# Problem 1

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

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

The data is ingested. Following is the glimpse of the data: -

| | 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 |

Th unnamed columns is really not required. We can remove it: -

| | 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 |

| | carat | cut | color | clarity | depth | table | x | y | z | price |
|---|---|---|---|---|---|---|---|---|---|---|
| **26962** | 1.11 | Premium | G | SI1 | 62.3 | 58.0 | 6.61 | 6.52 | 4.09 | 5408 |
| **26963** | 0.33 | Ideal | H | IF | 61.9 | 55.0 | 4.44 | 4.42 | 2.74 | 1114 |
| **26964** | 0.51 | Premium | E | VS2 | 61.7 | 58.0 | 5.12 | 5.15 | 3.17 | 1656 |
| **26965** | 0.27 | Very Good | F | VVS2 | 61.8 | 56.0 | 4.19 | 4.20 | 2.60 | 682 |
| **26966** | 1.25 | Premium | J | SI1 | 62.0 | 58.0 | 6.90 | 6.88 | 4.27 | 5166 |

**Checking data types: -**

| | Data Type |
|---|---|
| carat | float64 |
| cut | object |
| color | object |
| clarity | object |
| depth | float64 |
| table | float64 |
| x | float64 |
| y | float64 |
| z | float64 |
| price | int64 |

As can be seen, all the data types are int/float except cut, color and clarity.

A brief description about the variables: -

**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.

**Width**: The Width of the cubic zirconia's Table expressed as a Percentage of its average Diameter.

**Price**: The price of the crystal

**x, y, z**: Length, Breadth and height of the crystal.

**Shape of Data:**

```
print('Shape of data is',df.shape[0],'x',df.shape[1])
```

```
Shape of data is 26967 x 10
```

### Checking for duplicates:

```
print('Duplicates:', df.duplicated().sum())
```

```
Duplicates: 34
```

Its better to remove the duplicates.

```
df.drop_duplicates(inplace = True)
print('Duplicates:',df.duplicated().sum())
print('Shape:',df.shape[0],' x',df.shape[1])
```

```
Duplicates: 0
Shape: 26933  x 10
```

Successfully treating and removing the duplicates.

### Checking NULL values

| | Missing values |
|---|---|
| carat | 0 |
| cut | 0 |
| color | 0 |
| clarity | 0 |
| depth | 697 |
| table | 0 |
| x | 0 |
| y | 0 |
| z | 0 |
| price | 0 |

### Checking for zeros

```
df[df.x==0]
```

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

```
df[df.y==0]
```

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

```
df[df.z==0]
```

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

These values need to be treated, which will be taken care in the next question.

**Summary**

| | count | mean | std | min | 25% | 50% | 75% | max |
|---|---|---|---|---|---|---|---|---|
| **carat** | 26933.0 | 0.798010 | 0.477237 | 0.2 | 0.40 | 0.70 | 1.05 | 4.50 |
| **depth** | 26236.0 | 61.745285 | 1.412243 | 50.8 | 61.00 | 61.80 | 62.50 | 73.60 |
| **table** | 26933.0 | 57.455950 | 2.232156 | 49.0 | 56.00 | 57.00 | 59.00 | 79.00 |
| **x** | 26933.0 | 5.729346 | 1.127367 | 0.0 | 4.71 | 5.69 | 6.55 | 10.23 |
| **y** | 26933.0 | 5.733102 | 1.165037 | 0.0 | 4.71 | 5.70 | 6.54 | 58.90 |
| **z** | 26933.0 | 3.537769 | 0.719964 | 0.0 | 2.90 | 3.52 | 4.04 | 31.80 |
| **price** | 26933.0 | 3937.526120 | 4022.551862 | 326.0 | 945.00 | 2375.00 | 5356.00 | 18818.00 |

It can be clearly seen that depth column has many missing values.

Basic info: -

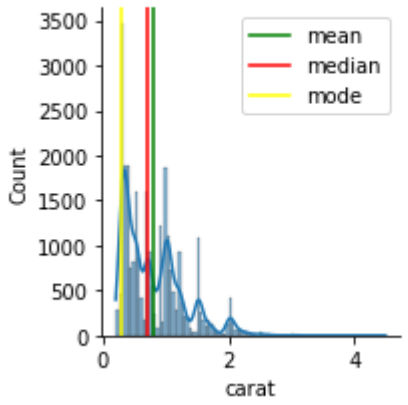
```
#   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   carat    26967 non-null  float64
 1   cut      26967 non-null  object
 2   color    26967 non-null  object
 3   clarity  26967 non-null  object
 4   depth    26270 non-null  float64
 5   table    26967 non-null  float64
 6   x        26967 non-null  float64
 7   y        26967 non-null  float64
 8   z        26967 non-null  float64
 9   price    26967 non-null  int64
```

**Univariate Analysis: -**

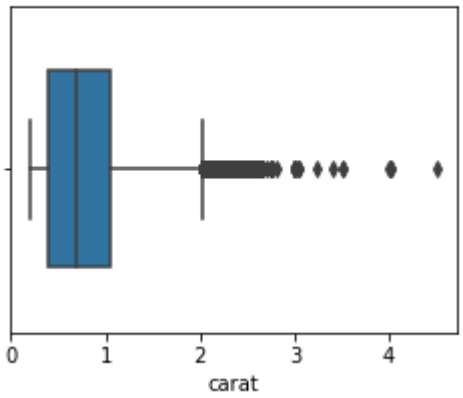
**1. Carat**

```
Description of carat  is: -
count    26967.000000
mean         0.798375
std          0.477745
min          0.200000
25%          0.400000
50%          0.700000
75%          1.050000
max          4.500000
Name: carat, dtype: float64

------------------------------
Mean is:  0.7983754218118336
Median is:  0.7
Mode is:  0.3
```

Distribution of carat is: -




Boxplot of carat is: -

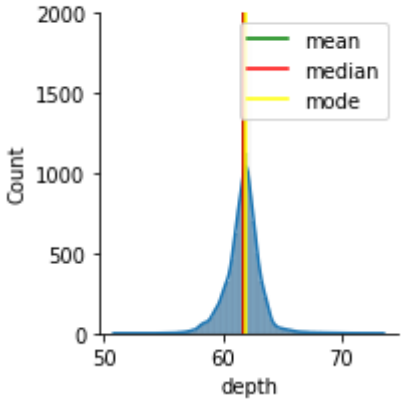



The mean and median are quite close apart with median on the lower side, representing a right skewed distribution. Less standard deviation and presence of outliers.

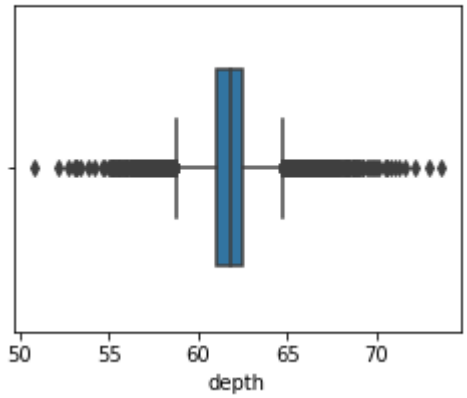
**2. Depth**

```
Description of depth  is: -
count    26270.000000
mean        61.745147
std          1.412860
min         50.800000
25%         61.000000
50%         61.800000
75%         62.500000
max         73.600000
Name: depth, dtype: float64

------------------------------
Mean is:  61.745146555006194
Median is:  61.8
Mode is:  62.0
```

Distribution of depth is: -



Boxplot of depth is: -

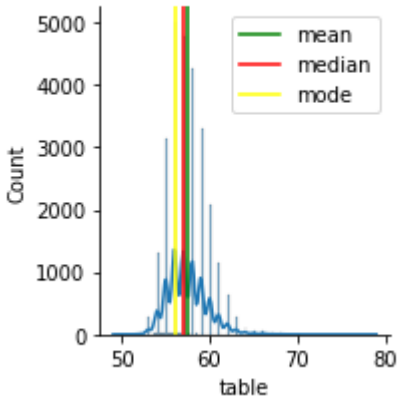



The distribution is almost tending to Normal. Mean~Median~Mode. Standard deviation is quite low. Presence of outliers on both sides of central value.

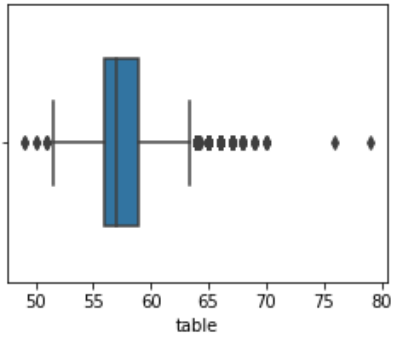
### 3. Table

```
Description of table  is: -
count    26967.000000
mean        57.456080
std          2.232068
min         49.000000
25%         56.000000
50%         57.000000
75%         59.000000
max         79.000000
Name: table, dtype: float64

------------------------------
Mean is:  57.45607965290908
Median is:  57.0
Mode is:  56.0
------------------------------
```

Distribution of table is: -




Boxplot of table is: -

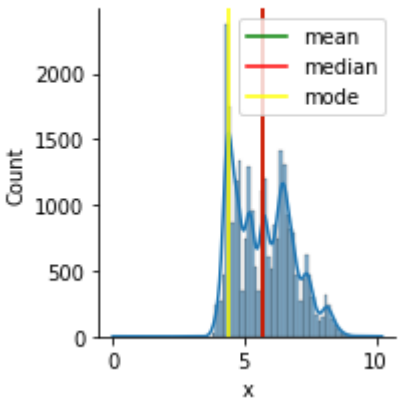


Mean~Median~Mode, indicating a normal distribution, but the dist. plot offers something else. There are several peaks in the data set, insisting several modes, however, I have shown only a single mode here. Standard deviation is very less, indicating a uniform distribution. Boxplot suggests presence of outliers.

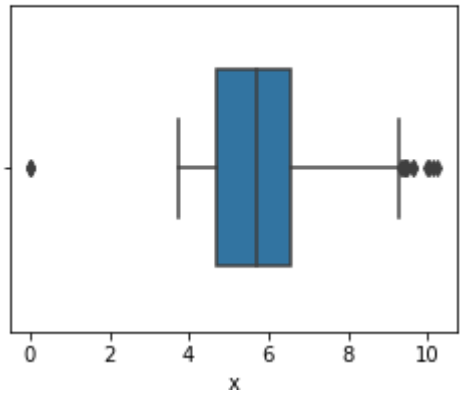
**4. X**

```
Description of x  is: -
count    26967.000000
mean         5.729854
std          1.128516
min          0.000000
25%          4.710000
50%          5.690000
75%          6.550000
max         10.230000
Name: x, dtype: float64

------------------------------
Mean is:  5.729853524678309
Median is:  5.69
Mode is:  4.38
```

Distribution of x is: -




Boxplot of x is: -

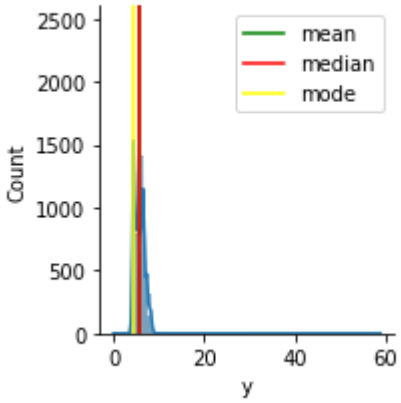



There are several peaks in the distribution, indicating presence of more than one mode. Mean and median are quite separated. Standard deviation<Mean, representing a less COV indicating a uniformity. Boxplot suggest presence of outliers.

