PREDECTIVE MODELING

PROJECT ON LINEAR REGRESSION, LOGISTIC REGRESSION &LDA

PREDECTIVE MODELLING

PROJECT ON LINEAR REGRESSION, LOGISTIC REGRESSION & LDA

CASE STUDY I

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

Data Dictionary:

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

Dataset for Problem 1: cubic_zirconia.csv

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.

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?

1.3. Encode the data (having string values) for Modeling. 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

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

SOLUTION TO THE QUESTIONS

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.

Import all Libraries:

We have to import all the relevant libraries in Jupyter notebook before importing the dataset.

```
from sklearn.datasets import load_boston
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import statsmodels.api as sm
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from statsmodels.stats.outliers_influence import variance_inflation_factor
```

Read the dataset:

Import the dataset to the Jupiter note book.

```
df=pd.read_csv("cubic_zirconia.csv")
```

Preview the dataset and see all dataset is loaded correctly

	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

In this dataset we do not require the first column "Unnamed: 0" for doing the analysis. Hence we will drop the column before stepping into further steps. The dataset has 10 columns and 26967 rows.

The data set has the below data types:

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

The dataset shows that for the variable "depth" there are 697 null values. We have to treat those rows where no values are picked by replacing the mean of the column

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

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

We could find one the NA values are filled up with the mean of the column of the dataset. Once these are done we could find there are no NA values available in the dataset.

```
for column in df.columns:
    if df[column].dtype != 'object':
        mean = df[column].mean()
        df[column] = df[column].fillna(mean)

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

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

Type of the dataset:

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

The EDA of the dataset:

	carat	cut	color	clarity	depth	table	x	y	z	price
count	26967.000000	26967	26967	26967	26967.000000	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	1.394481	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	61.100000	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

We could find in this dataset there are certain data type as "Object". Hence the Cut, color and clarity of the diamond values are not picked up. Hence we need to convert the categorical data into variable data. We will be checking the unique values for 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 will be converting these categorical variables to dummy variables by splitting the unique values received for cut, color and clarity.

```
df = pd.get_dummies(df, columns=['cut','color','clarity'],drop_first=True)
```

	carat	depth	table	x	y	z	price	cut_Good	cut_Ideal	cut_Premium	...	color_H	color_I	color_J	clarity_IF	clarity_SI1	clarity_SI2	clarity_VS1
0	0.30	62.1	58.0	4.27	4.29	2.66	499	0	1	0	...	0	0	0	0	1	0	0
1	0.33	60.8	58.0	4.42	4.46	2.70	984	0	0	1	...	0	0	0	1	0	0	0

2 rows × 24 columns

Once the dummy variables are added, the shape of the data will be 26967 rows and 24 rows.

Check the Mean, Standard deviation, min and Max of the new dataset.

	count	mean	std	min	25%	50%	75%	max
carat	26933.0	0.793298	0.462127	0.200	0.40	0.70	1.05	2.025
depth	26933.0	61.749043	1.218503	59.000	61.10	61.80	62.50	64.600
table	26933.0	57.435544	2.157119	51.500	56.00	57.00	59.00	63.500
x	26933.0	5.729323	1.126175	1.950	4.71	5.69	6.55	9.310
y	26933.0	5.731255	1.118155	1.965	4.71	5.70	6.54	9.285
z	26933.0	3.536928	0.696753	1.190	2.90	3.52	4.04	5.750
price	26933.0	3735.832213	3468.207359	326.000	945.00	2375.00	5356.00	11972.500
cut_Good	26933.0	0.090410	0.286773	0.000	0.00	0.00	0.00	1.000
cut_Ideal	26933.0	0.401181	0.490147	0.000	0.00	0.00	1.00	1.000
cut_Premium	26933.0	0.255671	0.436246	0.000	0.00	0.00	1.00	1.000
cut_Very Good	26933.0	0.223778	0.416782	0.000	0.00	0.00	0.00	1.000
color_E	26933.0	0.182527	0.386285	0.000	0.00	0.00	0.00	1.000
color_F	26933.0	0.175361	0.380283	0.000	0.00	0.00	0.00	1.000
color_G	26933.0	0.209891	0.407238	0.000	0.00	0.00	0.00	1.000
color_H	26933.0	0.152044	0.359070	0.000	0.00	0.00	0.00	1.000
color_I	26933.0	0.102662	0.303523	0.000	0.00	0.00	0.00	1.000
color_J	26933.0	0.053466	0.224965	0.000	0.00	0.00	0.00	1.000
clarity_IF	26933.0	0.033082	0.178854	0.000	0.00	0.00	0.00	1.000
clarity_SI1	26933.0	0.243753	0.429353	0.000	0.00	0.00	0.00	1.000
clarity_SI2	26933.0	0.169458	0.375163	0.000	0.00	0.00	0.00	1.000
clarity_VS1	26933.0	0.151747	0.358782	0.000	0.00	0.00	0.00	1.000
clarity_VS2	26933.0	0.226228	0.418396	0.000	0.00	0.00	0.00	1.000
clarity_VVS1	26933.0	0.068281	0.252231	0.000	0.00	0.00	0.00	1.000

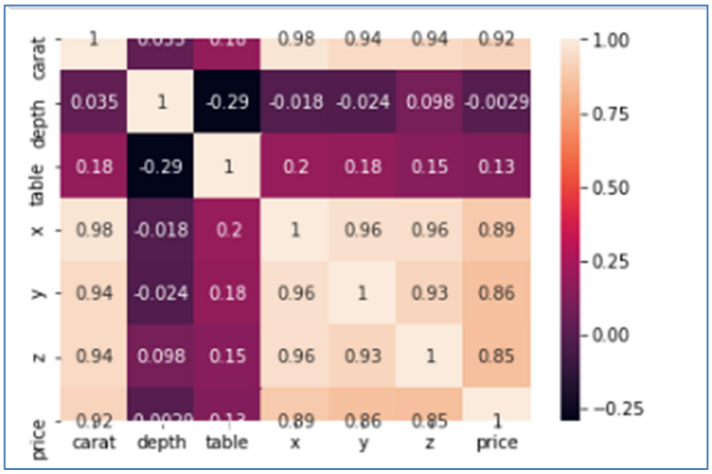
Check for any duplicates in dataset. If any duplicates are available remove those lines from the dataset.

