# #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.

## Removing unwanted column Unnamed: 0

## Reading the top 5 Records

Out[4]:

| | 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 |

```
<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
```

## Check the data.describe()

Out[6]:

| | carat | depth | table | x | y | z | price |
|---|---|---|---|---|---|---|---|
| count | 26967.000000 | 26270.000000 | 26967.000000 | 26967.000000 | 26967.000000 | 26967.000000 | 26967.000000 |
| mean | 0.798375 | 61.745147 | 57.456080 | 5.729854 | 5.733569 | 3.538057 | 3939.518115 |
| std | 0.477745 | 1.412860 | 2.232068 | 1.128516 | 1.166058 | 0.720624 | 4024.864666 |
| min | 0.200000 | 50.800000 | 49.000000 | 0.000000 | 0.000000 | 0.000000 | 326.000000 |
| 25% | 0.400000 | 61.000000 | 56.000000 | 4.710000 | 4.710000 | 2.900000 | 945.000000 |
| 50% | 0.700000 | 61.800000 | 57.000000 | 5.690000 | 5.710000 | 3.520000 | 2375.000000 |
| 75% | 1.050000 | 62.500000 | 59.000000 | 6.550000 | 6.540000 | 4.040000 | 5360.000000 |
| max | 4.500000 | 73.600000 | 79.000000 | 10.230000 | 58.900000 | 31.800000 | 18818.000000 |

## Check the columns

Out[7]:
```
Index(['carat', 'cut', 'color', 'clarity', 'depth', 'table', 'x', 'y', 'z',
       'price'],
      dtype='object')
```

```
<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
```

## Checking the shape of data?

Out[9]:
```
(26967, 10)
```

## Count the datatypes?

Out[10]:
```
float64    6
object     3
int64      1
dtype: int64
```

## Check the data set information

```
<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
```

## Checking the dataset missing values?

Out[12]:

| | Total | Percent |
|---|---|---|
| depth | 697 | 0.025846 |
| price | 0 | 0.000000 |
| z | 0 | 0.000000 |
| y | 0 | 0.000000 |
| x | 0 | 0.000000 |
| table | 0 | 0.000000 |
| clarity | 0 | 0.000000 |
| color | 0 | 0.000000 |
| cut | 0 | 0.000000 |
| carat | 0 | 0.000000 |

## Checking the Duplicates Values

Out[13]: 34

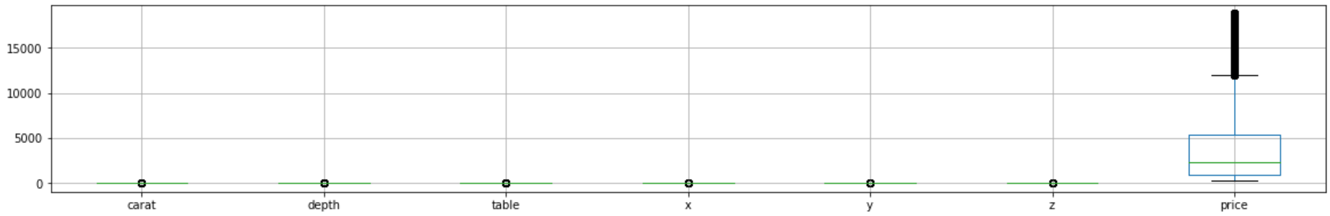
## Removing the duplicate Values

## Cross checking the duplicate values

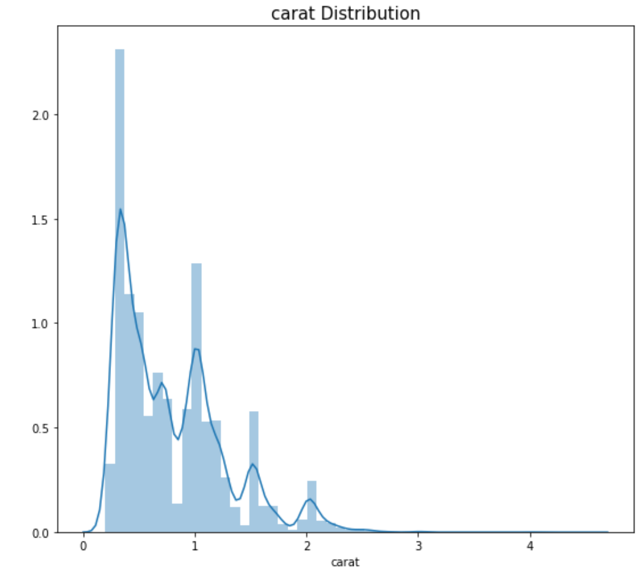
Out[15]: 0

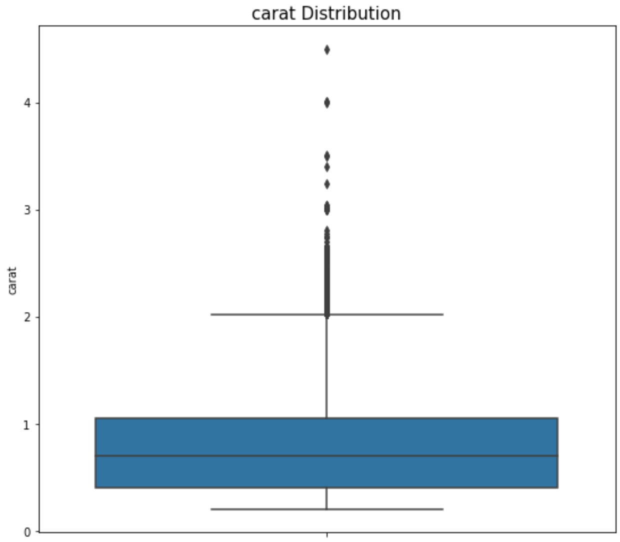
## Checking the outliers through Box plot

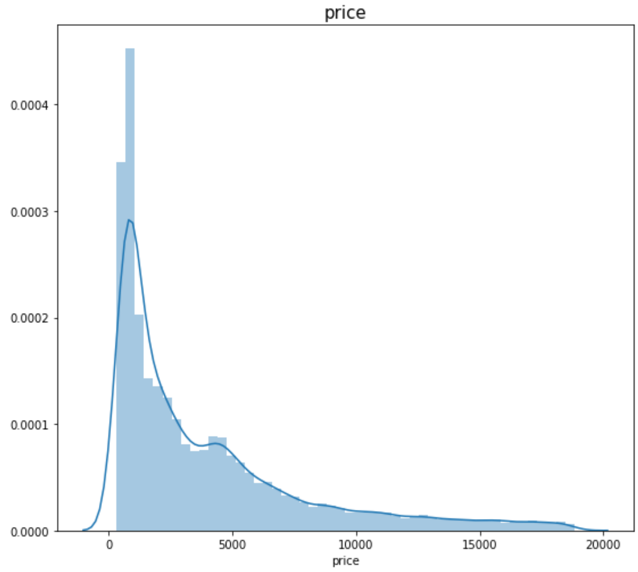
Out[16]: <matplotlib.axes._subplots.AxesSubplot at 0x18c16082608>