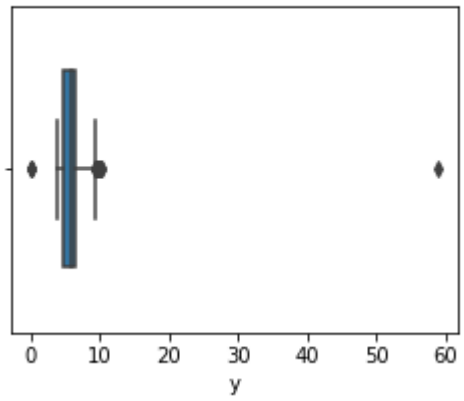
**5. Y**

```
Description of y  is: -
count    26967.000000
mean         5.733569
std          1.166058
min          0.000000
25%          4.710000
50%          5.710000
75%          6.540000
max         58.900000
Name: y, dtype: float64

------------------------------
Mean is:  5.733568806318799
Median is:  5.71
Mode is:  4.35
------------------------------
```

Distribution of y is: -




Boxplot of y is: -

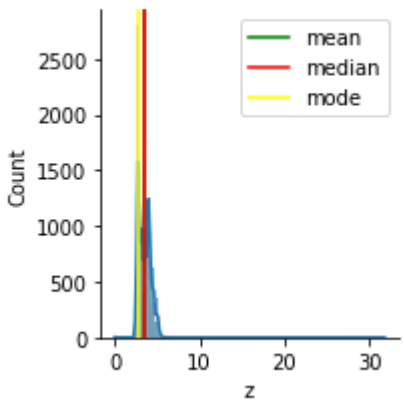



Mean>Median>Mode, indicating a right skewed distribution. Although the data is not very much spread out, we can see the right skewness. Std<Mean, indicating a bit uniformity in the data.

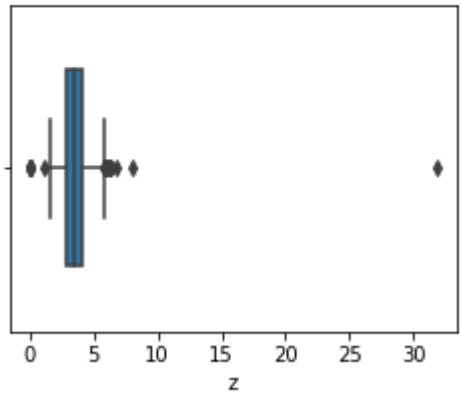
**6.Z**

```
Description of z  is: -
count    26967.000000
mean         3.538057
std          0.720624
min          0.000000
25%          2.900000
50%          3.520000
75%          4.040000
max         31.800000
Name: z, dtype: float64

------------------------------
Mean is:  3.5380572551637184
Median is:  3.52
Mode is:  2.69
------------------------------
```

Distribution of z is: -



Boxplot of z  is: -




Mean = Median, but mode is less than median. The distribution might be a bit right skewed. There are some outliers present in the data.

**Bivariate analysis**

Looking at the unique values present in the object variables.

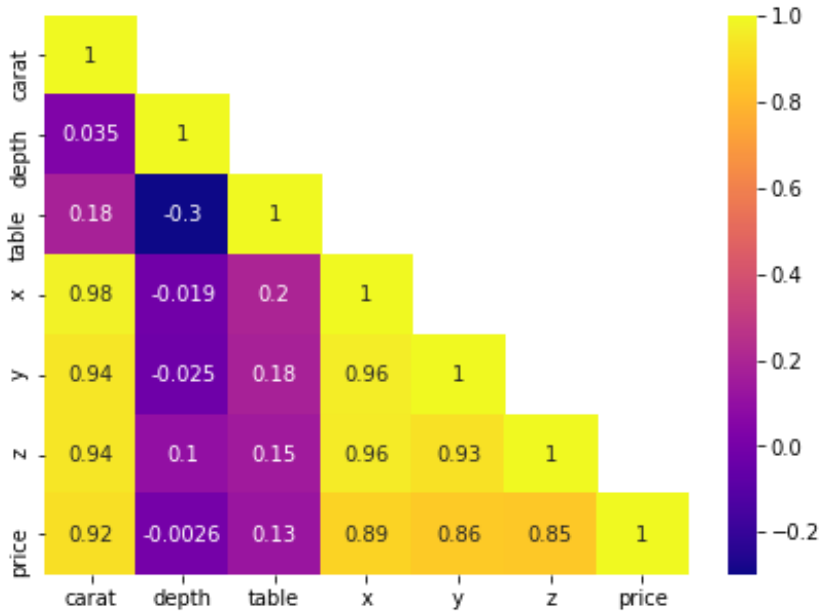
```
Unique values in column: CUT
Ideal        10816
Premium       6899
Very Good     6030
Good          2441
Fair           781
```

```
Unique values in column: COLOR
G    5661
E    4917
F    4729
H    4102
D    3344
I    2771
J    1443
```

```
Unique values in column: CLARITY
SI1     6571
VS2     6099
SI2     4575
VS1     4093
VVS2    2531
VVS1    1839
IF       894
I1       365
```

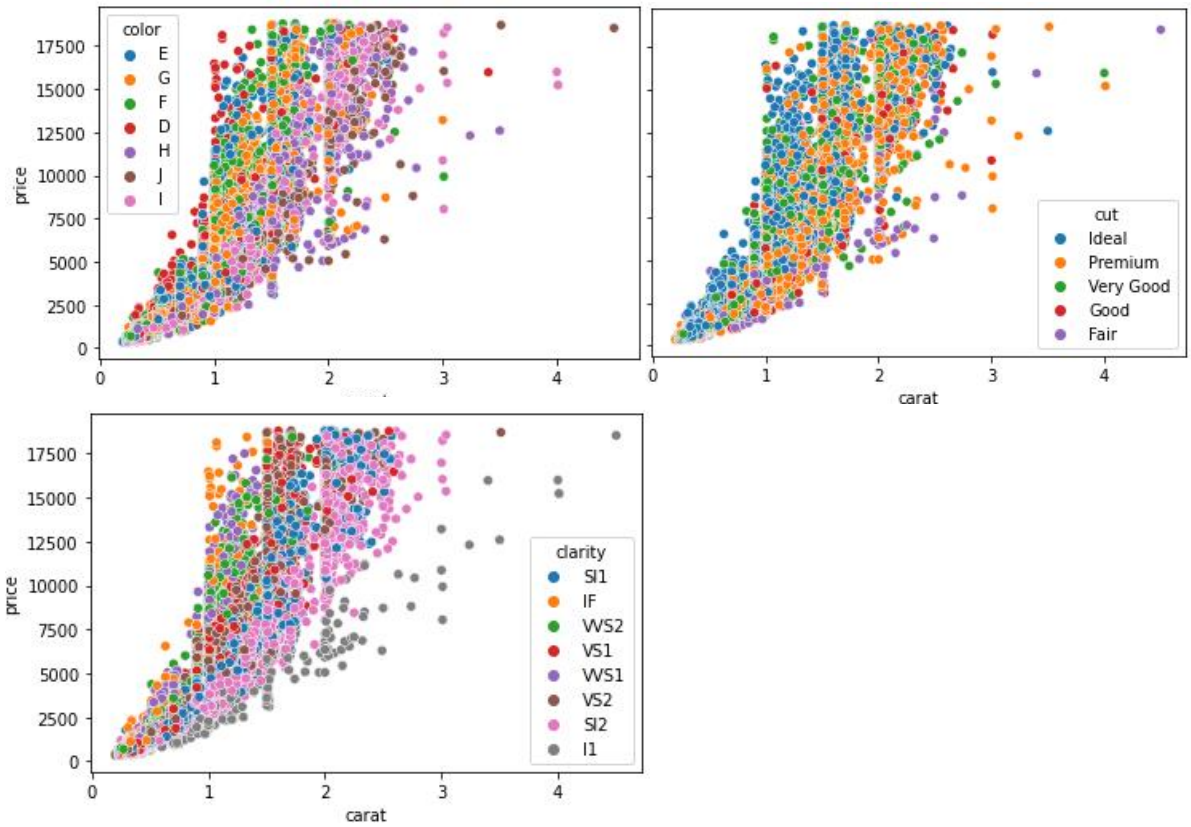
The ordinal level of each of the unique values is already being described in the beginning. Since the unique values are not same, we can't rely on bar plots for bivariate analysis, as we would be getting an average value, which is not wise to have when we have so many outliers in the data. Boxplots are better way to visualize.

Also, since the data columns are too many in, no, picturizing the pair plot in the report won't make much sense. A heatmap, can be shown, which would also give the same idea.

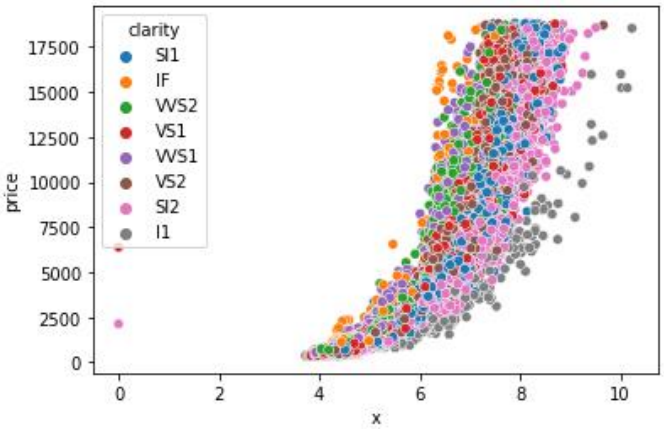


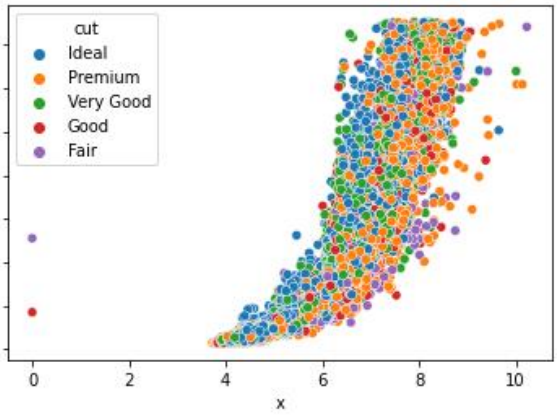

The heatmap clearly shows, there is heavy correlation between the ones shown in yellow. There is also presence of multi-collinearity. Price is very highly corelated with x, y, z and carat. Also, x, y, z is highly corelated amongst each other. Carat is also highly co-related with x, y and z. This multicollinearity is going to affect the linear model very badly, but at least, we know which ones are the good predictors.