```
print('Before',df.shape)
df.drop_duplicates(inplace=True)
print('After',df.shape)
```

```
Before (26967, 24)
After (26933, 24)
```

Heat Map:

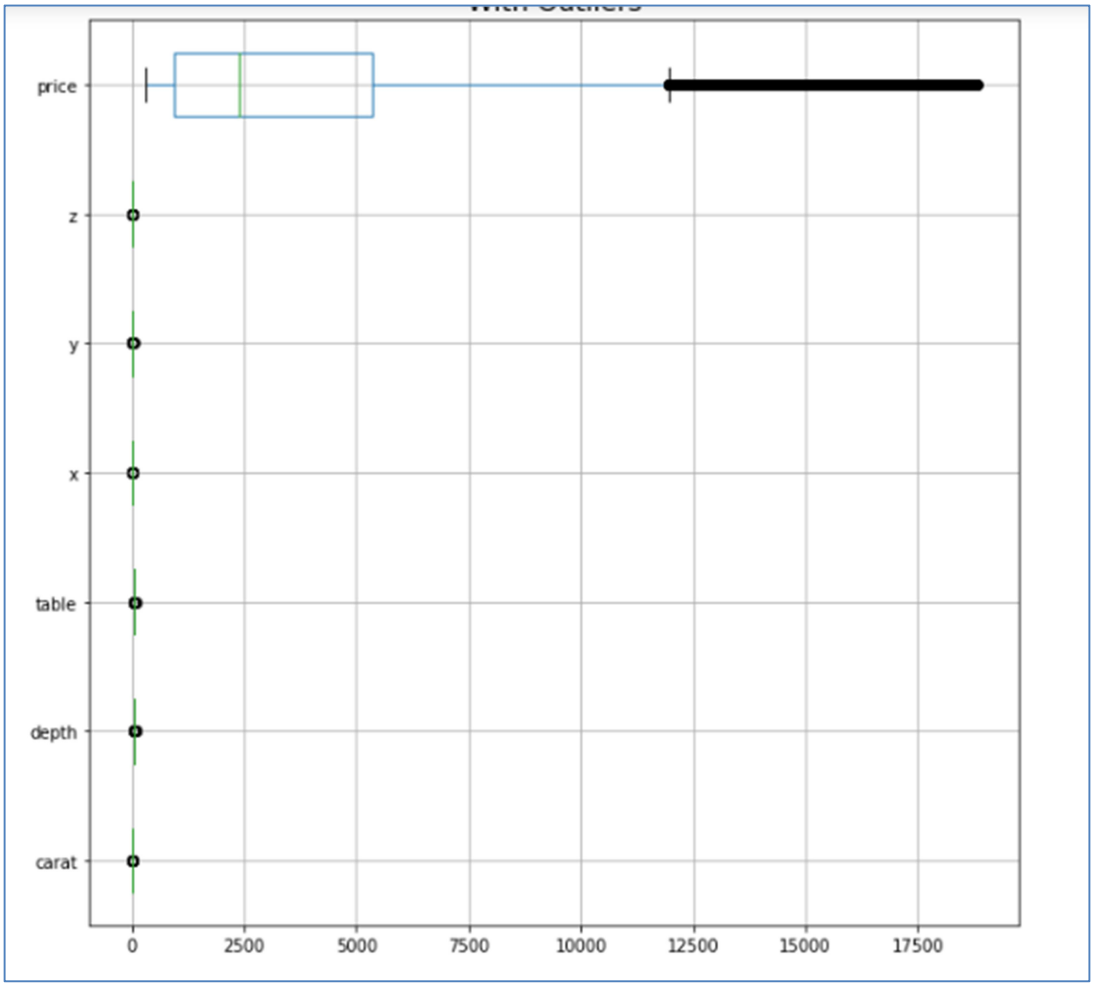
The heat map is used to see the correlation of the dataset.



The above heat map shows that there is a positively high correlation with the length, width, height, carat and price of the diamond. There are negative correlation in terms of the variable depth and table of the dataset.

Outliers and treatment of outliers:

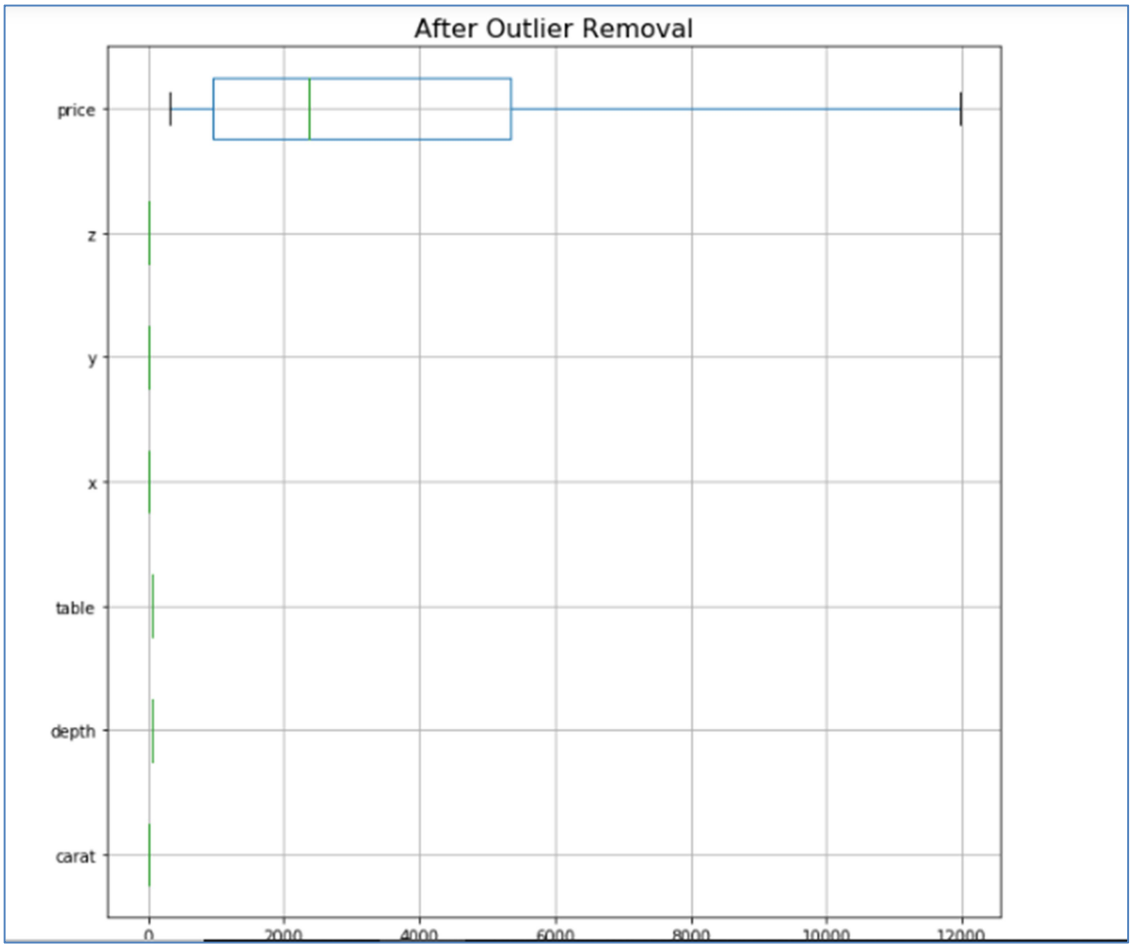
We will be checking if the dataset is having any outliers or not. The below graph shows there are outliers available for the variable and that need to be treated.



Treatment of the outliers:

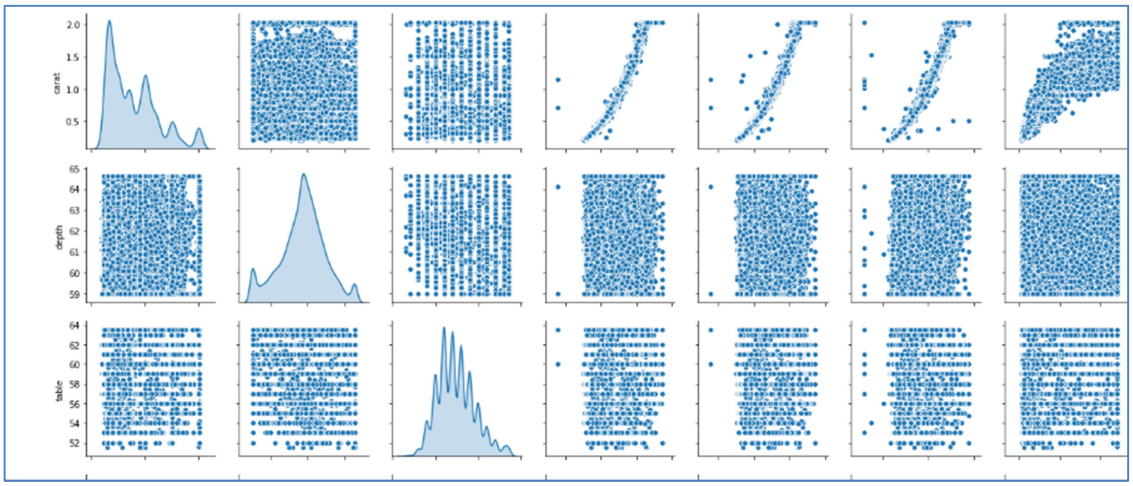
```
def remove_outlier(col):
    sorted(col)
    Q1,Q3=np.percentile(col,[25,75])
    IQR=Q3-Q1
    lower_range= Q1-(1.5 * IQR)
    upper_range= Q3+(1.5 * IQR)
    return lower_range, upper_range
```

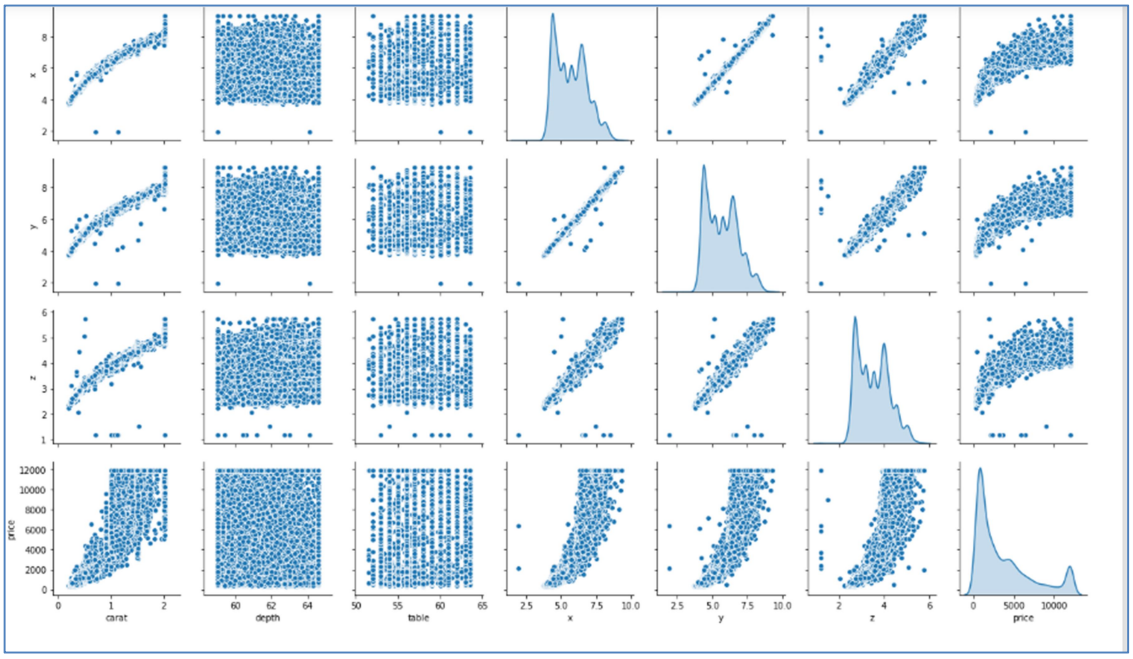
```
for column in df[cont].columns:
    lr,ur=remove_outlier(df[column])
    df[column]=np.where(df[column]>ur,ur,df[column])
    df[column]=np.where(df[column]<lr,lr,df[column])
```



Scatter Plot:

The scatter plot helps us understanding the skewedness of the data with respect of the variables carat, depth, table, length, height and weight, price of the diamonds.





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?

The variable depth of the diamond was having missing values for 697 values for which they are in placed with the mean of the column.

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

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