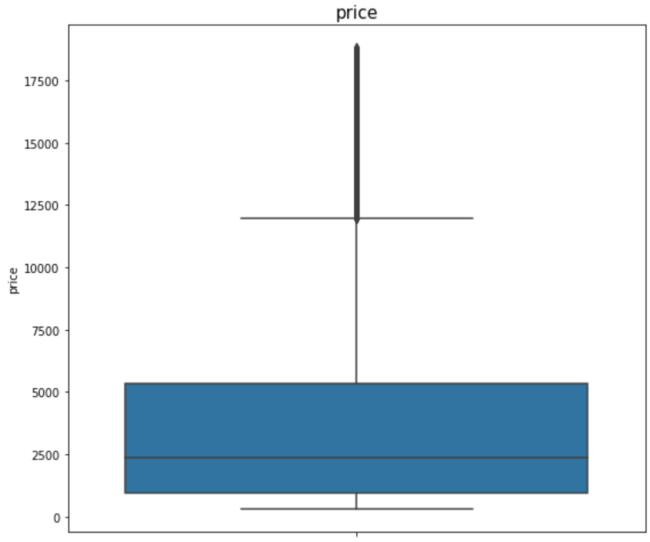



## Checking the distribution of the data




## Bi- Variate Analysis:

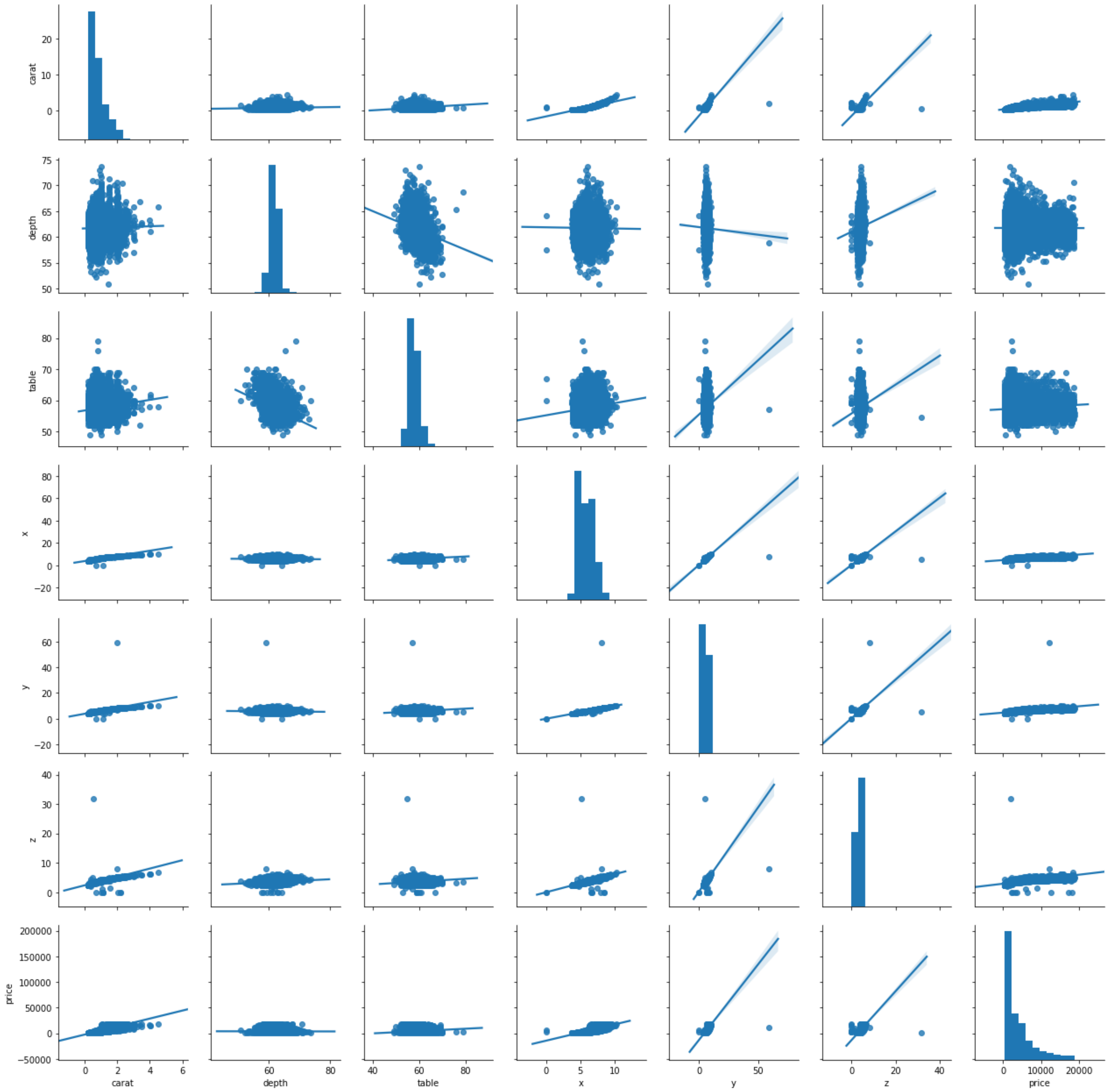
## Checking the correlation of variable

Out[18]:

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

## Plot scatter plots for every pair of attributes and histograms along the diagonal

```
C:\Users\kuldipwadhwa\Anaconda3\lib\site-packages\numpy\lib\histograms.py:824: Runtime
Warning: invalid value encountered in greater_equal
  keep = (tmp_a >= first_edge)
C:\Users\kuldipwadhwa\Anaconda3\lib\site-packages\numpy\lib\histograms.py:825: Runtime
Warning: invalid value encountered in less_equal
  keep &= (tmp_a <= last_edge)
```

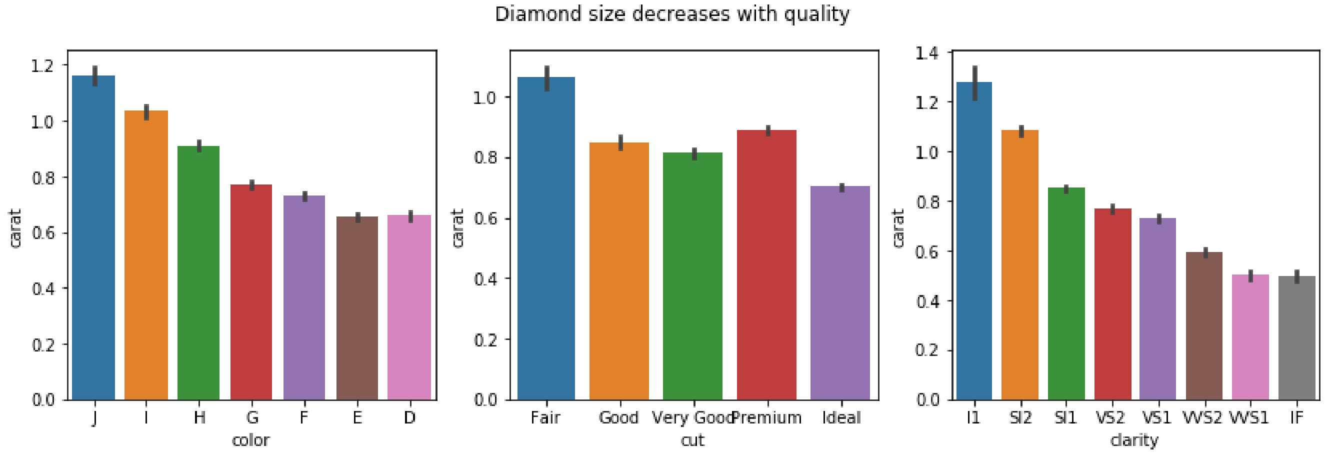


## Convering the cut,color and Clarity variables into categorial variables

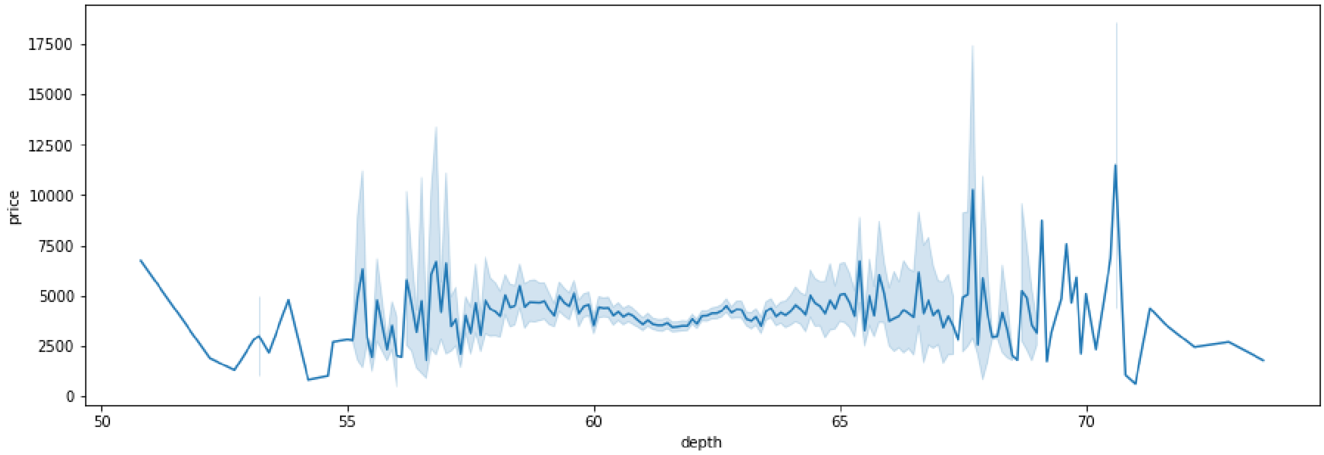
Out[21]: Text(0.5, 0.98, 'Price Respect to color,cut and clarity')


## Bar plot between carat and cut,clarity and color

Out[22]: Text(0.5, 0.98, 'Diamond size decreases with quality')