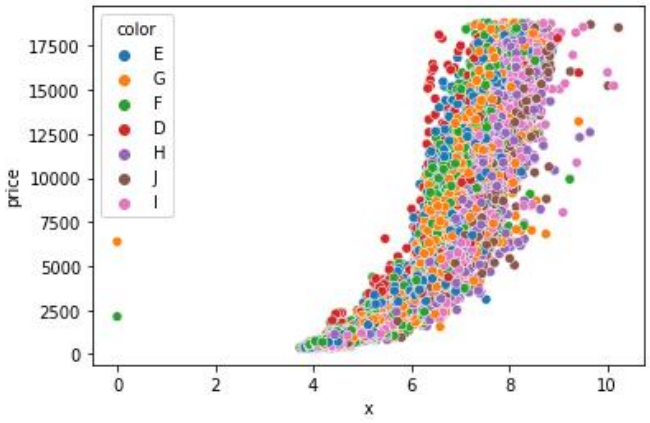
Following are a few scatterplots representing the correlation: -

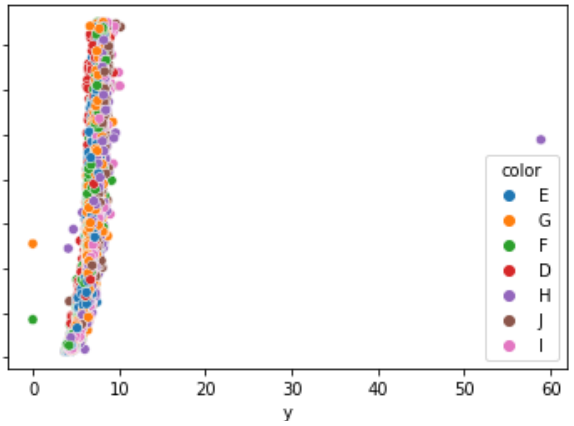


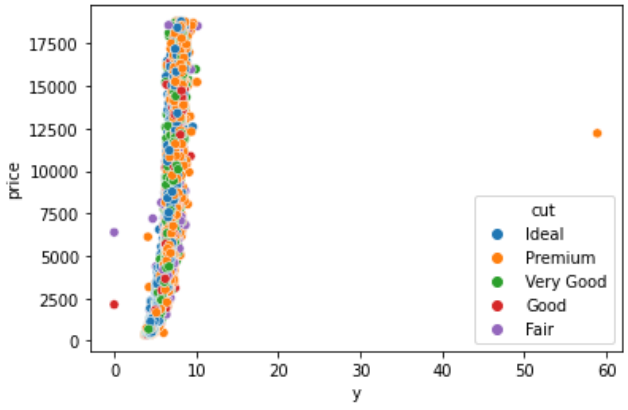

For carat, we can clearly see that there is an increase in the price, and that too is steepest for color D, quality ideal, clarity IF, which are ranked best for quality.

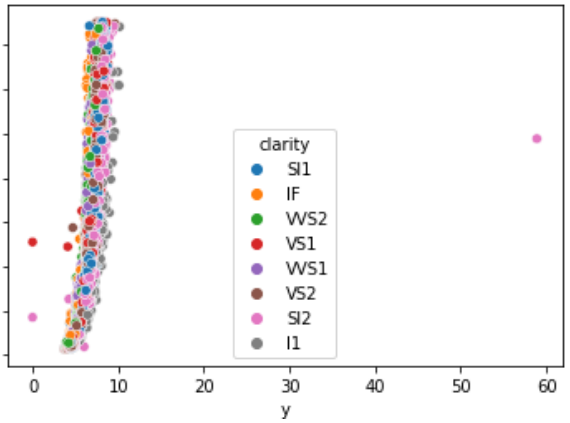

clarity
SI1
IF
VVS2
VS1
VVS1
VS2
SI2
I1
price
x

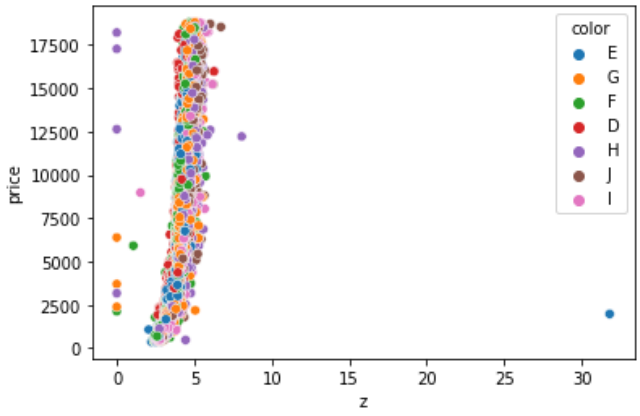

cut
Ideal
Premium
Very Good
Good
Fair
x

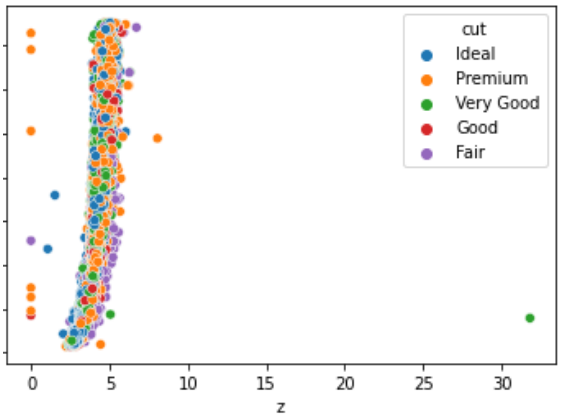

color
E
G
F
D
H
J
I
price
x

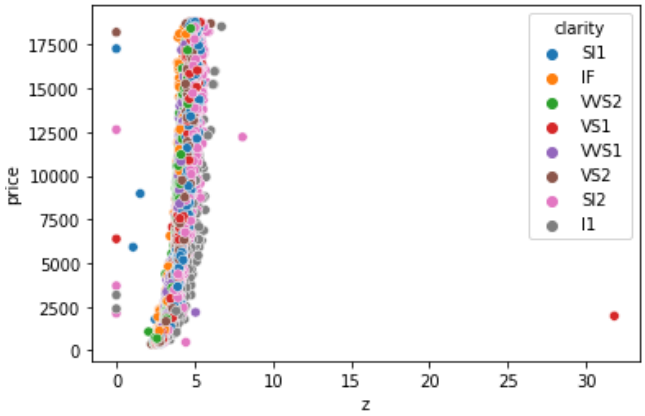

color
E
G
F
D
H
J
I
y


cut
Ideal
Premium
Very Good
Good
Fair
price
y


clarity
SI1
IF
VVS2
VS1
VVS1
VS2
SI2
I1
y


color
E
G
F
D
H
J
I
price
z


cut
Ideal
Premium
Very Good
Good
Fair
z

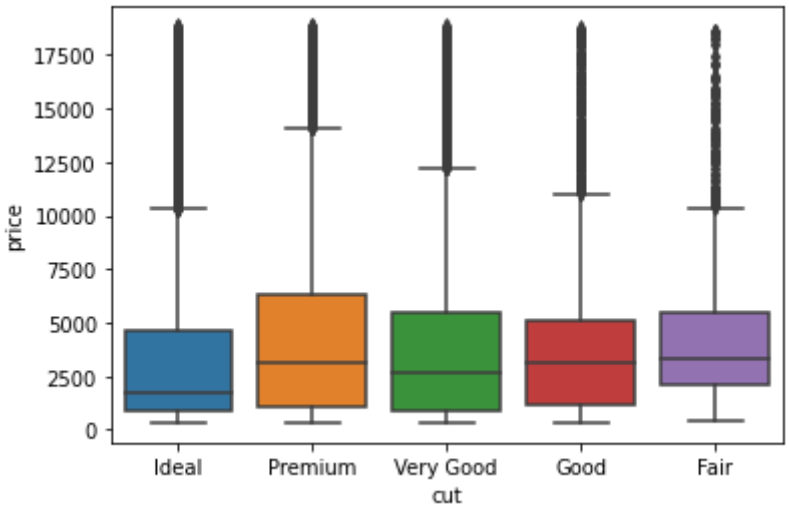



From the plots above we can infer, there is steep increase in price with the dimensions. The slope is steepest for the ones which have the best quality.

The results of the scatterplot are synonymous with what we discussed in the heatmap.

Let's have a look at the effect of categorical variables on price.

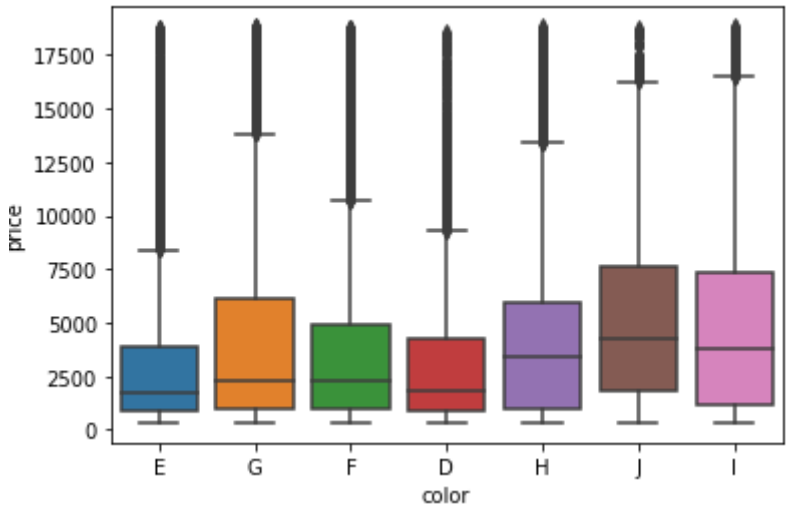
**Cut: -**




The increasing quality of cut goes like-> Fair-> Good-> Very Good-> Premium->Ideal

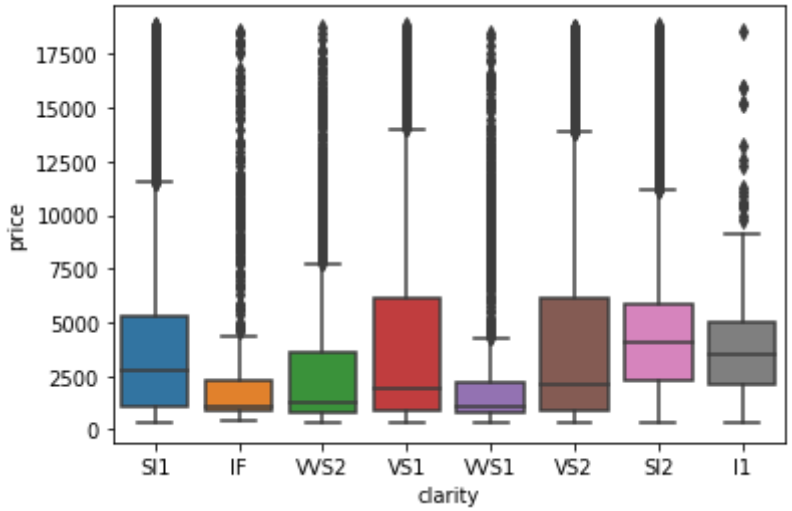
There are indeed outliers present in the data. However, if we ignore them, the max price goes for premium cut. Although, ideal cut is the best in quality, it does not yield equivalent pricing.

**Color: -**




As per the given data, the order of color from best to worst is: D->J (Best to Worst). However, the quality does not seem to reflect on price. If we ignore the outliers, the best color quality D has substantially low price, while on the other hand, I and J being the worst, have high prices.

**Clarity**




There are too many outliers here. If we look at quality of cut: FL, IF, VVS1, VVS2, VS1, VS2, SI1, SI2, I1, I2, I3 in decreasing order. In that case VVS1 and VVS2, the ones with best quality have comparatively low-price range. Nonetheless, VS1 and VS2 have justified prices for their quality.

