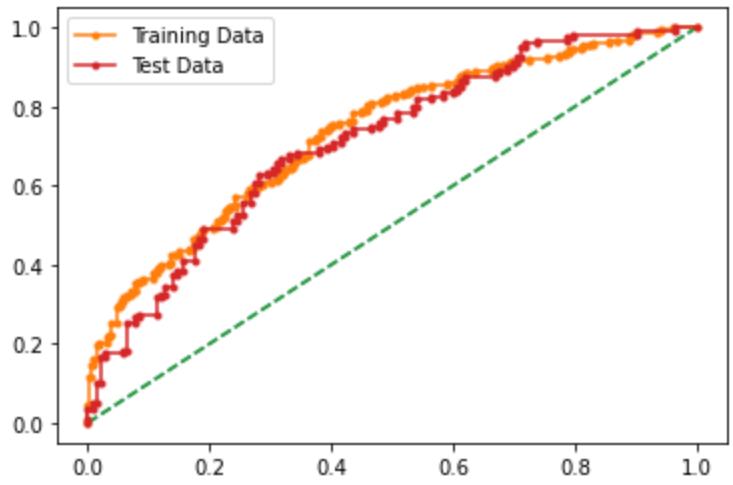
Classification Report of the training data:

	precision	recall	f1-score	support
0	0.67	0.74	0.70	329
1	0.65	0.58	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 test data:

	precision	recall	f1-score	support
0	0.64	0.77	0.70	142
1	0.64	0.49	0.56	120
accuracy			0.64	262
macro avg	0.64	0.63	0.63	262
weighted avg	0.64	0.64	0.63	262

```
AUC for the Training Data: 0.733
AUC for the Test Data: 0.714
```



Above figure gives the AUC score for the training and testing data. ROC curve is also shown above.

Comparing both the models we can clearly see that the LDA algorithm is more effective when you check for the predictions. We can clearly see LDA is best optimized.