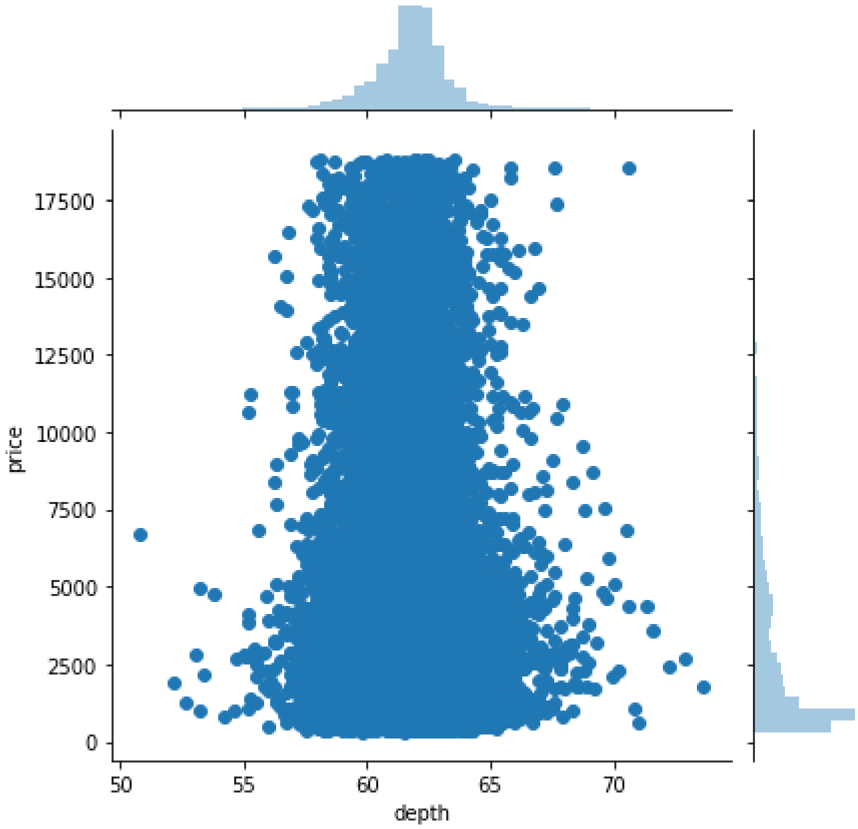



## line plot between price and depth

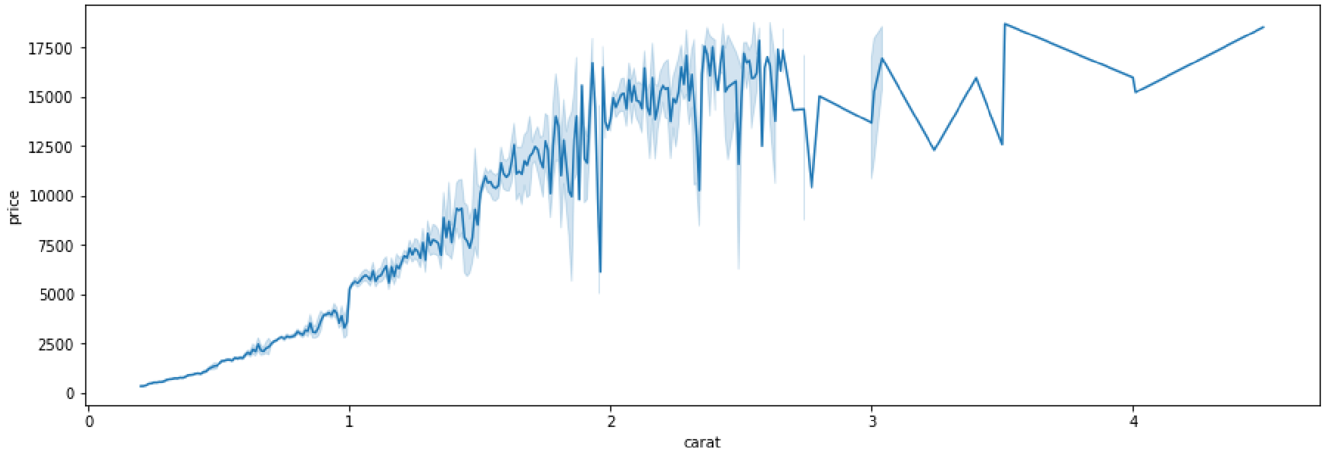



## jointplot between price and depth

<Figure size 1080x360 with 0 Axes>

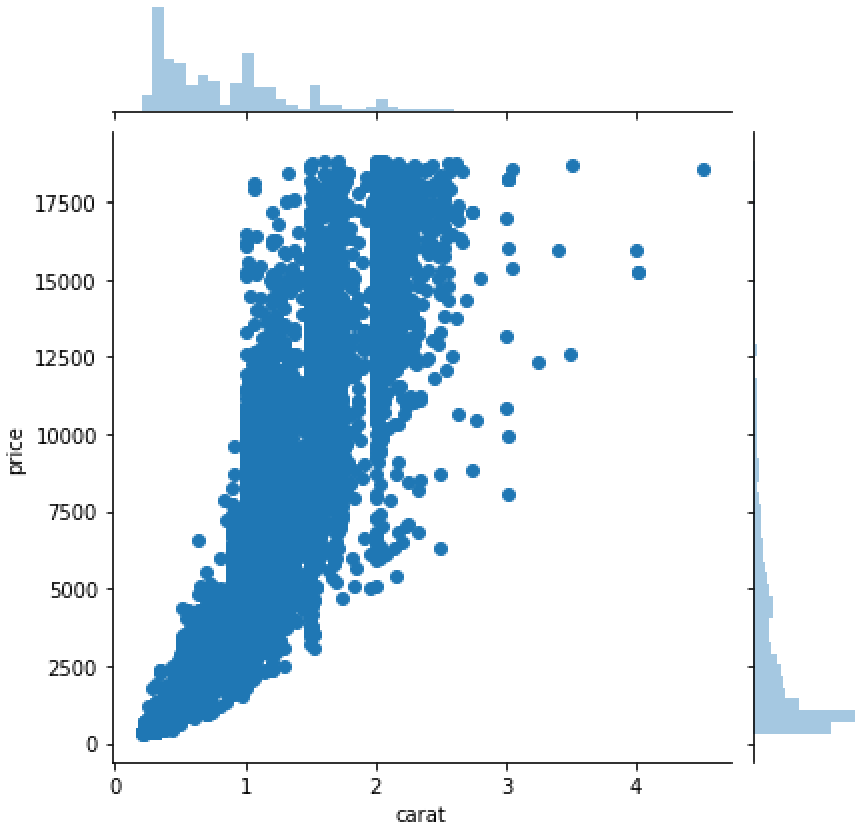



## line plot between price and carat

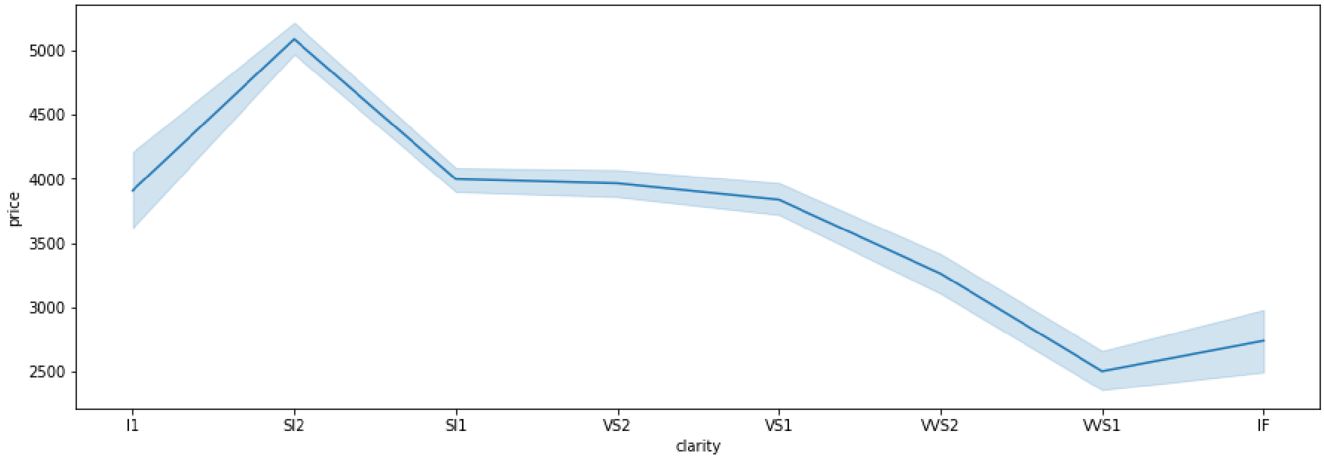



## jointplot between price and carat

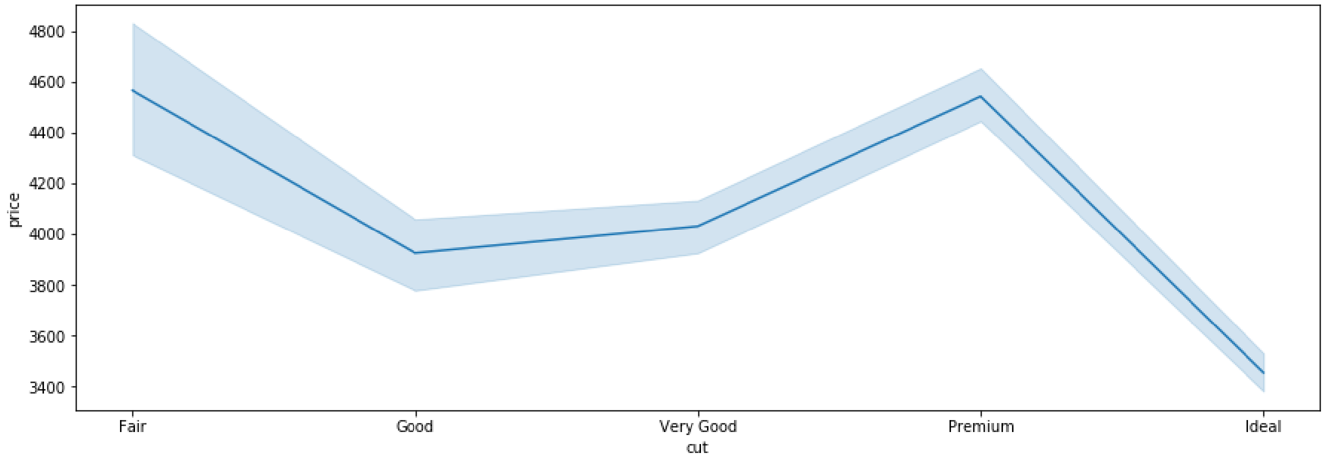
<Figure size 1080x360 with 0 Axes>



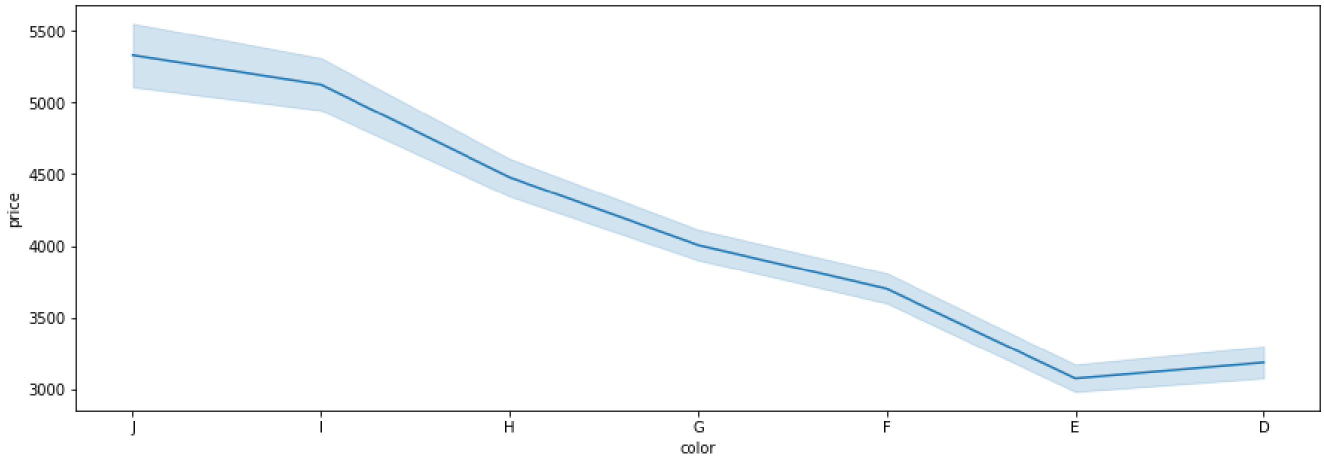

## Line plot between price and clarity



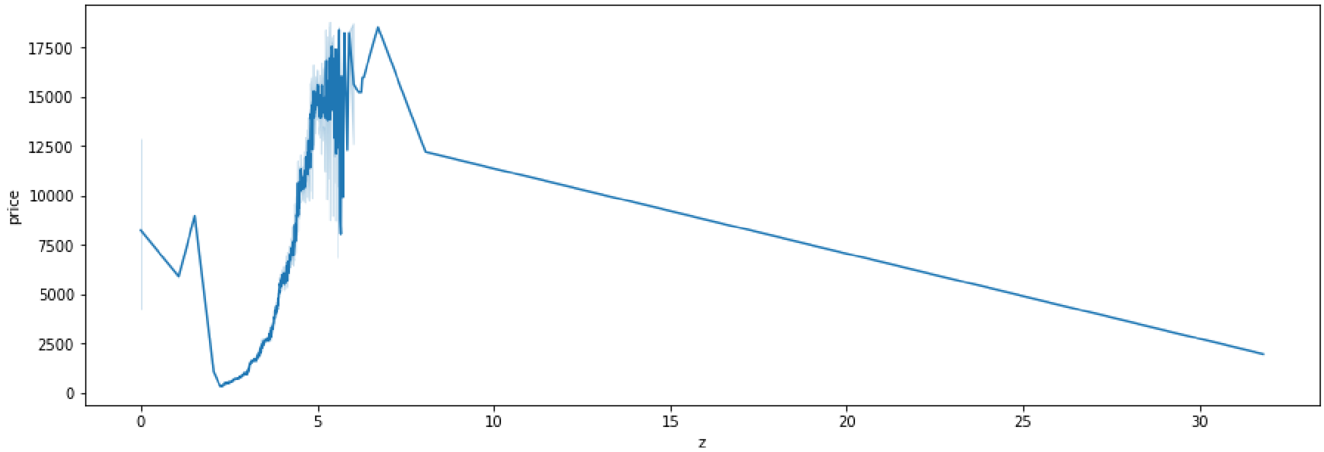

## Line plot between price and cut



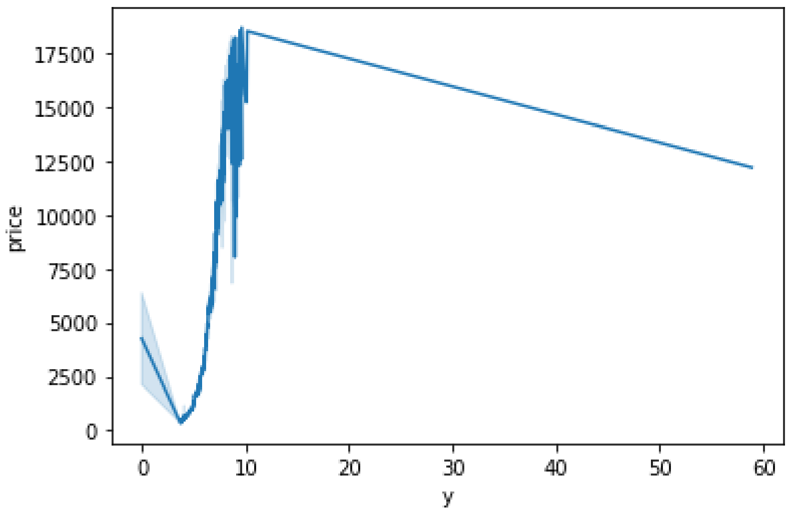

## Line plot between price and color



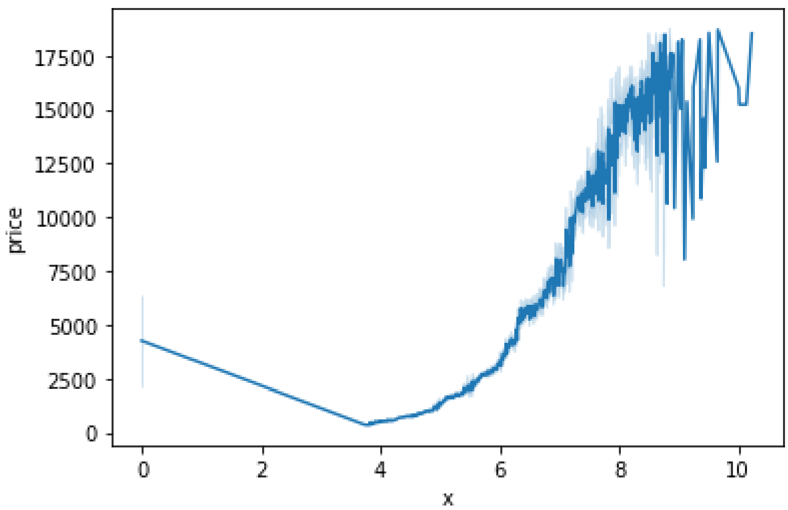

## Line plot between price and z dimension



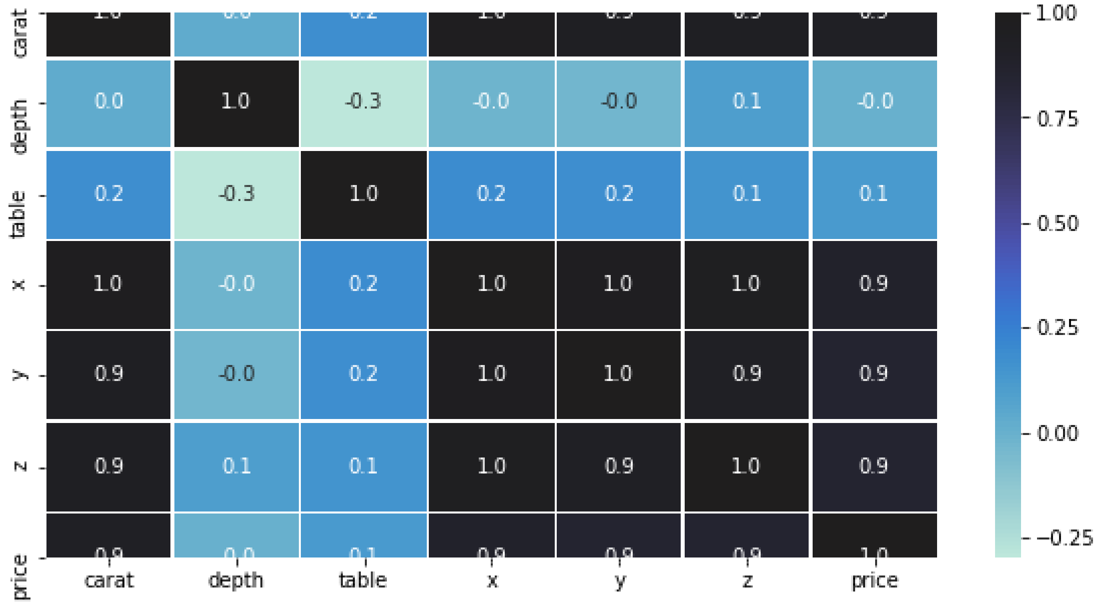

## Line plot between price and y dimension




## Line plot between price and x dimension



## Correlation among pairs of continuous variables




## Final EDA Analysis

- There are total 26967 rows and 10 columns . There are 697 null values in depth column of the dataset .
- There are zero values of x,y,z column in the dataset. which is the dimension of the dataset which is practically no possible
- price and carat variables are not normally distributed
- There is highly corelation between dimensions (x,y,z) and well as carat variables
- There is highly corelation between price and carat variables

## 1.2 Impute null values if present, also check for the values which are equal to zero. Do they have any meaning or do we need to change them or drop them? Do you think scaling is necessary in this case?