```
for column in df.columns:
    if df[column].dtype != 'object':
        mean = df[column].mean()
        df[column] = df[column].fillna(mean)

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

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

The dataset shared to us was having three categorical variables. They are the cut of the diamond, color of the diamond and clarity of the diamond. While doing the linear regression we need to convert these categorical data to variable data. In order to convert the categorical we will use the concept of dummy variables.

```
for column in df.columns:
    if df[column].dtype == 'object':
        print(column.upper(),': ',df[column].nunique())
        print(df[column].value_counts().sort_values())
        print('\n')
```

```
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
```

Coverting the Categorical to Dummy Variables

```
: df = pd.get_dummies(df, columns=['cut','color','clarity'],drop_first=True)
```

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

z	price	cut_Good	cut_Ideal	cut_Premium	...	color_H	color_I	color_J	clarity_IF	clarity_SI1	clarity_SI2	clarity_VS1	clarity_VS2	clarity_VVS1	clarity_VVS2
6	499	0	1	0	...	0	0	0	0	1	0	0	0	0	0
0	984	0	0	1	...	0	0	0	1	0	0	0	0	0	0

For the new dataset we will again checking the null values present in the dataset. We could find that there are no null values present in the dataset. The dataset was having "0" is X, Y and Z variable (length, height and weight) of the diamond. Out of which the weight of the diamond was having the maximum number of zeros in the column Z. While checking the duplicates these lines are picked up and they are removed from the dataset. There are zeros updates when the X and Y column are updated. In that case we will leave the data as such. We will not be scaling the data for linear regression as this would affect our final result. We will be getting identical regression results if the data is scaled. Hence we are not scaling this dataset.