## 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 already have checked the null values and the zeroes present. Displaying them again: -

| | Missing values |
|---|---|
| carat | 0 |
| cut | 0 |
| color | 0 |
| clarity | 0 |
| depth | 697 |
| table | 0 |
| x | 0 |
| y | 0 |
| z | 0 |
| price | 0 |

```
df[df.x==0]
```

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

```
df[df.y==0]
```

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

```
df[df.z==0]
```

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

We can clearly see the missing values and the zeroes. Null values are present in the depth column, whereas zeroes in x, y and z. X, Y and Z are the dimensions of the crystal which can't be zero. The Null values can be directly imputed, but the zeroes need to be converted to np.nan then imputed.

```
df.x = df.x.replace(0,np.nan)
df.y = df.y.replace(0,np.nan)
df.z = df.z.replace(0,np.nan)
```

The zeroes are replaced with np.nan. The new null values look like this: -

```
carat        0
cut          0
color        0
clarity      0
depth      697
table        0
x            2
y            2
z            8
price        0
```

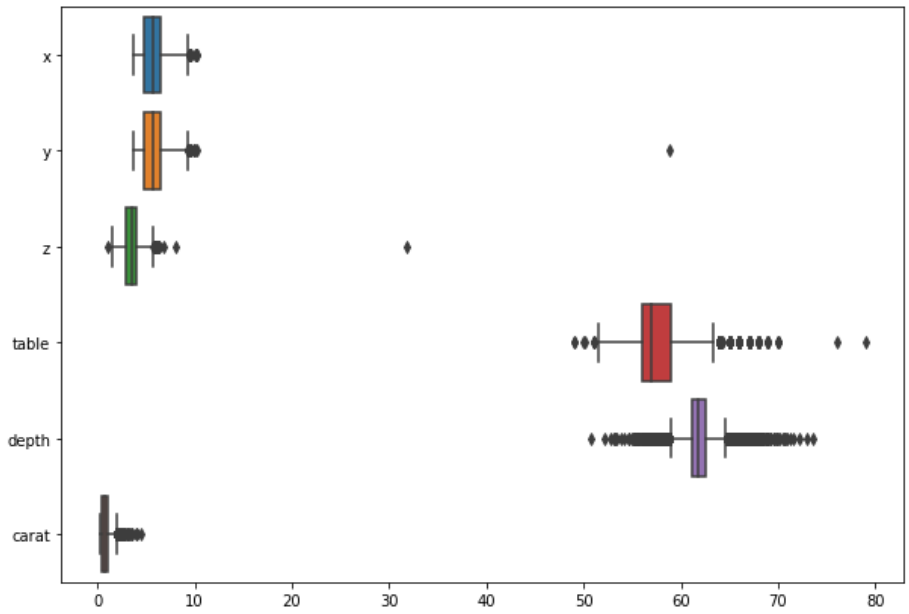
As can be seen, now even x, y and z have null values. These can be directly imputed with the median.

The null value table after imputing them: -

| | NULL Values |
|---|---|
| carat | 0 |
| cut | 0 |
| color | 0 |
| clarity | 0 |
| depth | 0 |
| table | 0 |
| x | 0 |
| y | 0 |
| z | 0 |
| price | 0 |

The data is complete now, with no null values and zeroes.

As far as scaling is considered we can have a look at the boxplot first: -



There are outliers present, and the scale is not the same, but not too vast as well. Scaling the variables would restrict them to +1 and -1. For all the variables negative values won't make any sense and hence, scaling the variables would lead to loss of interpretability.

Hence, I strongly feel, there is absolutely no need to scale the variables. Also, treating the outliers would make us loose quite a major chunk of the data. So, no need to treat them as well.

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

To begin with, we know cut, color and clarity are the variables. These variables can't be directly label encoded. These are ordinal in nature. We need to explicitly encode them.

The ordinal values are given as: variable with highest quality, awarded the highest no, and vice-versa. For e.g., the color variable has values from D to J, D being the best and J the worst. D is assigned 7, while J as 1. The other variables are encoded accordingly. The data frame looks like this after encoding.

| carat | cut | color | clarity | depth | table | x | y | z | price |
|---|---|---|---|---|---|---|---|---|---|
| 0.30 | 5 | 6 | 3 | 62.1 | 58.0 | 4.27 | 4.29 | 2.66 | 499 |
| 0.33 | 4 | 4 | 8 | 60.8 | 58.0 | 4.42 | 4.46 | 2.70 | 984 |
| 0.90 | 3 | 6 | 6 | 62.2 | 60.0 | 6.04 | 6.12 | 3.78 | 6289 |
| 0.42 | 5 | 5 | 5 | 61.6 | 56.0 | 4.82 | 4.80 | 2.96 | 1082 |
| 0.31 | 5 | 5 | 7 | 60.4 | 59.0 | 4.35 | 4.43 | 2.65 | 779 |

| | Data Type |
|---|---|
| carat | float64 |
| cut | int64 |
| color | int64 |
| clarity | int64 |
| depth | float64 |
| table | float64 |
| x | float64 |
| y | float64 |
| z | float64 |
| price | int64 |

The next step is to separate the target value from the data set and split into train and test. (The steps can be seen in detail in the python file)

The model is created and coefficients are found: -

```
Coefficient of  carat  is:  11131.97172130182
Coefficient of  cut  is:  107.28987644346343
Coefficient of  color  is:  334.09629862475083
Coefficient of  clarity  is:  504.5027691971787
Coefficient of  depth  is:  -82.0351502755951
Coefficient of  table  is:  -30.081791696879765
Coefficient of  x  is:  -984.3267054058525
Coefficient of  y  is:  10.011040432151123
Coefficient of  z  is:  -45.305242609685095

The intercept is: 3660.4626049032972
```

The coefficients seem to be a bit deceptive. In the scatterplots we had seen that x, y and z were highly corelated with price, but the model does not say so. This is due to multicollinearity which existed in the data set, and we can't do much to handle it. Although we have option to use PCA, but at the cost of interpretability.

Observing the error and accuracy: -

```
Model score for train data: 0.9087174183809242
Model score for test data: 0.9080038175630045


MSE for train data is: 1467435.9518321683
MSE for test data is: 1510966.0280178315


RMSE Train: 1211.3777081621438
RMSE test: 1229.2135811232447
```

The model shows accuracy of 90% on both train and test data, which is almost perfect.

The ambiguity in the variables can be checked by creating the model using stats models. Following are the results: -

OLS Regression Results

| | | | |
|---|---|---|---|
| **Dep. Variable:** | price | **R-squared:** | 0.909 |
| **Model:** | OLS | **Adj. R-squared:** | 0.909 |
| **Method:** | Least Squares | **F-statistic:** | 2.084e+04 |
| **Date:** | Mon, 08 Feb 2021 | **Prob (F-statistic):** | 0.00 |
| **Time:** | 12:07:42 | **Log-Likelihood:** | -1.6060e+05 |
| **No. Observations:** | 18853 | **AIC:** | 3.212e+05 |
| **Df Residuals:** | 18843 | **BIC:** | 3.213e+05 |
| **Df Model:** | 9 | | |

| | coef | std err | t | P>\|t\| | [0.025 | 0.975] |
|---|---|---|---|---|---|---|
| Intercept | 3660.4626 | 723.109 | 5.062 | 0.000 | 2243.103 | 5077.822 |
| carat | 1.113e+04 | 93.346 | 119.255 | 0.000 | 1.09e+04 | 1.13e+04 |
| cut | 107.2899 | 9.739 | 11.017 | 0.000 | 88.201 | 126.378 |
| color | 334.0963 | 5.480 | 60.962 | 0.000 | 323.354 | 344.838 |
| clarity | 504.5028 | 5.916 | 85.283 | 0.000 | 492.908 | 516.098 |
| depth | -82.0352 | 7.890 | -10.397 | 0.000 | -97.500 | -66.570 |
| table | -30.0818 | 4.984 | -6.036 | 0.000 | -39.851 | -20.313 |
| x | -984.3267 | 50.690 | -19.419 | 0.000 | -1083.684 | -884.970 |
| y | 10.0110 | 23.763 | 0.421 | 0.674 | -36.567 | 56.589 |
| z | -45.3052 | 41.512 | -1.091 | 0.275 | -126.672 | 36.062 |

| | | | |
|---|---|---|---|
| Omnibus: | 4008.482 | Durbin-Watson: | 1.981 |
| Prob(Omnibus): | 0.000 | Jarque-Bera (JB): | 166200.863 |
| Skew: | 0.083 | Prob(JB): | 0.00 |
| Kurtosis: | 17.545 | Cond. No. | 6.99e+03 |

We can clearly see, the P value for y and z is >05, and thus the Null hypothesis can't be rejected. In the linear model, y and z are redundant and can't include in the model. A new model without y and z looks like the following.

OLS Regression Results

| | | | |
|---|---|---|---|
| Dep. Variable: | price | R-squared: | 0.909 |
| Model: | OLS | Adj. R-squared: | 0.909 |
| Method: | Least Squares | F-statistic: | 2.680e+04 |
| Date: | Mon, 08 Feb 2021 | Prob (F-statistic): | 0.00 |
| Time: | 12:07:51 | Log-Likelihood: | -1.6060e+05 |
| No. Observations: | 18853 | AIC: | 3.212e+05 |
| Df Residuals: | 18845 | BIC: | 3.213e+05 |
| Df Model: | 7 | | |
| Covariance Type: | nonrobust | | |

| | coef | std err | t | P>\|t\| | [0.025 | 0.975] |
|---|---|---|---|---|---|---|
| Intercept | 3819.1281 | 709.321 | 5.384 | 0.000 | 2428.795 | 5209.461 |
| carat | 1.113e+04 | 93.298 | 119.316 | 0.000 | 1.09e+04 | 1.13e+04 |
| cut | 107.4674 | 9.733 | 11.042 | 0.000 | 88.390 | 126.544 |
| color | 334.0463 | 5.480 | 60.957 | 0.000 | 323.305 | 344.788 |
| clarity | 504.4472 | 5.915 | 85.281 | 0.000 | 492.853 | 516.041 |
| depth | -84.7044 | 7.522 | -11.260 | 0.000 | -99.449 | -69.960 |
| table | -29.9917 | 4.978 | -6.024 | 0.000 | -39.750 | -20.234 |
| x | -1002.1567 | 39.625 | -25.291 | 0.000 | -1079.825 | -924.488 |

| | | | |
|---|---|---|---|
| Omnibus: | 4009.014 | Durbin-Watson: | 1.981 |
| Prob(Omnibus): | 0.000 | Jarque-Bera (JB): | 166315.113 |
| Skew: | 0.083 | Prob(JB): | 0.00 |
| Kurtosis: | 17.550 | Cond. No. | 6.83e+03 |

The p values are<05, and we can completely reject the Null hypothesis. X has a negative coefficient, which goes against the observations of scatterplot. This is indeed due to multicollinearity. We can have a look at the VIF to ensure multicollinearity: -

```
VIF for carat is:  82.35348832171735
VIF for cut is:  15.008355763026957
VIF for color is:  8.52527887553793
VIF for clarity is:  8.452756345131997
VIF for depth is:  565.7131948750647
VIF for table is:  545.902158911356
VIF for x is:  1135.2003852329688
VIF for y is:  348.0183352189224
VIF for z is:  383.2874142963292
```

As we can see, there is a high VIF for every variable, which confirms the presence of multicollinearity.

The similar procedure has been carried out for scaled variables, and it has yielded similar results (accuracy on train and test), however, its not apt for interpretation.

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

The linear model which we got definitely gave us interpretable results, but we do not really know, what is the impact of individual entities in the object variables i.e., cut, colour and clarity on price. To give intact insights, we need to create model for individual entities too. For this we need to create dummies for each of them, and then find our final linear model.

| | Coefficients |
|---|---|
| Intercept | 4115.764425 |
| carat | 11407.260850 |
| depth | -65.887948 |
| table | -26.271618 |
| x | -1098.051334 |
| cut_Fair | 313.183739 |
| cut_Good | 798.918434 |
| cut_Ideal | 1083.447850 |

| | Coefficients |
|---|---|
| cut_Premium | 977.295550 |
| cut_verygood | 942.918852 |
| color_D | 1504.801538 |
| color_E | 1305.056009 |
| color_F | 1221.561682 |
| color_G | 1036.614533 |
| color_H | 536.208892 |
| color_J | -884.273819 |

| | Coefficients |
|---|---|
| clarity_I1 | -3412.384260 |
| clarity_IF | 1979.816403 |
| clarity_SI1 | 391.945803 |
| clarity_SI2 | -559.170482 |
| clarity_VS1 | 1291.059120 |
| clarity_VS2 | 1012.061639 |
| clarity_VVS1 | 1735.403293 |
| clarity_VVS2 | 1677.032908 |

If we might notice the coefficients obtained are not exactly the same. This occurs due to factors such as multicollinearity and the different mechanisms used for convergence. Previously the data set had fewer no of columns, but the data set with dummies have suddenly increased no of columns. There is a deviation, but not up to an extent which would degrade the interpretability of the model.