## Checking the dataset missing values?

Out[36]:

| | Total | Percent |
|---|---|---|
| depth | 697 | 0.025846 |
| price | 0 | 0.000000 |
| z | 0 | 0.000000 |
| y | 0 | 0.000000 |
| x | 0 | 0.000000 |
| table | 0 | 0.000000 |
| clarity | 0 | 0.000000 |
| color | 0 | 0.000000 |
| cut | 0 | 0.000000 |
| carat | 0 | 0.000000 |

## Imputing the null value with mean

## Checking the missing value again

Out[38]:

| | Total | Percent |
|---|---|---|
| price | 0 | 0.0 |
| z | 0 | 0.0 |
| y | 0 | 0.0 |
| x | 0 | 0.0 |
| table | 0 | 0.0 |
| depth | 0 | 0.0 |
| clarity | 0 | 0.0 |
| color | 0 | 0.0 |
| cut | 0 | 0.0 |
| carat | 0 | 0.0 |

## Checking the variables equal to 0

## Checking the whether x dimension variable equal to 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
6215    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
```

## Imputing x dimension variable equal to 0 with mean of x dimension

## Checking the whether x dimension variable equal to 0 still exist

Out[42]:

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

## Checking the whether y dimension variable equal to 0

| | carat | cut | color | clarity | depth | table | x | y | z | price |
|---|---|---|---|---|---|---|---|---|---|---|
| 5821 | 0.71 | Good | F | SI2 | 64.1 | 60.0 | 5.73 | 0.0 | 0.0 | 2130 |
| 6215 | 0.71 | Good | F | SI2 | 64.1 | 60.0 | 5.73 | 0.0 | 0.0 | 2130 |
| 17506 | 1.14 | Fair | G | VS1 | 57.5 | 67.0 | 5.73 | 0.0 | 0.0 | 6381 |

## Imputing y dimension variable equal to 0 with mean of y dimension

## Checking the whether y dimension variable equal to 0 still exist

Out[45]:

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

## Checking the whether z dimension variable equal to 0

| | carat | cut | color | clarity | depth | table | x | y | z | price |
|---|---|---|---|---|---|---|---|---|---|---|
| 5821 | 0.71 | Good | F | SI2 | 64.1 | 60.0 | 5.73 | 5.73 | 0.0 | 2130 |
| 6034 | 2.02 | Premium | H | VS2 | 62.7 | 53.0 | 8.02 | 7.95 | 0.0 | 18207 |
| 6215 | 0.71 | Good | F | SI2 | 64.1 | 60.0 | 5.73 | 5.73 | 0.0 | 2130 |
| 10827 | 2.20 | Premium | H | SI1 | 61.2 | 59.0 | 8.42 | 8.37 | 0.0 | 17265 |
| 12498 | 2.18 | Premium | H | SI2 | 59.4 | 61.0 | 8.49 | 8.45 | 0.0 | 12631 |
| 12689 | 1.10 | Premium | G | SI2 | 63.0 | 59.0 | 6.50 | 6.47 | 0.0 | 3696 |
| 17506 | 1.14 | Fair | G | VS1 | 57.5 | 67.0 | 5.73 | 5.73 | 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 |

## Checking the whether z dimension variable equal to 0 still exist

Out[48]:

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

## Checking the whether price,table,depth,carat variable equal to 0

```
Checking the whether price variable equal to 0 Empty DataFrame
Columns: [carat, cut, color, clarity, depth, table, x, y, z, price]
Index: []
Checking the whether table variable equal to 0 Empty DataFrame
Columns: [carat, cut, color, clarity, depth, table, x, y, z, price]
Index: []
Checking the whether depth variable equal to 0 Empty DataFrame
Columns: [carat, cut, color, clarity, depth, table, x, y, z, price]
Index: []
Checking the whether carat variable equal to 0 Empty DataFrame
Columns: [carat, cut, color, clarity, depth, table, x, y, z, price]
Index: []
```

## 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?

- There are 697 null values in the depth variable of the dataset . which is .02 percent of dataset.
- I have imputed with null values of depth variable with mean of the depth variables
- There are very very few entries of x,y,z variables which is equal to zero .
- It is not possible any diamond without dimensions. so we have imputed with means of th respective variable
- As far as Linear Regression is concerned with respect to scaling .
- It is totally depends what you want to achieve .
- if you want to focus on increasing the Accuracy score , In such case scaling is not required.
- Sometimes During building the Model , Intercept come out very large which is meaningless In such cases scaling is required.
- After Scaling the data Intercept would be close to 0 and There would be significant change in coefficient value.

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

## Checking the dataset information

```
<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      26967 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
```

## Converting the object variables into codes

```
feature: cut
[Ideal, Premium, Very Good, Good, Fair]
Categories (5, object): [Fair, Good, Ideal, Premium, Very Good]
[2 3 4 1 0]


feature: color
[E, G, F, D, H, J, I]
Categories (7, object): [D, E, F, G, H, I, J]
[1 3 2 0 4 6 5]


feature: clarity
[SI1, IF, VVS2, VS1, VVS1, VS2, SI2, I1]
Categories (8, object): [I1, IF, SI1, SI2, VS1, VS2, VVS1, VVS2]
[2 1 7 4 6 5 3 0]
```

## Copy all the predictor variables into X dataframe and Copy target into the y dataframe.

## Split X and y into training and test set in 70:30 ratio

Out[55]:

| | carat | cut | color | clarity | depth | table | x | y | z |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0.30 | 2 | 1 | 2 | 62.1 | 58.0 | 4.27 | 4.29 | 2.66 |
| 1 | 0.33 | 3 | 3 | 1 | 60.8 | 58.0 | 4.42 | 4.46 | 2.70 |
| 2 | 0.90 | 4 | 1 | 7 | 62.2 | 60.0 | 6.04 | 6.12 | 3.78 |
| 3 | 0.42 | 2 | 2 | 4 | 61.6 | 56.0 | 4.82 | 4.80 | 2.96 |
| 4 | 0.31 | 2 | 2 | 6 | 60.4 | 59.0 | 4.35 | 4.43 | 2.65 |

## Linear Regression Model

## Invoke the LinearRegression function and find the bestfit model on training data

Out[56]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

## Coefficients for each of the independent attributes

```
The coefficient for carat is 11115.123995871338
The coefficient for cut is 52.75700811952051
The coefficient for color is -272.89855410465265
The coefficient for clarity is 289.20568829014167
The coefficient for depth is -155.33247334643733
The coefficient for table is -95.8618330510858
The coefficient for x is -1172.329314138646
The coefficient for y is 0.477584781321203
The coefficient for z is -38.06126261614758
```

## Check the intercept for the model

.