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

```
carat            0
depth            0
table            0
x                0
y                0
z                0
price            0
cut_Good         0
cut_Ideal        0
cut_Premium      0
cut_Very Good    0
color_E          0
color_F          0
color_G          0
color_H          0
color_I          0
color_J          0
clarity_IF       0
clarity_SI1      0
clarity_SI2      0
clarity_VS1      0
clarity_VS2      0
clarity_VVS1     0
clarity_VVS2     0
dtype: int64
```

The EDA of the dataset are visualized as below:

	count	mean	std	min	25%	50%	75%	max
carat	26933.0	0.793298	0.462127	0.200	0.40	0.70	1.05	2.025
depth	26933.0	61.749043	1.218503	59.000	61.10	61.80	62.50	64.600
table	26933.0	57.435544	2.157119	51.500	56.00	57.00	59.00	63.500
x	26933.0	5.729323	1.126175	1.950	4.71	5.69	6.55	9.310
y	26933.0	5.731255	1.118155	1.965	4.71	5.70	6.54	9.285
z	26933.0	3.536928	0.696753	1.190	2.90	3.52	4.04	5.750
price	26933.0	3735.832213	3468.207359	326.000	945.00	2375.00	5356.00	11972.500
cut_Good	26933.0	0.090410	0.286773	0.000	0.00	0.00	0.00	1.000
cut_Ideal	26933.0	0.401181	0.490147	0.000	0.00	0.00	1.00	1.000
cut_Premium	26933.0	0.255671	0.436246	0.000	0.00	0.00	1.00	1.000
cut_Very Good	26933.0	0.223778	0.416782	0.000	0.00	0.00	0.00	1.000
color_E	26933.0	0.182527	0.386285	0.000	0.00	0.00	0.00	1.000
color_F	26933.0	0.175361	0.380283	0.000	0.00	0.00	0.00	1.000
color_G	26933.0	0.209891	0.407238	0.000	0.00	0.00	0.00	1.000
color_H	26933.0	0.152044	0.359070	0.000	0.00	0.00	0.00	1.000
color_I	26933.0	0.102662	0.303523	0.000	0.00	0.00	0.00	1.000
color_J	26933.0	0.053466	0.224965	0.000	0.00	0.00	0.00	1.000
clarity_IF	26933.0	0.033082	0.178854	0.000	0.00	0.00	0.00	1.000
clarity_SI1	26933.0	0.243753	0.429353	0.000	0.00	0.00	0.00	1.000
clarity_SI2	26933.0	0.169458	0.375163	0.000	0.00	0.00	0.00	1.000

1.3 Encode the data (having string values) for Modeling. 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 the data for modeling:

Price is considered as the target variable; hence the train and test splits are done on the basis of target variable.

In order to process the splits we have to use the library from sklearn for test and train split.

```
X = df.drop('price', axis=1)#copy all predictor in X Dataframe
Y = df[['price']]# Target in Y dataframe
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.30 , random_state=1)
```

Linear Regression:

In statistics, **linear regression** is a **linear** approach to modeling the relationship between a scalar response (dependent variable) and one or more explanatory variables (independent variables).

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

Calculate the coefficient values for each variable:

```
for idx, col_name in enumerate(X_train.columns):
    print("The coefficient for {} is {}".format(col_name, regression_model.coef_[0][idx]))
```

```
The coefficient for carat is 9126.93571059195
The coefficient for depth is -15.014396489796695
The coefficient for table is -18.585758559241
The coefficient for x is -1190.2766851747635
The coefficient for y is 837.3564745735196
The coefficient for z is -163.63843722667417
The coefficient for cut_Good is 481.8131839939364
The coefficient for cut_Ideal is 714.646127014467
The coefficient for cut_Premium is 674.7717434404591
The coefficient for cut_Very Good is 606.8965827395969
The coefficient for color_E is -181.90972759472314
The coefficient for color_F is -256.81260630027936
The coefficient for color_G is -429.3811110593348
The coefficient for color_H is -855.9905503632012
The coefficient for color_I is -1323.9291307344315
The coefficient for color_J is -1928.0517124934531
The coefficient for clarity_IF is 4004.0105702429055
The coefficient for clarity_SI1 is 2519.9221438718046
The coefficient for clarity_SI2 is 1684.4607103791645
The coefficient for clarity_VS1 is 3342.5738468842774
The coefficient for clarity_VS2 is 3039.9316871280844
The coefficient for clarity_VVS1 is 3772.2977876502373
The coefficient for clarity_VVS2 is 3757.778854116535
```

Calculate the Intercept:

The **regression** slope **intercept formula**, $b_0 = y - b_1 * x$ is really just an algebraic variation of the **regression** equation, $y' = b_0 + b_1x$ where "b_0" is the y-**intercept** and b_1x is the slope. Once you've found the **linear regression** equation, all that's required is a little algebra to **find** the y-**intercept** (or the slope)

```
intercept = regression_model.intercept_[0]

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

```
The intercept for our model is -1846.1195153134768
```

Performance Matrix:

Calculate the R-Square & RMSE values on the Test and Train data:

R-squared is a statistical measure of how close the data are to the fitted regression line. 0% indicates that the model explains none of the variability of the response data around its mean. 100% indicates that the model explains all the variability of the response data around its mean.

R Square on Train data is showing 94%

```
regression_model.score(X_train, y_train)
```

```
0.9402045565696221
```

Rsquare on Test data is taking up as 94%

```
regression_model.score(X_test, y_test)
```

```
0.9419074462493612
```

The **RMSE** is the square root of the variance of the residuals. It indicates the absolute fit of the model to the data–how close the observed data points are to the model's predicted values. Whereas R-squared is a relative measure of fit, **RMSE** is an absolute measure of fit

RMSE on Train data

```
predicted_train=regression_model.fit(X_train, y_train).predict(X_train)
np.sqrt(metrics.mean_squared_error(y_train,predicted_train))
```

```
847.4553059754451
```

RMSE on Test data

```
predicted_test=regression_model.fit(X_train, y_train).predict(X_test)
np.sqrt(metrics.mean_squared_error(y_test,predicted_test))
```