As asked in the problem to list the most important features here are, they: -

| | index | Coefficients |
|---|---|---|
| 1 | carat | 11407.260850 |
| 0 | Intercept | 4115.764425 |
| 17 | clarity_IF | 1979.816403 |
| 22 | clarity_VVS1 | 1735.403293 |
| 23 | clarity_VVS2 | 1677.032908 |
| 10 | color_D | 1504.801538 |

From the above coefficients, it is very easy to distinguish between the profitable and non-profitable diamonds. The company must target at the parameters with positive coefficients, and try to minimize the ones with negative coefficients.

For e.g., it should try to sell diamonds with max carat, clarity index IF, VVS1, VVS2 and color D, and minimize diamonds which have large table, depth and x. Although, these results might not match with the scatter plots we have seen earlier, but the model is 90% accurate, and hence we can rely on it.

# Problem 2

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.

Ingesting the data, and looking at its head: -

| | 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 |

**Checking data Types**

```
 #   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
```

There are 7 columns, two out of which are object, while rest are integers.

Holiday Package: Opted for Holiday Package yes/no?

Salary:  Employee salary

Age: Age in years

Edu: Years of formal education

No young children/No of older children: The number of young children (younger than 7 years)/no of children>7 yrs.

Foreign: foreigner yes/no.

**Checking Null values and duplicates**

| | Missing Values |
|---|---|
| Holliday_Package | 0 |
| Salary | 0 |
| age | 0 |
| educ | 0 |
| no_young_children | 0 |
| no_older_children | 0 |
| foreign | 0 |

```
print('Duplicates in the data:', df.duplicated().sum())
```

```
Duplicates in the data: 0
```

No missing values, nor any duplicates.

```
print('Shape of data is:',df.shape[0],' x ',df.shape[1])
```

```
Shape of data is: 872  x  7
```

The data contains 872 rows and 7 columns.

**Summary of the data: -**

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

Salary columns seem to have some outliers, and also it is a bit right skewed. The two children columns are not continuous, they are discreet, and can be converted into categorical at further stages in the question.

**Checking unique values: -**

```
df['Holliday_Package'].value_counts()
```

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

```
df['foreign'].value_counts()
```

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

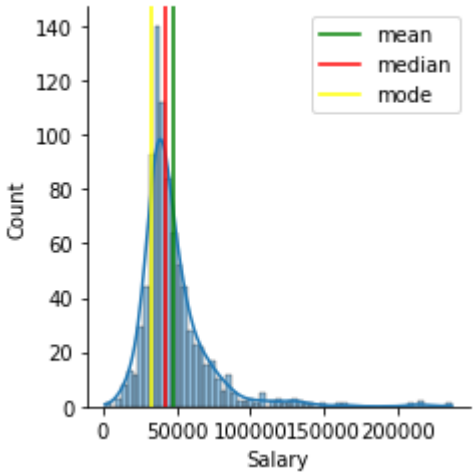
Holiday Package is the target variable, which already seems to be balanced in nature.

**Univariate analysis**

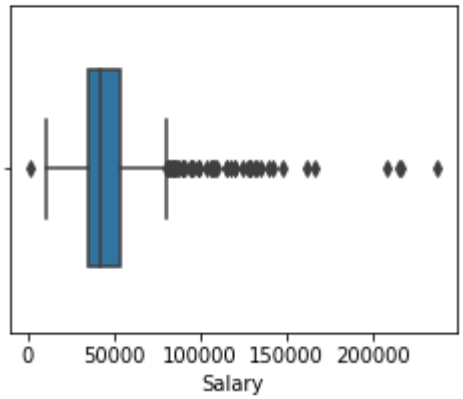
**1. Salary: -**

```
Description of Salary  is: -
count       872.000000
mean      47729.172018
std       23418.668531
min        1322.000000
25%       35324.000000
50%       41903.500000
75%       53469.500000
max      236961.000000
Name: Salary, dtype: float64

------------------------------
Mean is:  47729.172018348625
Median is:  41903.5
Mode is:  32197
------------------------------
```

Distribution of Salary is: -




Boxplot of Salary is: -

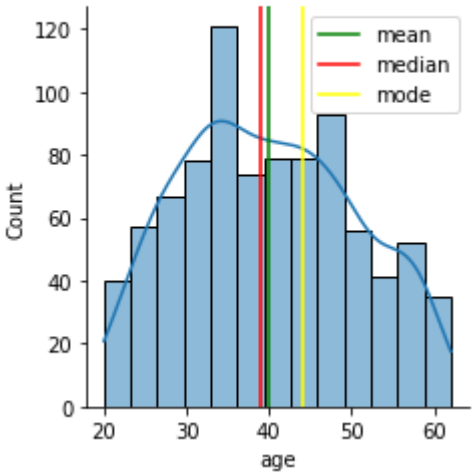



The salary variable is a bot right skewed, mean>median>mode. There are too many outliers in the data. The std deviation is also too much.

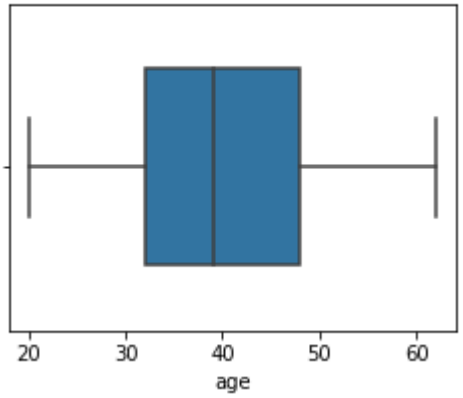
## 2. Age

```
Description of age  is: -
count    872.000000
mean      39.955275
std       10.551675
min       20.000000
25%       32.000000
50%       39.000000
75%       48.000000
max       62.000000
Name: age, dtype: float64

------------------------------
Mean is:  39.955275229357795
Median is:  39.0
Mode is:  44
------------------------------
```

Distribution of age is: -




Boxplot of age is: -

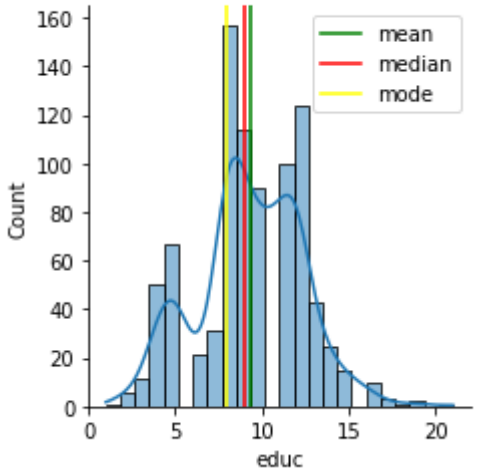



The age variable seems to be distributed almost normally, with multiple modes. Mean and median are almost equal, std deviation<mean resulting in a low COV. No outliers present

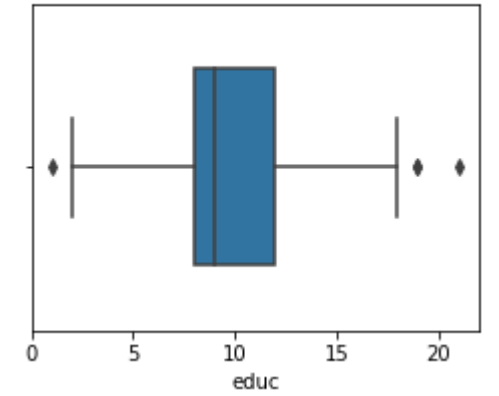
## 3. Educ

```
Description of educ  is: -
count    872.000000
mean       9.307339
std        3.036259
min        1.000000
25%        8.000000
50%        9.000000
75%       12.000000
max       21.000000
Name: educ, dtype: float64

----------------------------
Mean is:  9.307339449541285
Median is:  9.0
Mode is:  8
```

Distribution of educ is: -



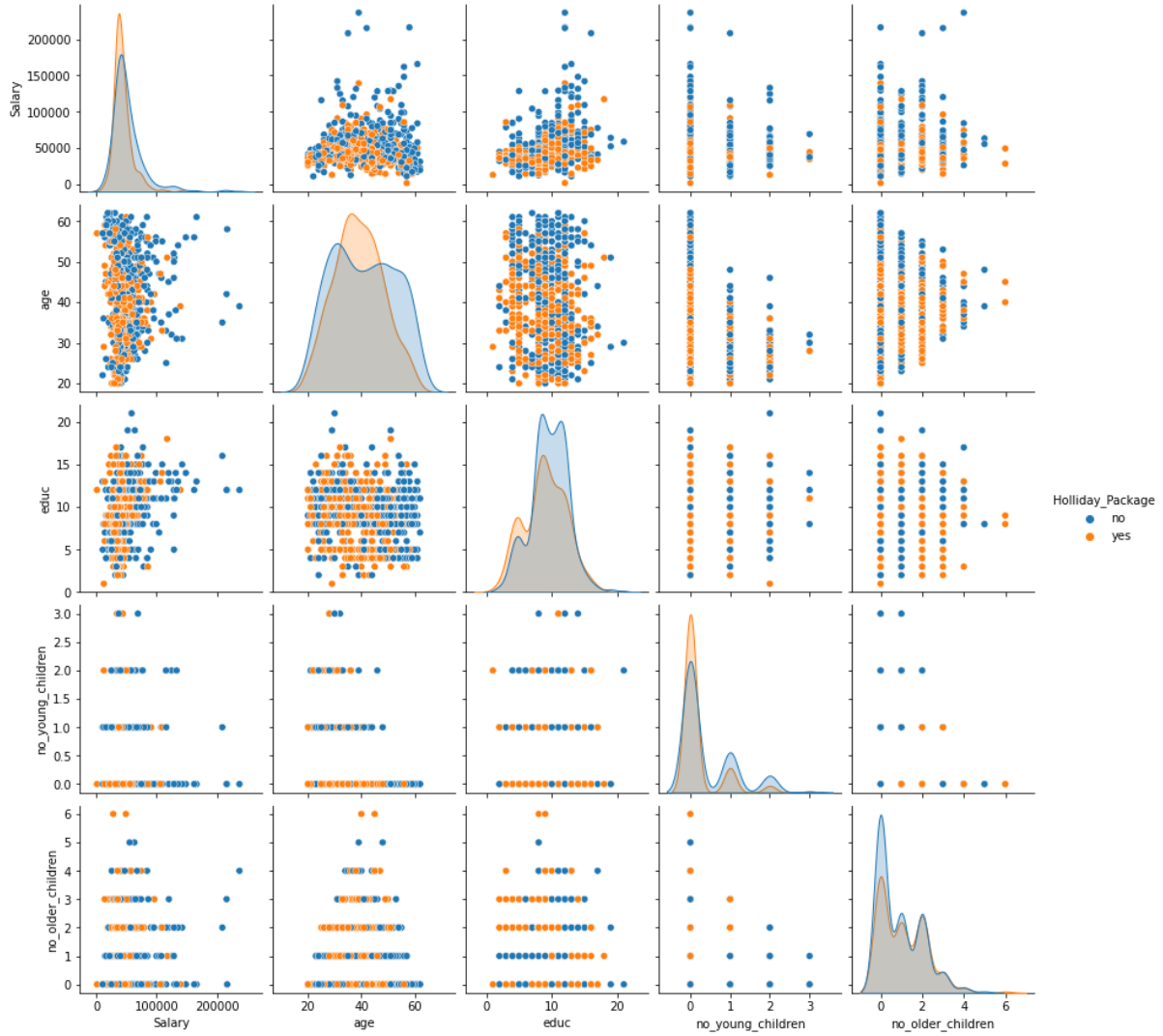
Boxplot of educ is: -




The education column represents discrete values. The mean and median are quite similar, with mode at around 8. There are some outliers present in the column.