```
The intercept for our model is 16492.338515622454
```

## R square on training data

Out[59]:
```
0.8872372986665522
```

## R square on testing data

Out[60]:
```
0.8881877672566298
```

## Finding the RMSE

Out[61]:
```
1348.1920364395191
```

## Linear Regression using statsmodels

## concatenate X and y into a single dataframe

Out[62]:

| | carat | cut | color | clarity | depth | table | x | y | z | price |
|---|---|---|---|---|---|---|---|---|---|---|
| 11687 | 0.41 | 2 | 5 | 7 | 62.3 | 56.0 | 4.77 | 4.73 | 2.96 | 1061 |
| 9728 | 1.71 | 2 | 6 | 2 | 62.8 | 57.0 | 7.58 | 7.55 | 4.75 | 6320 |
| 1936 | 0.33 | 1 | 2 | 2 | 61.8 | 62.0 | 4.40 | 4.45 | 2.74 | 536 |
| 26220 | 0.70 | 4 | 4 | 2 | 62.8 | 57.0 | 5.61 | 5.66 | 3.54 | 2214 |
| 18445 | 0.70 | 2 | 0 | 3 | 62.1 | 56.0 | 5.67 | 5.71 | 3.53 | 2575 |

Out[63]:
```
Index(['carat', 'cut', 'color', 'clarity', 'depth', 'table', 'x', 'y', 'z',
       'price'],
      dtype='object')
```

Out[65]:
```
Intercept    16492.338516
carat        11115.123996
cut             52.757008
color         -272.898554
clarity        289.205688
depth         -155.332473
table          -95.861833
x            -1172.329314
y                0.477585
z              -38.061263
dtype: float64
```

## Summary of variables

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.887
Model:                            OLS   Adj. R-squared:                  0.887
Method:                 Least Squares   F-statistic:                 1.649e+04
Date:                Sun, 05 Jul 2020   Prob (F-statistic):               0.00
Time:                        20:23:31   Log-Likelihood:            -1.6285e+05
No. Observations:               18876   AIC:                         3.257e+05
Df Residuals:                   18866   BIC:                         3.258e+05
Df Model:                           9
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept   1.649e+04    684.790     24.084      0.000    1.52e+04    1.78e+04
carat       1.112e+04    101.941    109.035      0.000    1.09e+04    1.13e+04
cut           52.7570      9.770      5.400      0.000      33.607      71.907
color       -272.8986      6.055    -45.067      0.000    -284.768    -261.030
clarity      289.2057      5.899     49.030      0.000     277.644     300.767
depth       -155.3325      8.187    -18.973      0.000    -171.379    -139.286
table        -95.8618      4.750    -20.182      0.000    -105.172     -86.552
x          -1172.3293     55.660    -21.062      0.000   -1281.427   -1063.231
y              0.4776     26.538      0.018      0.986     -51.540      52.495
z            -38.0613     45.977     -0.828      0.408    -128.181      52.058
==============================================================================
Omnibus:                     4466.273   Durbin-Watson:                   1.974
Prob(Omnibus):                  0.000   Jarque-Bera (JB):           180775.451
Skew:                           0.367   Prob(JB):                         0.00
Kurtosis:                      18.143   Cond. No.                     5.94e+03
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly speci
fied.
[2] The condition number is large, 5.94e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

## Calculate MSE

Out[67]: 1823875.9900153258

## Root Mean Squared Error - RMSE

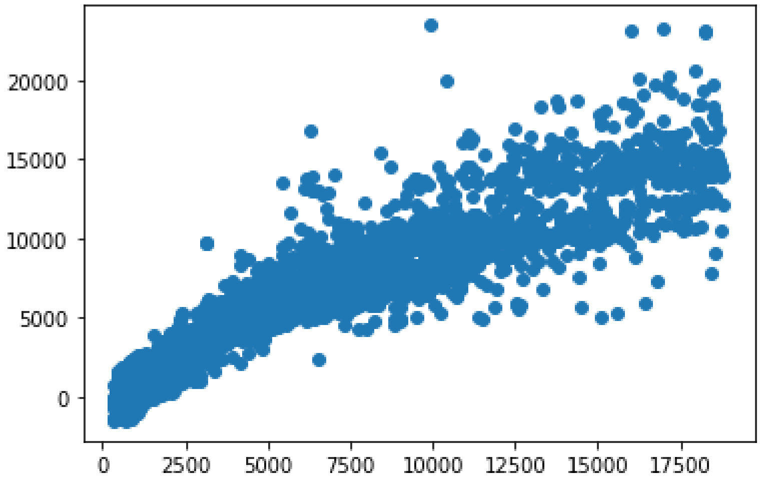
Out[68]: 1350.509529775827

## Another way of calculating RMSE

Out[69]: 1350.8674038923411

## Prediction on Test data

Out[70]:
```
18031    12810.713359
26051     9374.943508
16279      392.528969
16466      722.528016
19837     6918.351029
dtype: float64
```




Out[72]:
```
Intercept    16492.338516
carat        11115.123996
cut             52.757008
color         -272.898554
clarity        289.205688
depth         -155.332473
table          -95.861833
x            -1172.329314
y                0.477585
z              -38.061263
dtype: float64
```

```
(16492.34) * Intercept + (11115.12) * carat + (52.76) * cut + (-272.9) * color + (289.
21) * clarity + (-155.33) * depth + (-95.86) * table + (-1172.33) * x + (0.48) * y +
(-38.06) * z +
```

## ITERATION 2 (Model with scaling )

## Applying the Z score

## Invoke the LinearRegression function and find the bestfit model on training data

Out[75]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

## Coefficients for each of the independent attributes

```
The coefficient for carat is 1.32132067688744
The coefficient for cut is 0.013483663639021256
The coefficient for color is -0.11573943460177429
The coefficient for clarity is 0.12352934255900051
The coefficient for depth is -0.053881753553561805
The coefficient for table is -0.05310584855621543
The coefficient for x is -0.32836815367635175
The coefficient for y is 0.00014051567802965704
The coefficient for z is -0.006876911994713354
```

## Intercept of Model

```
The intercept for our model is -8.461705207477702e-17
```

## Model score

Out[78]:
```
0.8882979838903627
```

## Calculate MSE

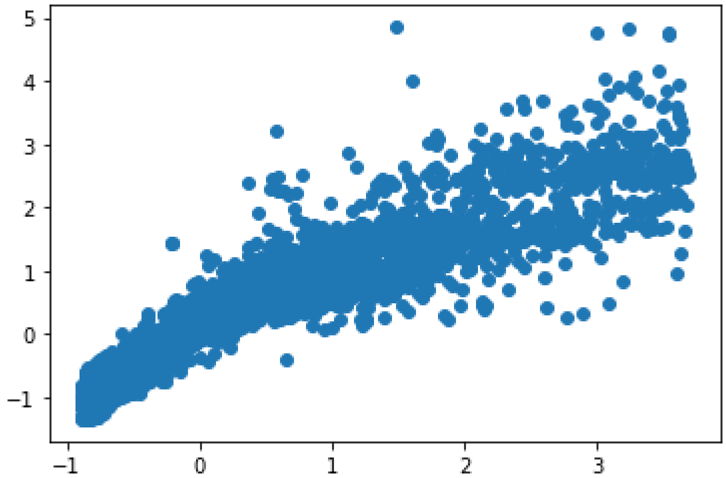
## Calculate the RMSE

Out[80]:
```
0.334218515509894
```

## predict price

## Regression, plot the predicted y value vs actual y values for the test data

Out[82]:
```
<matplotlib.collections.PathCollection at 0x18c190bef08>
```




## Linear Regression using statsmodels

## Concatenate X and y into a single dataframe

Out[83]:

| | carat | cut | color | clarity | depth | table | x | y | z | |
|---|---|---|---|---|---|---|---|---|---|---|
| 11687 | -0.811481 | -0.546431 | 1.401996 | 1.852795 | 0.396713 | -0.655388 | -0.851728 | -0.848860 | -0.797139 | -0.7 |
| 9728 | 1.907677 | -0.546431 | 1.988276 | -1.057867 | 0.755120 | -0.206551 | 1.642751 | 1.534337 | 1.666221 | 0.5 |
| 1936 | -0.978814 | -1.519306 | -0.356843 | -1.057867 | 0.038307 | 2.037633 | -1.180183 | -1.085489 | -1.099898 | -0.8 |
| 26220 | -0.204900 | 1.399320 | 0.815717 | -1.057867 | 0.755120 | -0.206551 | -0.106048 | -0.062912 | 0.001045 | -0.4 |
| 18445 | -0.204900 | -0.546431 | -1.529403 | -0.475735 | 0.253351 | -0.655388 | -0.052785 | -0.020657 | -0.012717 | -0.3 |

Out[84]:

```
Index(['carat', 'cut', 'color', 'clarity', 'depth', 'table', 'x', 'y', 'z',
       'price'],
      dtype='object')
```

## Coefficients and Intercept

Out[86]:

```
Intercept    3.642919e-17
carat        1.321321e+00
cut          1.348366e-02
color       -1.157394e-01
clarity      1.235293e-01
depth       -5.388175e-02
table       -5.310585e-02
x           -3.283682e-01
y            1.405157e-04
z           -6.876912e-03
dtype: float64
```

## Stats Summary

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.887
Model:                            OLS   Adj. R-squared:                  0.887
Method:                 Least Squares   F-statistic:                 1.649e+04
Date:                Sun, 05 Jul 2020   Prob (F-statistic):               0.00
Time:                        20:23:36   Log-Likelihood:                -6185.7
No. Observations:               18876   AIC:                         1.239e+04
Df Residuals:                   18866   BIC:                         1.247e+04
Df Model:                           9
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept   3.643e-17      0.002   1.49e-14      1.000      -0.005       0.005
carat          1.3213      0.012    109.035      0.000       1.298       1.345
cut            0.0135      0.002      5.400      0.000       0.009       0.018
color         -0.1157      0.003    -45.067      0.000      -0.121      -0.111
clarity        0.1235      0.003     49.030      0.000       0.119       0.128
depth         -0.0539      0.003    -18.973      0.000      -0.059      -0.048
table         -0.0531      0.003    -20.182      0.000      -0.058      -0.048
x             -0.3284      0.016    -21.062      0.000      -0.359      -0.298
y              0.0001      0.008      0.018      0.986      -0.015       0.015
z             -0.0069      0.008     -0.828      0.408      -0.023       0.009
==============================================================================
Omnibus:                     4466.273   Durbin-Watson:                   1.974
Prob(Omnibus):                  0.000   Jarque-Bera (JB):           180775.451
Skew:                           0.367   Prob(JB):                         0.00
Kurtosis:                      18.143   Cond. No.                         15.5
==============================================================================

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

## Calculate MSE

Out[88]:
```
0.11276270133344801
```

## Root Mean Squared Error - RMSE

Out[89]:
```
0.33580158030218976
```

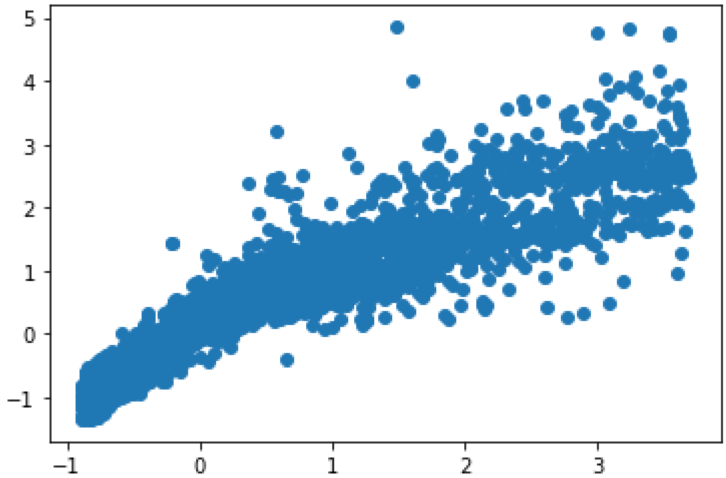
## Another way of calculating RMSE

Out[90]:
```
0.335890565008498
```

## Prediction on Test data

Out[91]:
```
18031    2.207040
26051    1.350694
16279   -0.889244
16466   -0.811610
19837    0.735777
dtype: float64
```

## Regression, plot the predicted y value vs actual y values for the test data



## Coefficient of variables

```
Out[93]: Intercept    3.642919e-17
         carat        1.321321e+00
         cut          1.348366e-02
         color       -1.157394e-01
         clarity      1.235293e-01
         depth       -5.388175e-02
         table       -5.310585e-02
         x           -3.283682e-01
         y            1.405157e-04
         z           -6.876912e-03
         dtype: float64
```

## Final Linear Equation

```
(0.0) * Intercept + (1.32) * carat + (0.01) * cut + (-0.12) * color + (0.12) * clarity
+ (-0.05) * depth + (-0.05) * table + (-0.33) * x + (0.0) * y + (-0.01) * z +
```

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

## Linear Regression Performance Metrics

- After scaling the data , intercept has become close to 0 and there is significant change in coefficient values
- After scaling the data, Accuracy score did not change as result
- R square on training data is 88 %
- R square on test data is 88 %
- Value of RMSE(Scaled data) is .33
- Value of MSE(Scaled date) is .11
- Final Linear Equation as below
- (0.0) * Intercept + (1.32) * carat + (0.01) * cut + (-0.12) * color + (0.12) * clarity + (-0.05) * depth + (-0.05) * table + (-0.33) * x + (0.0) * y + (-0.01) * z +

- e.g carat will increase by 1 unit price will be influenced by 1.32 unit

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

## Business Insights and Recommendations.

- From the plot between carat and price it is quite clear if carat is in small to medium then price will be in upward trend but there are some exception where carat size is high when price is high
- From plot between clarity and price , it is quite evident .clarity and price is in zigzag trend .if we consider this sequence I1', 'SI2', 'SI1', 'VS2','VS1', 'VVS2', 'VVS1', 'IF' price will move upward direction from I1', 'SI2 and start downward from SI1 to VVS1 and will bit up to VVS1 to IF
- From the plot between color and price it is quite evident .if we consider the sequence J', 'I', 'H', 'G', 'F', 'E', 'D' color and price is in downward trend and slightly up from E to D
- From the plot between cut and price it is quite evident .if we consider the sequence Fair', 'Good', 'Very Good', 'Premium', 'Ideal cut and price is in zigzag trend , price is showing bit downside from fair to good cut , price is upward trend from good to premium and slightly down from premium to ideal
- The final Linear Regression equation is
- (0.0) * Intercept + (1.32) * carat + (0.01) * cut + (-0.12) * color + (0.12) * clarity + (-0.05) * depth + (-0.05) * table + (-0.33) * x + (0.0) * y + (-0.01) * z +

- There are factors which are postively influence the price . Most important postive factor is Carat ,Clarity,Cut
- There are factors which are Negatively influence the price table,depth,color,x
- We should focus on more on Carat , Clarity and Cut and try to have more controlled on table , depth, color and x

## 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.

## Removing unwanted column

## Reading the top 5 Records

Out[97]:

| | 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 the Holiday data

Out[98]:

| | 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 |

## Check the data set information