```
837.3145344145406
```

We can retrieve the same results while using the Stats Model. The below model shares us printed for of regression analysis

OLS Regression Results

Dep. Variable:	price	R-squared:	0.940
Model:	OLS	Adj. R-squared:	0.940
Method:	Least Squares	F-statistic:	1.287e+04
Date:	Sat, 27 Jun 2020	Prob (F-statistic):	0.00
Time:	21:40:36	Log-Likelihood:	-1.5386e+05
No. Observations:	18853	AIC:	3.078e+05
Df Residuals:	18829	BIC:	3.080e+05
Df Model:	23		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-1846.1195	661.115	-2.792	0.005	-3141.964	-550.275
carat	9126.9357	76.198	119.779	0.000	8977.581	9276.290
depth	-15.0144	8.511	-1.764	0.078	-31.697	1.668
table	-18.5858	3.843	-4.836	0.000	-26.118	-11.053
x	-1190.2767	120.503	-9.878	0.000	-1426.474	-954.079
y	837.3565	120.629	6.942	0.000	600.913	1073.799
z	-163.6384	89.794	-1.822	0.068	-339.643	12.366
cut_Good	481.8132	44.224	10.895	0.000	395.131	568.495
cut_Ideal	714.6461	43.069	16.593	0.000	630.227	799.065
cut_Premium	674.7717	41.377	16.308	0.000	593.669	755.875
cut_Very_Good	606.8966	42.338	14.335	0.000	523.910	689.883
color_E	-181.9097	22.750	-7.996	0.000	-226.502	-137.318
color_F	-256.8126	23.219	-11.060	0.000	-302.325	-211.301
color_G	-429.3811	22.528	-19.060	0.000	-473.537	-385.225
color_H	-855.9906	24.109	-35.505	0.000	-903.246	-808.735
color_I	-1323.9291	26.806	-49.389	0.000	-1376.471	-1271.387
color_J	-1928.0517	33.046	-58.345	0.000	-1992.824	-1863.279
clarity_IF	4004.0106	66.185	60.497	0.000	3874.282	4133.739
clarity_SI1	2519.9221	56.628	44.500	0.000	2408.927	2630.917
clarity_SI2	1684.4607	56.909	29.599	0.000	1572.914	1796.008
clarity_VS1	3342.5738	57.751	57.879	0.000	3229.376	3455.772
clarity_VS2	3039.9317	56.955	53.374	0.000	2928.295	3151.569
clarity_VVS1	3772.2978	60.998	61.843	0.000	3652.737	3891.859
clarity_VVS2	3757.7789	59.412	63.250	0.000	3641.326	3874.232

Omnibus:	4751.364	Durbin-Watson:	1.982
Prob(Omnibus):	0.000	Jarque-Bera (JB):	17534.131
Skew:	1.230	Prob(JB):	0.00
Kurtosis:	7.034	Cond. No.	9.12e+03

As per the above report the R Square is 94% and Adjusted R Square is also 94%. The overall P value is less than alpha, so rejecting H0 and accepting Ha that at least 1 regression co-efficient is not 0. Here all regression co-efficient are not 0

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

The final Linear Regression equation is

price = b0 + b1 * carat + b2 * depth + b3 * table + b4 * x + b5 * y + b6 * z + b7 * cut_Good + b8 * cut_Ideal + b9 * cut_Premium + b10 * cut_Very_Good + b11 * color_E + b12 * color_F + b13 * color_G + b14 * color_H + b15 * color_I + b16 * color_J + b17 * clarity_IF + b18 * clarity_SI1 + b19 * clarity_SI2 + b20 * clarity_VS1 + b21 * clarity_VS2 + b22 * clarity_VVS1 + b23 * clarity_VVS2

price=(-1846.12) * Intercept + (9126.94) * carat + (-15.01) * depth + (-18.59) * table + (-1190.28) * x + (837.36) * y + (-163.64) * z + (481.81) * cut_Good + (714.65) * cut_Ideal + (674.77) * cut_Premium + (606.9) * cut_Very_Good + (-181.91) * color_E + (-256.81) * color_F + (-429.38) * color_G + (-855.99) * color_H + (-1323.93) * color_I + (-1928.05) * color_J + (4004.01) * clarity_IF + (2519.92) * clarity_SI1 + (1684.46) * clarity_SI2 + (3342.57) * clarity_VS1 + (3039.93) * clarity_VS2 + (3772.3) * clarity_VVS1 + (3757.78) * clarity_VVS2

Carat is a factor in diamonds that are linked with the length, height and weight of the diamond. The carat of the diamond is decreased to 15 where the depth of the diamond is also decreased to 18. The price of the generally increases with the carat weight. Here the weight of the diamond is less hence the price also decreases. The depth and the table of the diamond were negatively correlated with the length, height and weight of the diamond. This would be one of the reason where the price is decreasing.

.

CASE STUDY: II

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.

Dataset for Problem 2: Holiday_Package.csv

Data Dictionary:

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

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.

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).

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.

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

SOLUTION TO THE QUESTIONS

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.

Load the Libraries:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
from sklearn.metrics import roc_auc_score,roc_curve,classification_report,confusion_matrix
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.metrics import confusion_matrix
from sklearn.preprocessing import scale
```

```
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")
```

Import the Dataset:

```
df=pd.read_csv("Holiday_Package.csv")
```

```
df.head()
```

	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

EDA:

The dataset has the column named "Unnamed", which need to be removed.

```
df=df.drop('Unnamed: 0',axis=1)
```

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

	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

Check the data type of the dataset:

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

The dataset is having the target variable (Holiday Packages) as object type. The Foreigners in the dataset is also marked as object as these both variables are in Yes or No option.

These object data types need to convert to integer data type for processing the LDA and the Logistic Regression.

Check the Null Values:
There are no null values present in the dataset. The dataset contains 872 rows and 7 columns.

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

Check for Duplicates:

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

```
Number of duplicate rows = 0
(872, 7)
```

There are no duplicate lines in the dataset that need to be removed.