**Bivariate analysis**

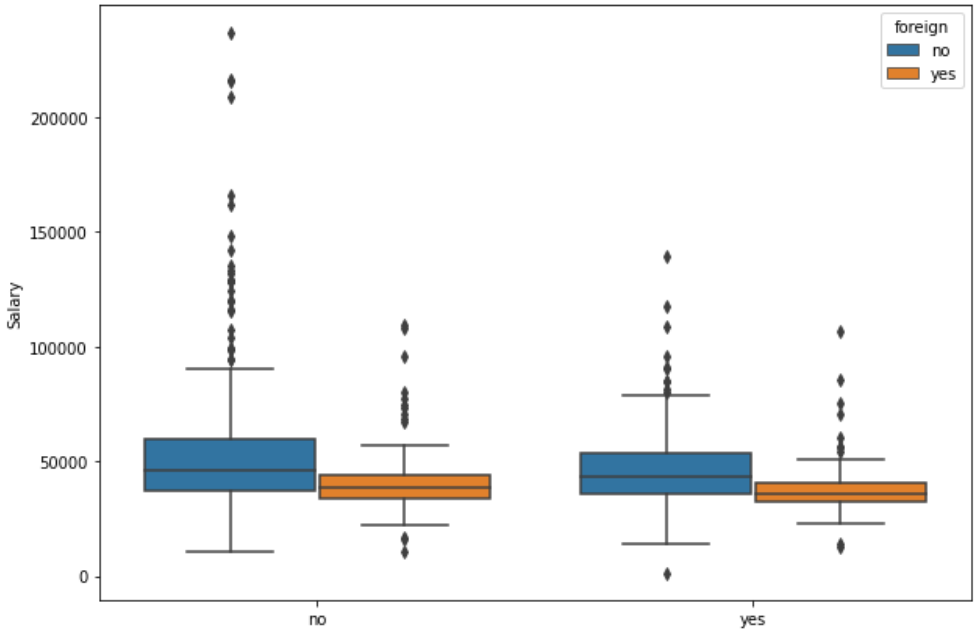
Following is the pair plot: -



The pair plot shows that there is almost none co-relation that exists between the variables. Also, from the diagonal we can see, there is almost complete overlapping for both the classes yes and no There is not a single variable, which can be used to clearly distinguish them.

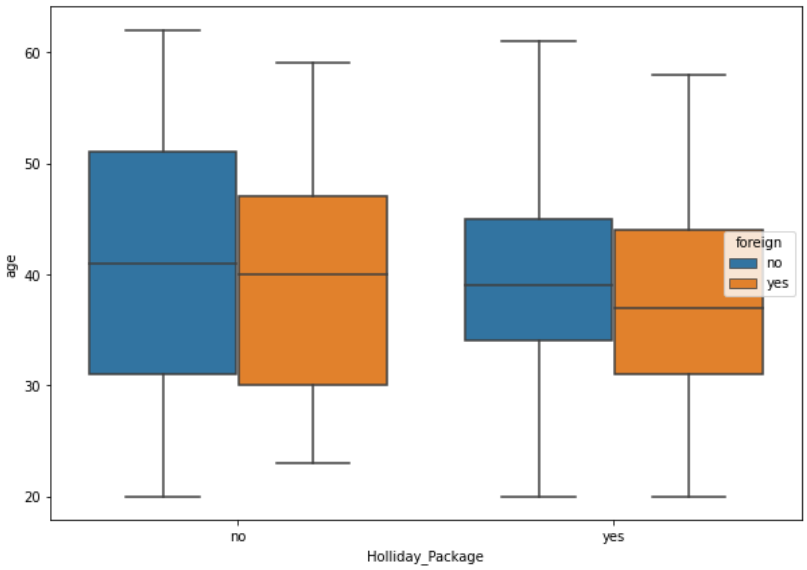
Let’s have a look at the bivariate boxplots: -

**Salary vs Holiday Package**



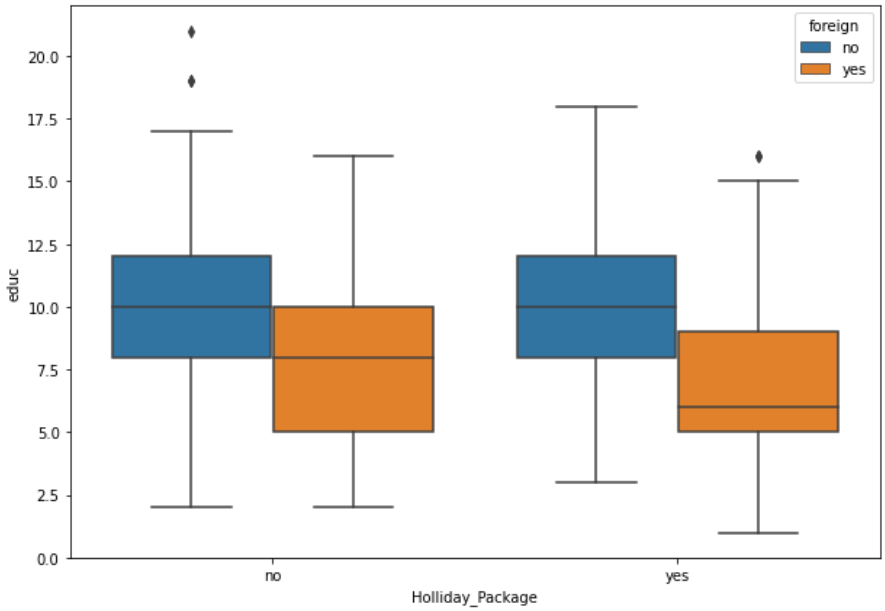

If we look at salary, those choosing the holiday package are the ones with low salary packages. Foreigners opting for the package have even less salary than those not opting for the package.

**Age vs holiday package**



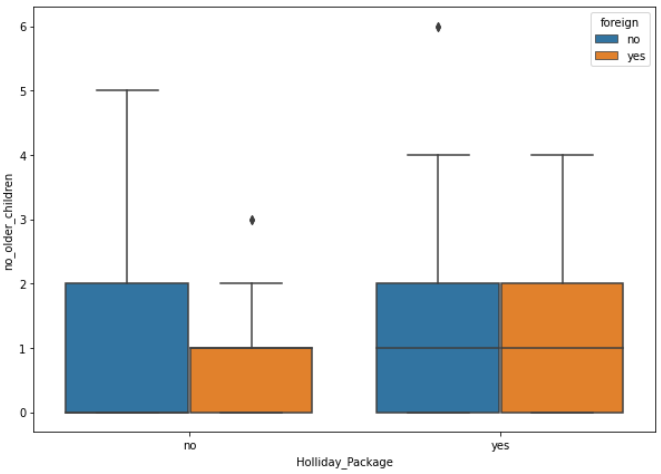

Young tourists are more likely to opt for the holiday package, than their counterparts.

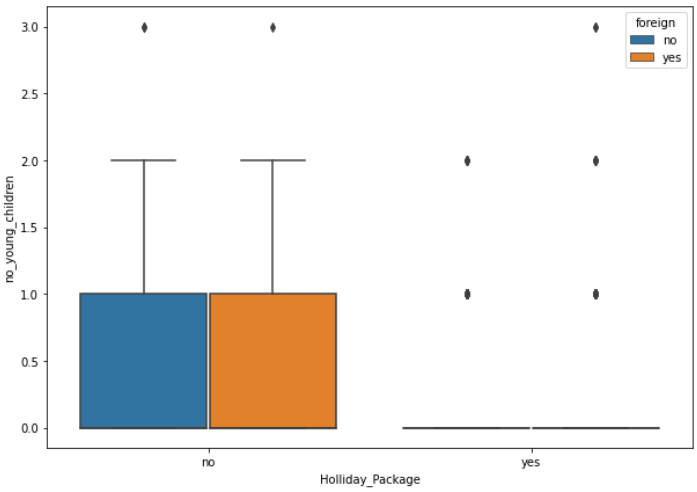
**Educ vs Holiday Package**




Natives opting for holiday package have comparatively more no of formal education years. While foreigners opting for the package have comparatively less no of formal education years.

**Old and Young children vs Holiday package: -**




As far as young children are considered, tourists that are having young children are not at all opting for the holiday package. Also, natives with more no of older children are not opting for the holiday package. On the other hand, foreigners, are more willing to go for the holiday package even with more no of children.

## 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)

Holiday Package is the target variable, lets see how is the distribution of yes's and nos.

```
df.Holliday_Package.value_counts(normalize = True)
```

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

The distribution is 54%: No, and 45.9%: Yes. The distribution is almost symmetrical.

The, variable, no of young children and no o fold children, is completely discreet in nature and can be converted into categorical.

Also, the columns holiday package and foreign need to be converted. The data after conversion looks: -

| | Data Type |
|---|---|
| Holliday_Package | int64 |
| Salary | float64 |
| age | int64 |
| educ | int64 |
| no_young_children | category |
| no_older_children | category |
| foreign | int64 |

**Logistic Regression: -**

The data is now ready to fit into the model.

To start with, the data is divided into the target variable and the rest. Further, it is split into train and test variables.

The train data is fitted into the logistic regression model to check the accuracy on train data and test data.

```
print('Accuracy on train data :',logit.score(x_train,train_labels))
```

```
Accuracy on train data : 0.519672131147541
```

```
print('Accuracy on test data:',logit.score(x_test,test_labels))
```

```
Accuracy on test data: 0.5305343511450382
```

Although, the model seems to be a good fit, the accuracy is too less. We need to use the grid search cv to refine the parameters further.

```
grid={'penalty':['l1','elasticnet','l2','none'],
      'solver':['sag','lbfgs','saga','liblinear','newton-cg'],
      'tol':[0.1,.001,.0001,.000001],
      'dual':[True,False],
      'C':[.25,.5,.75,1],
      'fit_intercept':[True,False],
      'class_weight':['dict','balanced'],
      }
```

The above hyper-parameters are used to find the best fit model. Following is the best estimator found after running the model.

```
print(grid_search.best_estimator_)
```

```
LogisticRegression(C=0.25, class_weight='balanced', max_iter=10000, n_jobs=2,
                   solver='newton-cg', tol=0.001)
```

```
best_model.score(x_train, train_labels)
```

```
0.680327868852459
```

```
best_model.score(x_test, test_labels)
```

```
0.6679389312977099
```

The accuracy has tremendously increased, and the performance has not reduced on the test data as well.