Check the unique counts in the Object:

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


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

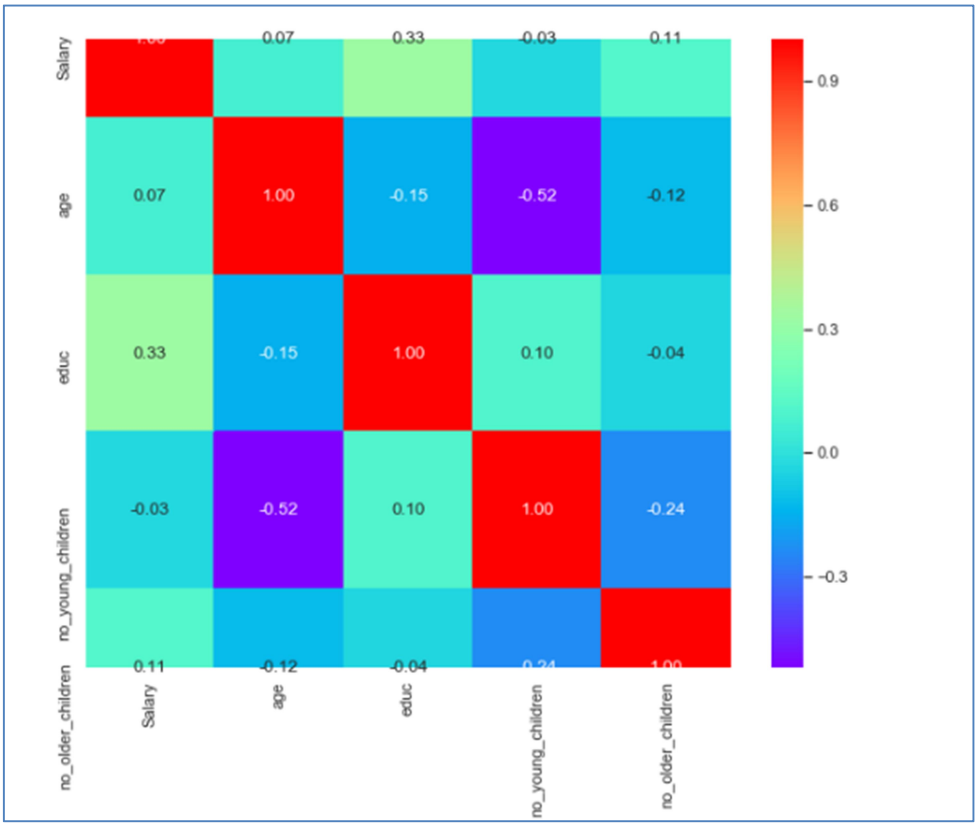
There are 471 employees has opted for Holiday Packages and 401 employees has no opted for Holiday Packages. There are 216 employees that are foreigners.

Check the description of the dataset:

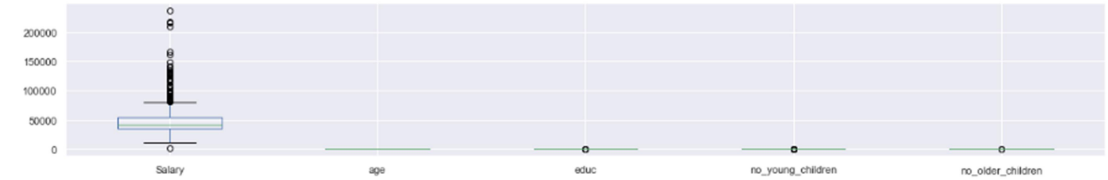
	count	mean	std	min	25%	50%	75%	max
Salary	872.0	47729.172018	23418.668531	1322.0	35324.0	41903.5	53469.5	236961.0
age	872.0	39.955275	10.551675	20.0	32.0	39.0	48.0	62.0
educ	872.0	9.307339	3.036259	1.0	8.0	9.0	12.0	21.0
no_young_children	872.0	0.311927	0.612870	0.0	0.0	0.0	0.0	3.0
no_older_children	872.0	0.982798	1.086786	0.0	0.0	1.0	2.0	6.0

Descriptive Analysis:

Heat Map: The correlation of the data as studied using the Heat Map. The correlation is calculated with the variable "Salary, Age, Education, Young Children and Older Children".

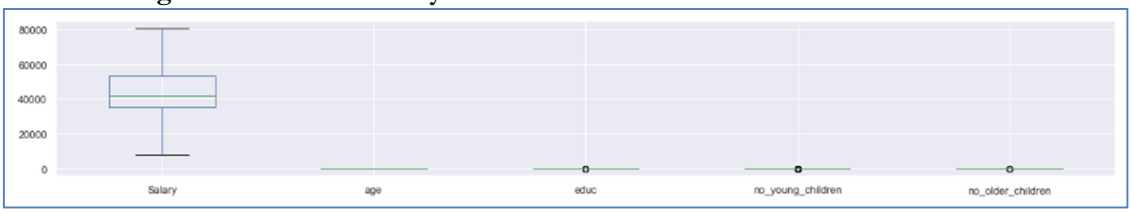


Check for the Outliers:



There are outliers in the variable "Salary" and we will be treating only the outlier for Salary variable, as treatment of outliers are not necessary as Treating the outliers by converting them to min/max values will cause most variables to have values to be the same. So, outliers are not treated in this case.

After Treating the Outliers for Salary:



Converting to Categorical Variable:

As we have Holiday Package and the Foreigners as Object data type. This dependent variable will be converted to categorical variable.

```
feature: Holliday_Package
[no, yes]
Categories (2, object): [no, yes]
[0 1]


feature: foreign
[no, yes]
Categories (2, object): [no, yes]
[0 1]
```

Check the data type:

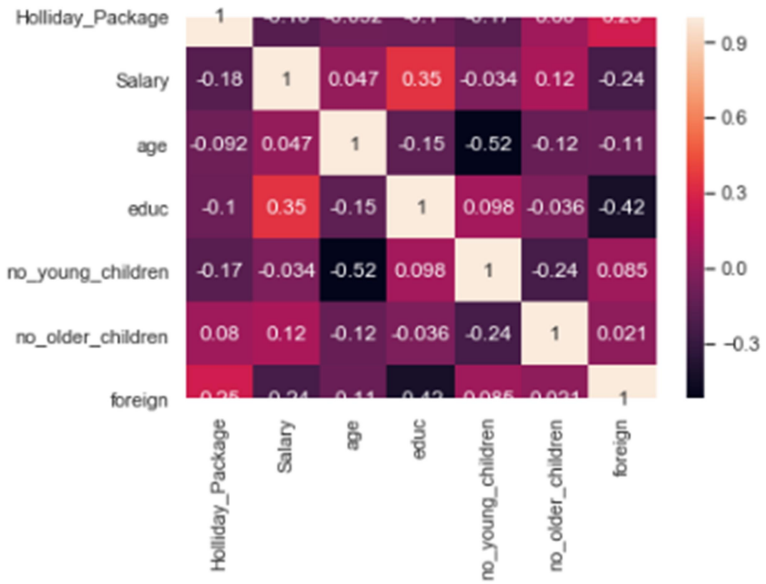
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 872 entries, 0 to 871
Data columns (total 7 columns):
Holliday_Package     872 non-null int8
Salary               872 non-null float64
age                  872 non-null int64
educ                 872 non-null int64
no_young_children    872 non-null int64
no_older_children    872 non-null int64
foreign              872 non-null int8
dtypes: float64(1), int64(4), int8(2)
memory usage: 35.9 KB
```

We could find that all the data type are in integers and float.

Description of the dataset after converting to categorical variable:

	count	mean	std	min	25%	50%	75%	max
Holliday_Package	872.0	0.459862	0.498672	0.00	0.0	0.0	1.0	1.00
Salary	872.0	45608.336869	15699.745151	8105.75	35324.0	41903.5	53469.5	80687.75
age	872.0	39.955275	10.551675	20.00	32.0	39.0	48.0	62.00
educ	872.0	9.307339	3.036259	1.00	8.0	9.0	12.0	21.00
no_young_children	872.0	0.311927	0.612870	0.00	0.0	0.0	0.0	3.00
no_older_children	872.0	0.982798	1.086786	0.00	0.0	1.0	2.0	6.00
foreign	872.0	0.247706	0.431928	0.00	0.0	0.0	0.0	1.00

The heat map showing the correlation for the whole dataset is as below:



Majority of variables are negatively correlated. The correlation between age of the employee and the children below 6 years of age has no correlation. Similarly the same is with the foreign employees and their education.

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

We have to convert the convert the variables that are in Object to categorical variable.

```
feature: Holliday_Package
[no, yes]
Categories (2, object): [no, yes]
[0 1]


feature: foreign
[no, yes]
Categories (2, object): [no, yes]
[0 1]
```

Train – Test Splits: ***Linear Regression***

The dependent variable for the modeling is considered as "Holiday Packages". Import the package from sklearn to process the test and train split.

```
X = df.drop('Holliday_Package', axis=1)
Y = df['Holliday_Package']
```

Split the data into Train- Test (70:30)

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.30 , random_state=1)
```

Apply Logistic Regression:

We are making some adjustments to the parameters in the Logistic Regression Class to get a better accuracy. Details of which can be found out on the site scikit-learn mentioned below Argument=solver{'newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga'}, default='lbfgs' Algorithm to use in the optimization problem. We are using the Argument here as "newton-cg" as we were getting better accuracy and results

```
model = LogisticRegression(solver='newton-cg',max_iter=10000,penalty='none',verbose=True,n_jobs=2)
model.fit(X_train, y_train)
```

```
[Parallel(n_jobs=2)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=2)]: Done   1 out of   1 | elapsed:    5.2s finished
```

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=10000,
                   multi_class='warn', n_jobs=2, penalty='none',
                   random_state=None, solver='newton-cg', tol=0.0001,
                   verbose=True, warm_start=False)
```

Predicting on the Train and Test:

```
ytrain_predict = model.predict(X_train)
ytest_predict = model.predict(X_test)
```

Getting the prediction class and Probs:

```
ytest_predict_prob=model.predict_proba(X_test)
pd.DataFrame(ytest_predict_prob).head()
```

	0	1
0	0.773610	0.226390
1	0.272671	0.727329
2	0.902697	0.097303
3	0.958479	0.041521
4	0.512982	0.487018

Split the data for LDA

```
X = df.drop('Holliday_Package', axis=1)
Y = df['Holliday_Package']
```

Apply LDA

```
clf = LinearDiscriminantAnalysis()
model=clf.fit(X,Y)
model
```

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

```
clf = LinearDiscriminantAnalysis()
model=clf.fit(X_train,y_train)
model
```

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

We will be even doing the test train splitting for LDA that would be helpful in getting an optimal result for identifying the best-fit model.

Predict the class:

```
pred_class = model.predict(X)
df['Prediction'] = pred_class
```

	Holliday_Package	Salary	age	educ	no_young_children	no_older_children	foreign	Prediction
0	0	48412.0	30	8	1	1	0	0
1	1	37207.0	45	8	0	1	0	0

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 Matrix for Logistic Regression:

Metrics	Train	Test
Accuracy	67.5%	66%
AUC	73.5%	70%
Precision	56%	56%
Recall	61%	58%
F1 test	69%	61%

Performance Matrix for LDA:

Metrics	Train	Test
Accuracy	67.5%	64%
AUC	74%	70%
Precision	56%	56%
Recall	61%	58%
F1 test	69%	61%

Confusion Matrix for Logistic Regression (Train & Test)

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

```
array([[252,  74],
       [124, 160]], dtype=int64)
```

```
print(classification_report(y_train, ytrain_predict))
```

```
              precision    recall  f1-score   support

           0       0.67      0.77      0.72       326
           1       0.68      0.56      0.62       284

    accuracy                           0.68       610
   macro avg       0.68      0.67      0.67       610
weighted avg       0.68      0.68      0.67       610
```

```
cnf_matrix=confusion_matrix(y_test, ytest_predict)
cnf_matrix
```

```
array([[102,  43],
       [ 52,  65]], dtype=int64)
```

```
print(classification_report(y_test, ytest_predict))
```

```
              precision    recall  f1-score   support

           0       0.66      0.70      0.68       145
           1       0.60      0.56      0.58       117

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

Confusion Matrix for LDA (Train & Test)

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

```
array([[252,  74],
       [124, 160]], dtype=int64)
```

```
confusion_matrix(y_train, ytrain_predict)
print(classification_report(y_train, ytrain_predict))
```

```
              precision    recall  f1-score   support

           0       0.67      0.78      0.72       326
           1       0.69      0.56      0.61       284

    accuracy                           0.68       610
   macro avg       0.68      0.67      0.67       610
weighted avg       0.68      0.68      0.67       610
```

```
cnf_matrix=confusion_matrix(y_test, ytest_predict)
cnf_matrix
```

```
array([[103,  42],
       [ 52,  65]], dtype=int64)
```

```
print(classification_report(y_test, ytest_predict))
```

```
              precision    recall  f1-score   support

           0       0.66      0.71      0.69       145
           1       0.61      0.56      0.58       117

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

Confusion matrix for LDA (without test and train split)

```
confusion_matrix(Y, pred_class)
```

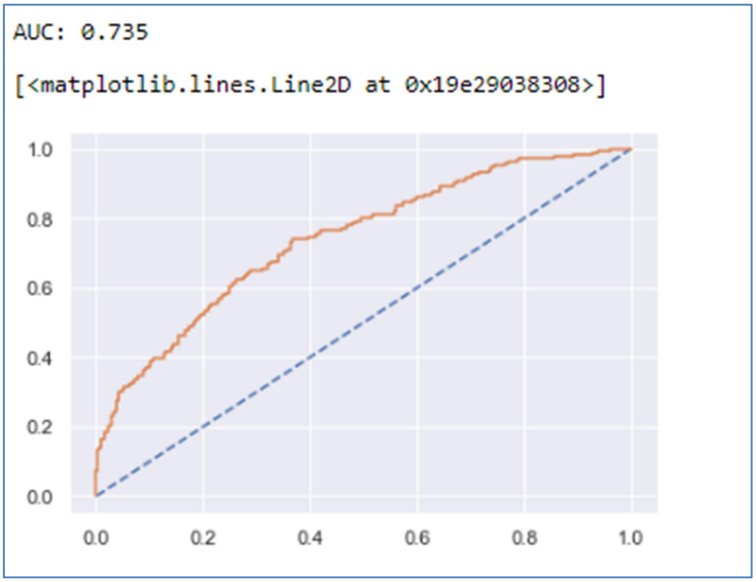
```
array([[364, 107],
       [181, 220]], dtype=int64)
```