**Performance Metrics: -**

**1. Classification report**

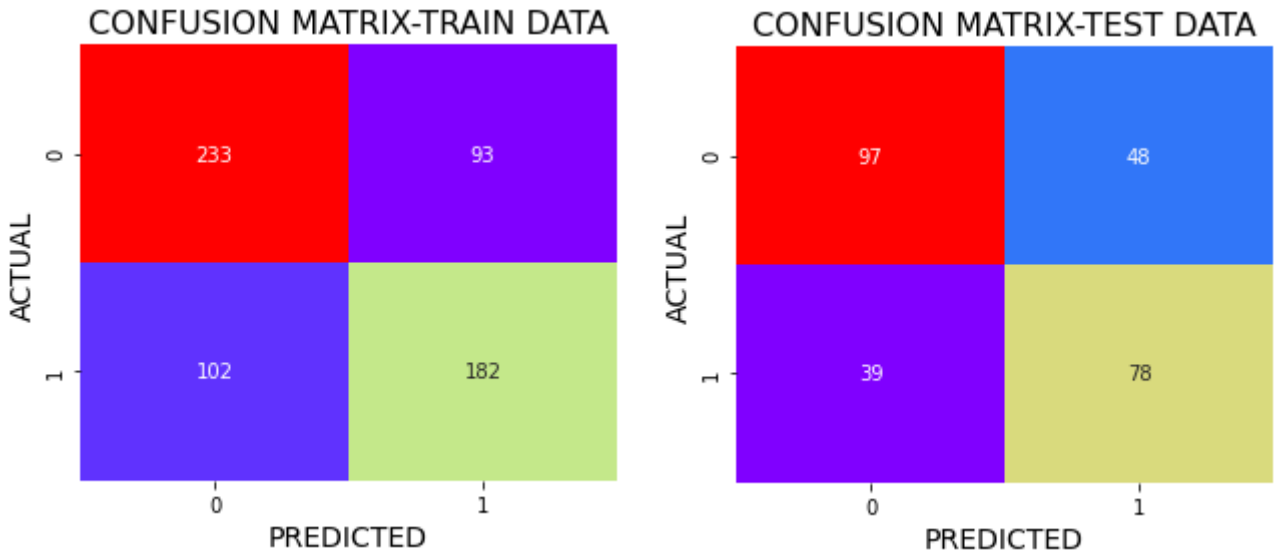
```
Classification report for train data: -
              precision    recall  f1-score   support

           0       0.70      0.71      0.70       326
           1       0.66      0.64      0.65       284

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

Classification report for test data: -

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.71 | 0.67 | 0.69 | 145 |
| 1 | 0.62 | 0.67 | 0.64 | 117 |
| accuracy | | | 0.67 | 262 |
| macro avg | 0.67 | 0.67 | 0.67 | 262 |
| weighted avg | 0.67 | 0.67 | 0.67 | 262 |

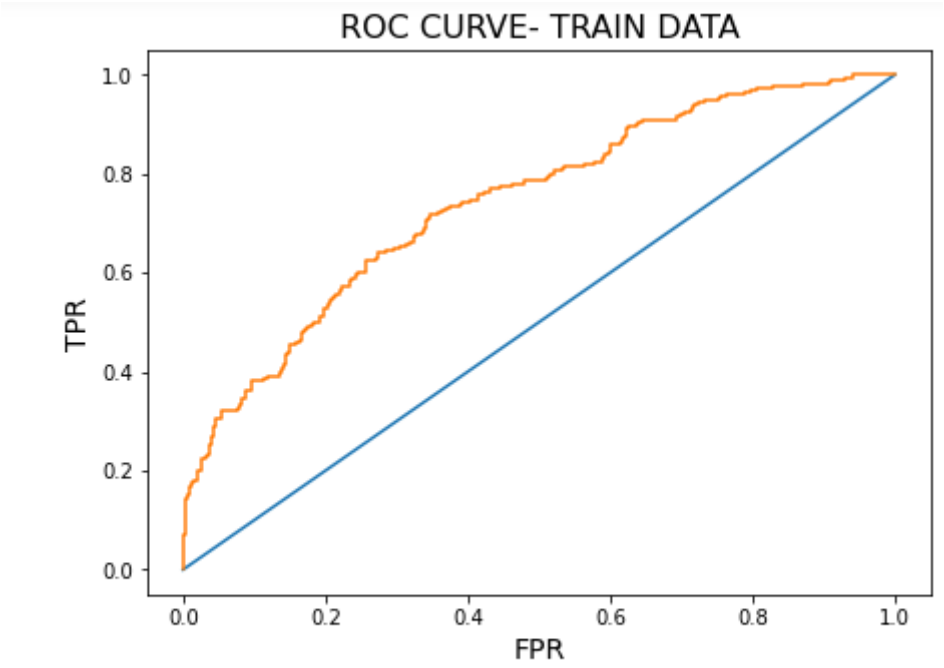
There is a good balance between recall and precision for both train and test data. There is no compromise in the accuracy either. The model is therefore a good fit.

**2. Confusion Matrix**

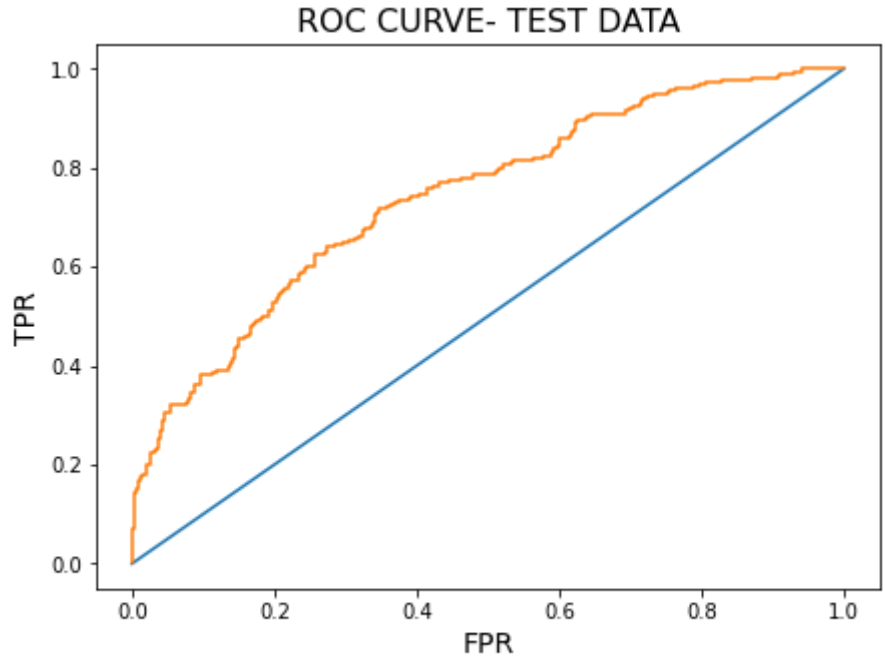



We have more true negative and true positives, compared to the false negative and positives, which have resulted in a good balcne of precision and recall, as seen in the classification report.

### 3. ROC Curve and AUC Score




```
ROC AUC Score: 0.7416076211872462
```




```
ROC AUC Score: 0.705334512231064
```

The AUC score has dropped a bit in the test data., but by a minor deviation, which again confirms model is not overfitting.

**Linear Discrimant Analysis: -**

The split data is fed into the LDA algorithm and model score is checked.

```
lda.score(x_train,train_labels)
```

```
0.6721311475409836
```

```
lda.score(x_test, test_labels)
```

```
0.6412213740458015
```

The accuracy is good enough, but let's check if tuning the parameters has any effect over the accuracy. Using the grid search cv for the same: -

```
grid_lda = {'solver':['svd','lsqr','eigen'],
            'shrinkage':['auto','float',None],
            'store_covariance':[True,False],
            'tol':[.1,.01,.001,.0001,.00001],
            }
```

```
best_model_lda.score(x_train, train_labels)
```

```
0.6721311475409836
```

```
best_model_lda.score(x_test, test_labels)
```

```
0.6412213740458015
```

Even after tuning, there is no significant effect over the model accuracy. We can move ahead with the performance metrics: -

**1. Classification Report**

```
Classification report on train data: -

              precision    recall  f1-score   support

           0       0.67      0.77      0.72       326
           1       0.68      0.56      0.61       284

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

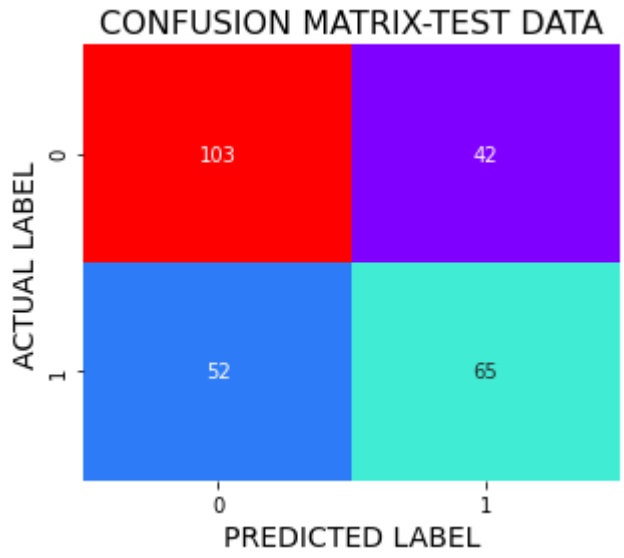
The metrics on training data are good enough. There is a balance between precision and recall, but for '1', the recall is quite poor.

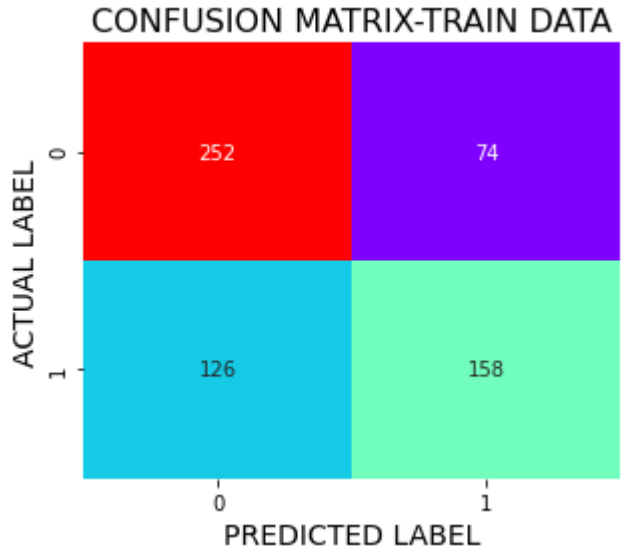
Classification report on test data: -

| | 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 |

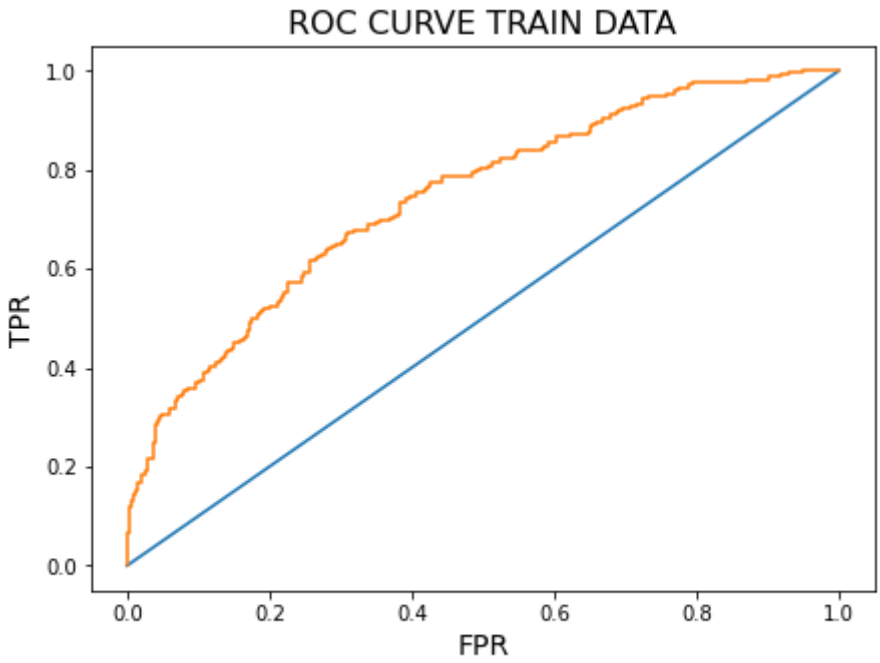
The precision seems to be good, for both the classes, but the recall is poor for the class 1.

**2. Confusion Matrix**

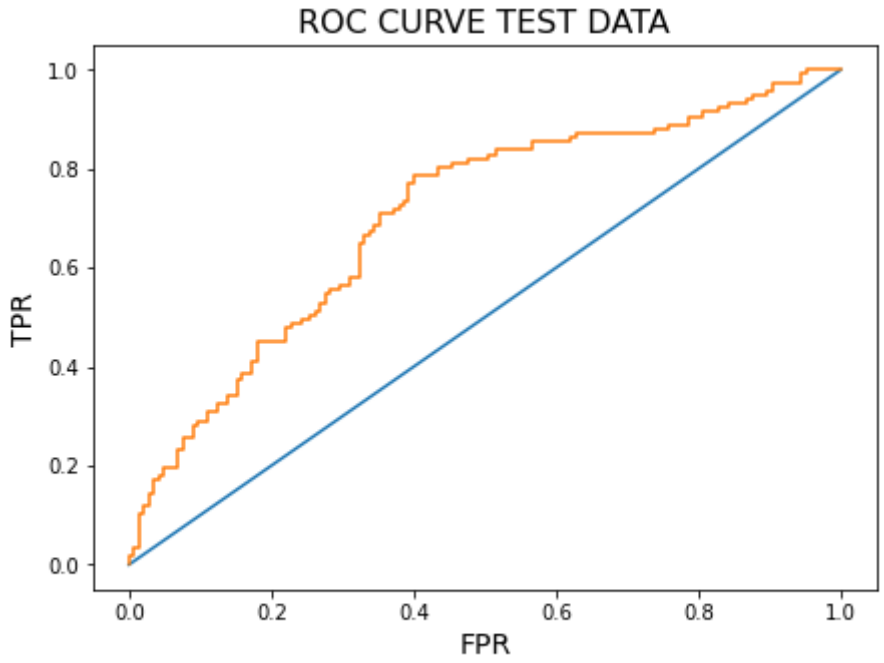


### 3. ROC Curve and AUC Score




```
ROC AUC SCORE 0.7421152682968979
```




```
ROC AUC SCORE 0.7029177718832891
```

The ROC AUC score is .74 and .70 for train and test data resepctively, which is similar to what we got for the logistic model.

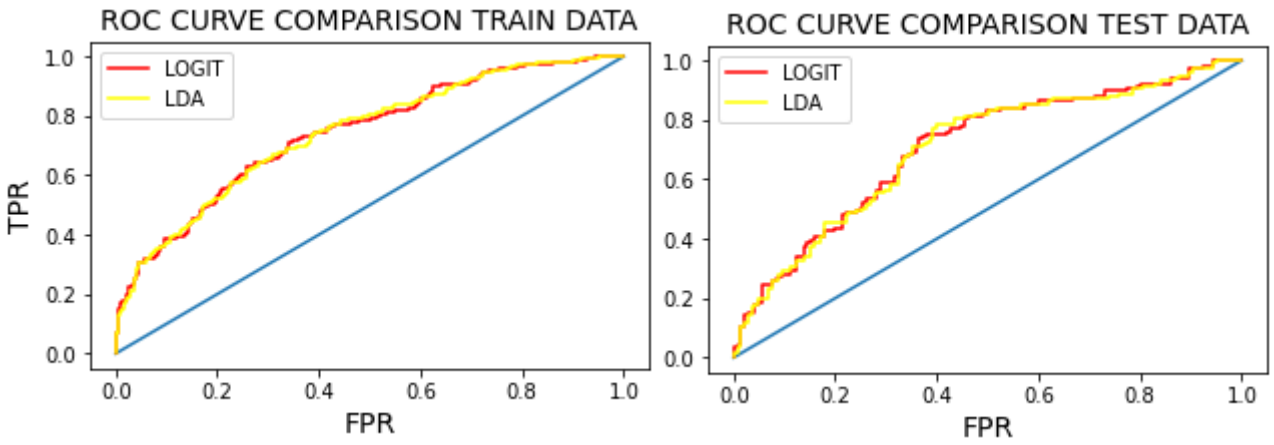
**Model comparison: -**

| | LOGIT TRAIN | LOGIT TEST | LDA TRAIN | LDA TEST |
|---|---|---|---|---|
| ACCURACY | 0.680328 | 0.667939 | 0.672131 | 0.641221 |
| PRECISION | 0.661818 | 0.619048 | 0.681034 | 0.607477 |
| RECALL | 0.640845 | 0.666667 | 0.556338 | 0.607477 |
| F1 SCORE | 0.651163 | 0.641975 | 0.612403 | 0.580357 |
| AUC | 0.741608 | 0.705335 | 0.742115 | 0.702918 |

The above table compares the logistic and the LDA models metrics on train and test data rspectively.

On comparing the two, we find that, the Logistic model is more accurate, with a better f1 score, and a better balance between recall and precision. Clearly Logistic Regression has outsmarted the LDA. Moreover, LDA would have worked fine, if there were certain variables, which were clearl distinguishers of the classes, but as we saw in the beginning itself, there were none such variables. As a result, Logistic regression did not go unstable, and give better results.

Finally, we can also compare the roc curves and the auc score. There is almost no difference between them: -



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

So far, the data set was ingested and EDA was performed. The data was clean and ready to feed to the model. Some minor changes were made, such as changing the no of younger and older children to categorical variables.

The data was split into train and test, and the training set was fed to the model. Model initially performed poor, but after tuning the parameters a decent performance was obtained both on train and test data.

The model could predict both the 0's and 1's with a decent accuracy. To increase the sales, we must identify the tourists, who are not opting for the holiday package. Identifying the 0's is the key here. More than that the company must dive deeper into the reasons for not opting for the holiday package.

The data has been already explored. Just a tabular representation: -

| Holliday_Package | Salary | age | educ |
|---|---|---|---|
| no | 43940 | 40 | 10 |
| yes | 39809 | 39 | 9 |

The above cross-tab represents, the median values of the listed columns. We can clearly see, those who have opted for the package have less median age, salary and education. Speculation won't help here, as it is quite weird, even people with high median salary have not opted for the holiday package. These are the ones who need to be focussed with promotional strategies so that they are also opting for the holiday package.

For e.g., the company can provide some discount to the highest tax payers, which would automatically involve the ones with the high salaries. Or else it can give some offers to those who have a master's degree, that would encourage the ones with higher formal education

Let's now look at the foreigners who have opted for the holiday package: -

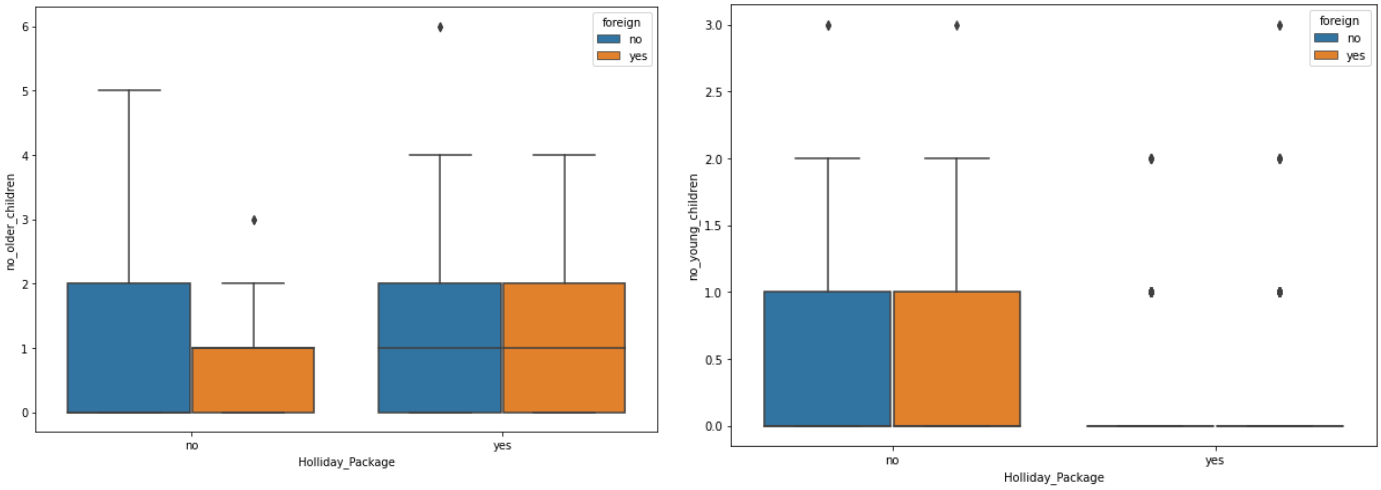
```
data.foreign.value_counts()
```

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

There are total 216 foreigners in the data set.

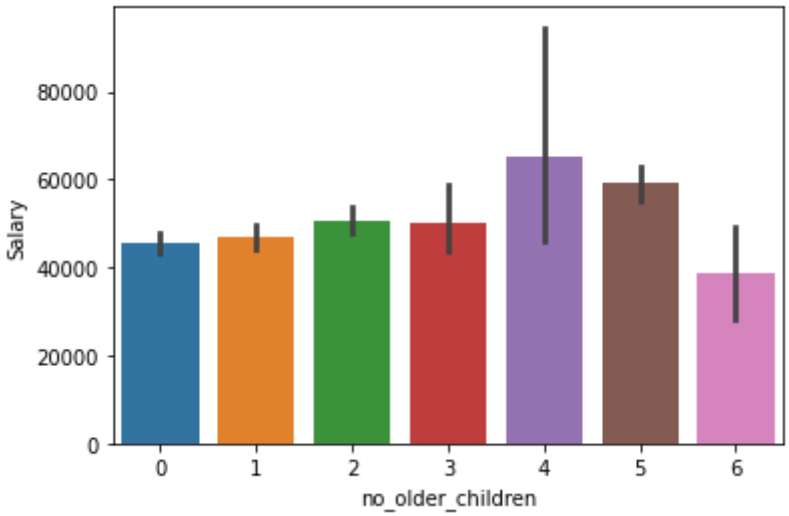
| Holliday_Package | foreign | no | yes |
|---|---|---|---|
| no | | 402 | 69 |
| yes | | 254 | 147 |

Out of 216 foreigners, 147 have opted for the holiday package, that means, almost 68% foreigners have opted for the trip. On the other hand, only 38% of the natives have opted for the package. This indicates, foreigners are more enthusiastic for travel when compared to the natives.




As far as no of children are considered, those with more no of young children do not opt for the package. It's a pretty obvious observation, as it is not at all advisable to travel with young children, which even can be infants.

Further those with more no of older children, have not opted for the package. The additional costs incurred might be one of the reasons.




If we look at the salary trend against no of children, those with 4 and 5 children have highest mean salaries. In this case, the previous assumption vanishes into thin air.

To summarise, more research is required to find the reason for those not opting for the holiday package. The logistic regression model could be used to identify the ones not opting for the package, and try to figure out the reasons for the same.