```
print(classification_report(Y, pred_class))
```

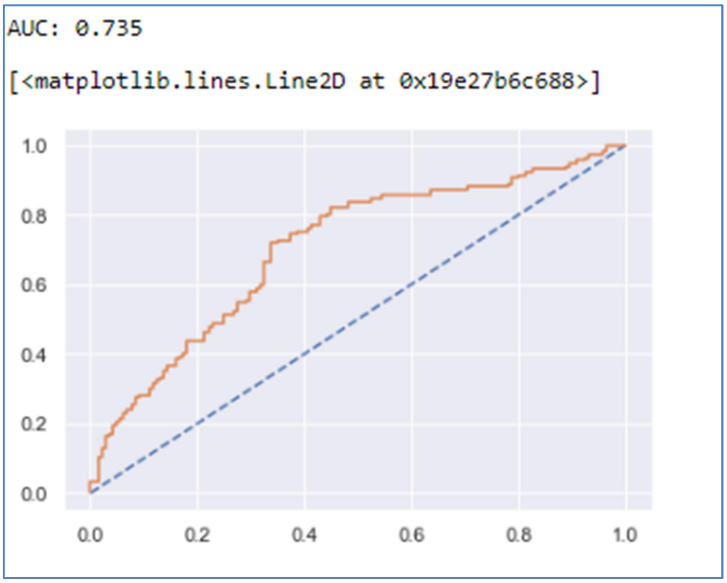
```
              precision    recall  f1-score   support

           0       0.67      0.77      0.72       471
           1       0.67      0.55      0.60       401

    accuracy                           0.67       872
   macro avg       0.67      0.66      0.66       872
weighted avg       0.67      0.67      0.66       872
```

Plot ROC Curve for Logistic Regression (Train- Test)

ROC for Training data

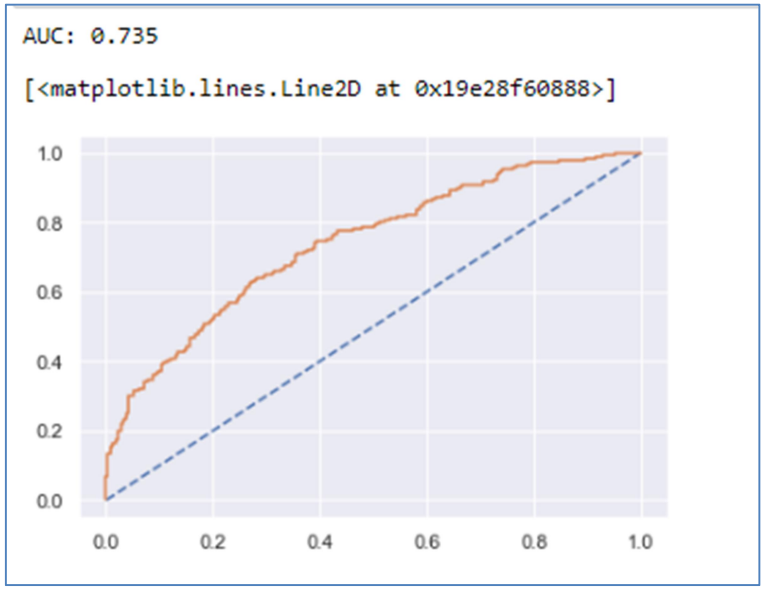


ROC for Test data

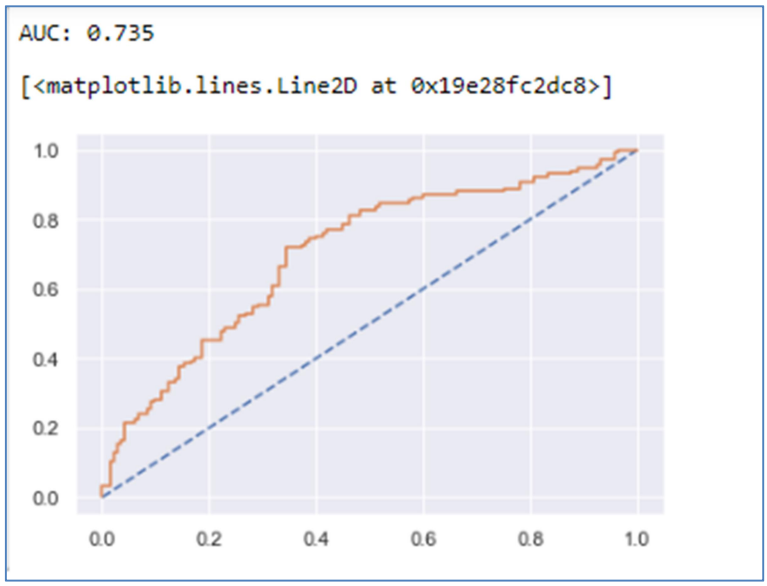


Plot ROC Curve for LDA (Train- Test data)

ROC Curve for Training Data



ROC for Test data



Comparison of Logistic Regression and LDA

	Logistic Train	Logistic Test	LDA Train	LDA Test
Accuracy	0.68	0.64	0.67	0.66
AUC	0.74	0.70	0.73	0.71
Recall	0.61	0.60	0.61	0.60
Precision	0.54	0.57	0.54	0.57
F1 Score	0.69	0.63	0.69	0.63

From the combined report of LDA and Logistic Regression, we can analyze that the Accuracy of the model is 1% better in LDA Training data. The AUC curve is also 1% better in Logistic Regression. Thus we can conclude that Logistic Regression Method is the better model compared to LDA.

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

Both the models that we have prepared are giving us similar results in terms of predictions. The predictions state that an employee who's earning was higher than 80,000 are the class where that opts for a holiday package. The company should always focus on the Salary of the employee and Age of the employee. Even the employee can look forward that employee who has young children below 3 years as their tickets and the accommodations are having exception in travelling. Hence, the company can target those employees and can look for them to opt for a package. As per our model and the data, 72% of the employees can be target to make them opt for a holiday packages.