```
<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
```

## Check the columns

Out[100]:
```
Index(['Holliday_Package', 'Salary', 'age', 'educ', 'no_young_children',
       'no_older_children', 'foreign'],
      dtype='object')
```

## Checking the shape of data?

Out[101]:
```
(872, 7)
```

## Count the datatypes?

Out[102]:
```
int64     5
object    2
dtype: int64
```

## Checking the dataset missing values?

Out[103]:

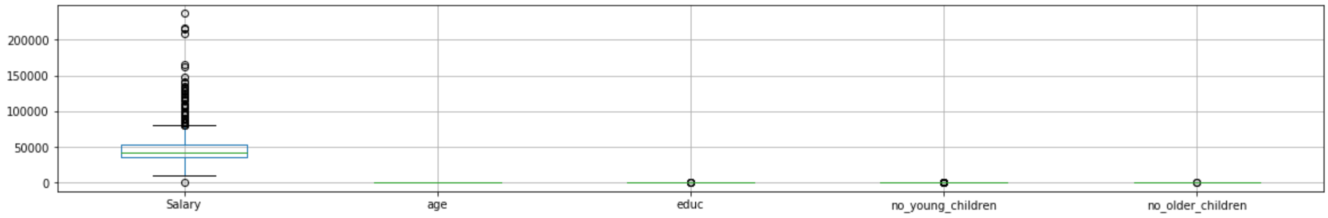
| | Total | Percent |
|---|---|---|
| foreign | 0 | 0.0 |
| no_older_children | 0 | 0.0 |
| no_young_children | 0 | 0.0 |
| educ | 0 | 0.0 |
| age | 0 | 0.0 |
| Salary | 0 | 0.0 |
| Holliday_Package | 0 | 0.0 |

## Checking the Duplicates Values

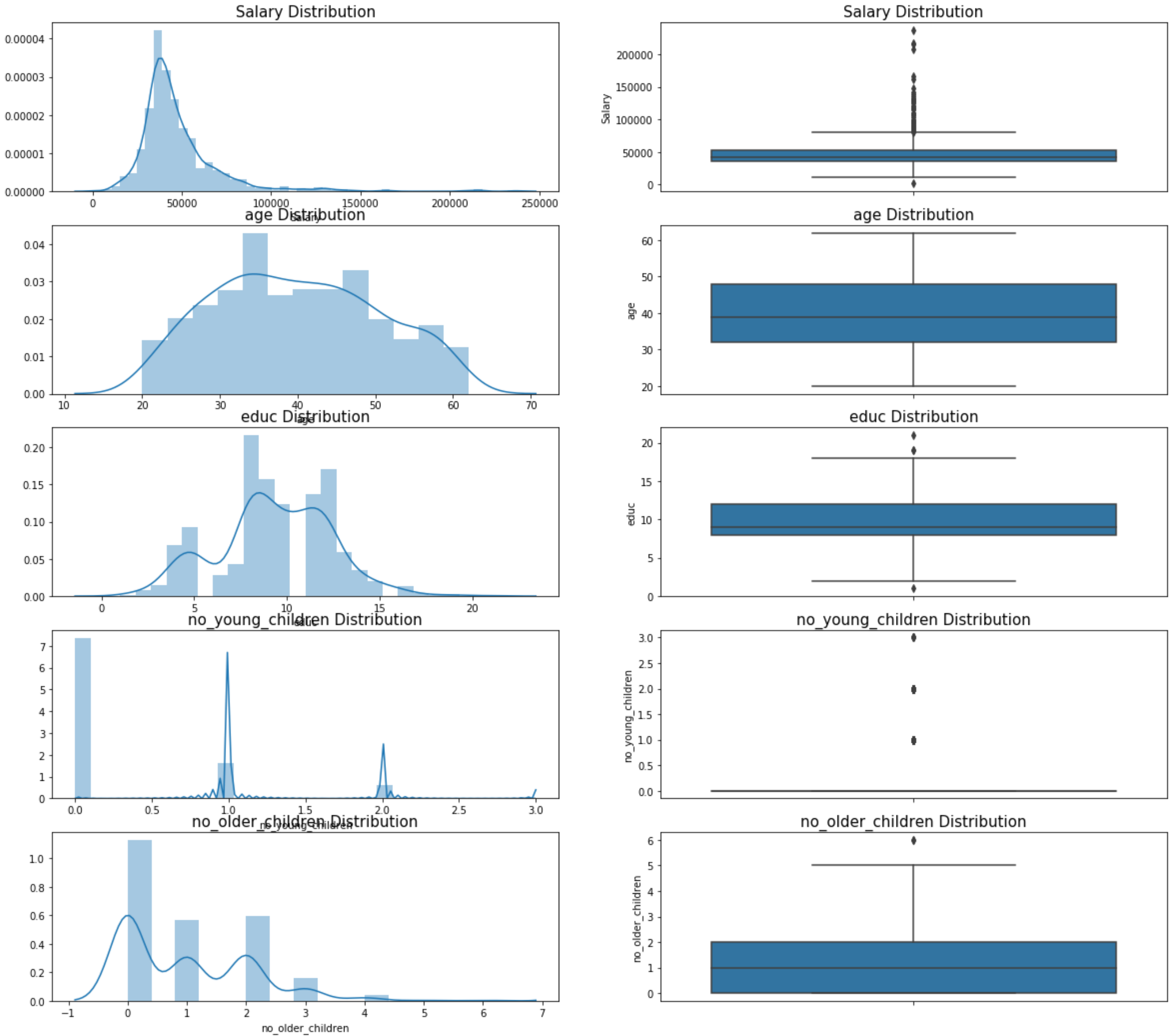
Out[104]: 0

## Checking the outliers through Box plot

Out[105]: <matplotlib.axes._subplots.AxesSubplot at 0x18c18fea3c8>




## Checking the distribution of the data


Salary Distribution
Salary
Salary Distribution
Salary
age Distribution
age
age Distribution
age
educ Distribution
educ
educ Distribution
educ
no_young_children Distribution
no_young_children
no_young_children Distribution
no_young_children
no_older_children Distribution
no_older_children
no_older_children Distribution
no_older_children

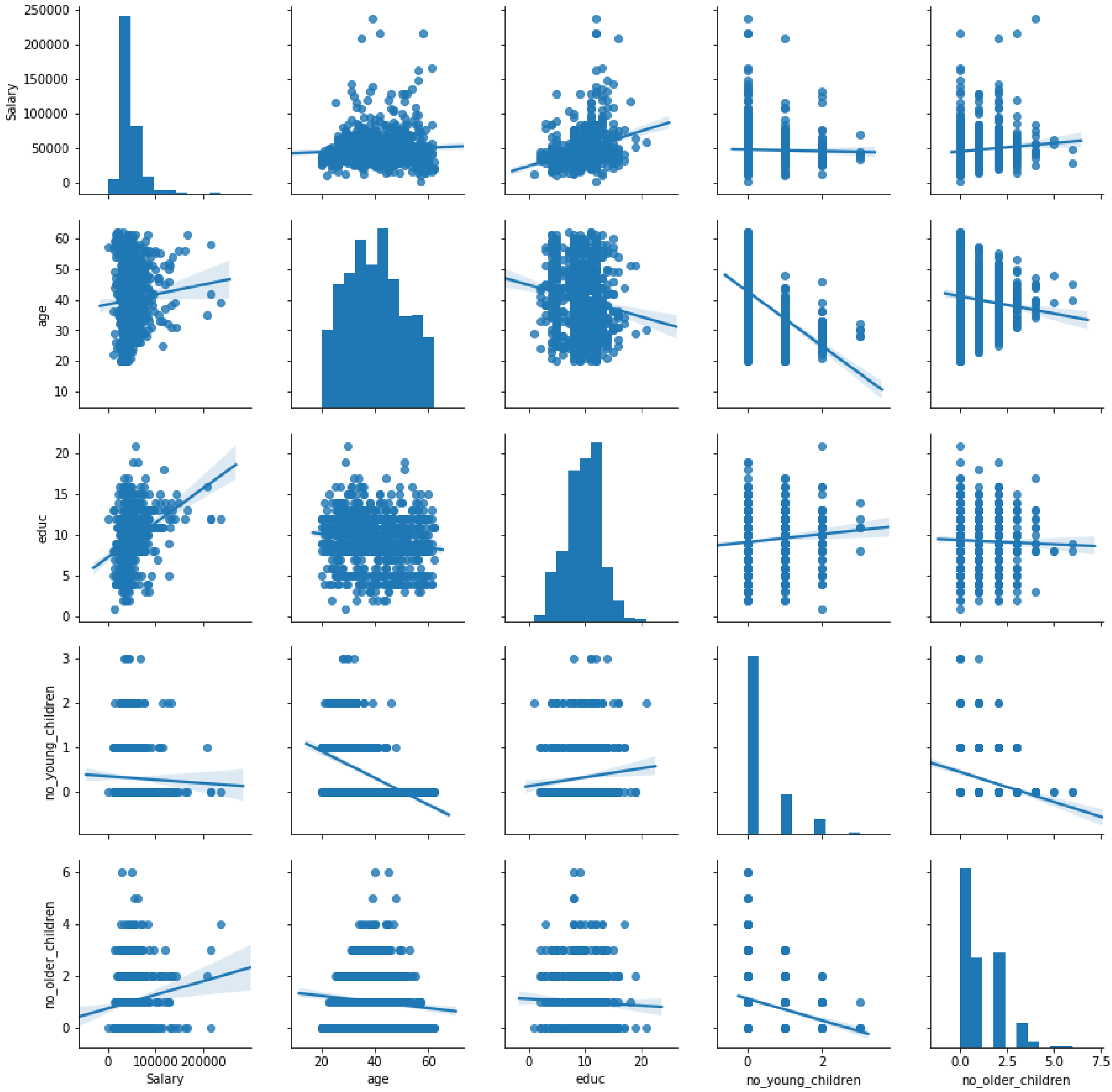
# Bi- Variate Analysis:

## Checking the correlation of variable

Out[107]:

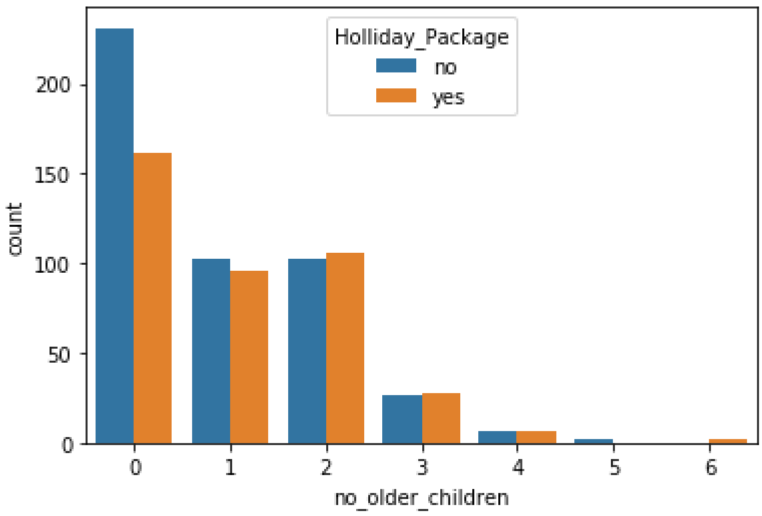
| | Salary | age | educ | no_young_children | no_older_children |
|---|---|---|---|---|---|
| Salary | 1.000000 | 0.071709 | 0.326540 | -0.029664 | 0.113772 |
| age | 0.071709 | 1.000000 | -0.149294 | -0.519093 | -0.116205 |
| educ | 0.326540 | -0.149294 | 1.000000 | 0.098350 | -0.036321 |
| no_young_children | -0.029664 | -0.519093 | 0.098350 | 1.000000 | -0.238428 |
| no_older_children | 0.113772 | -0.116205 | -0.036321 | -0.238428 | 1.000000 |

## Plots for every pair of attributes



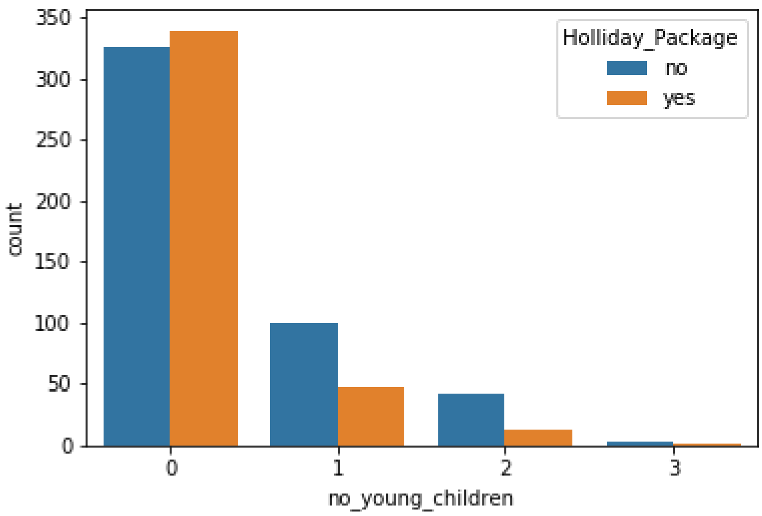

## Countplot between Holiday Package and No of older children

Out[109]: <matplotlib.axes._subplots.AxesSubplot at 0x18c177d5308>




## Countplot between Holiday Package and No of young children

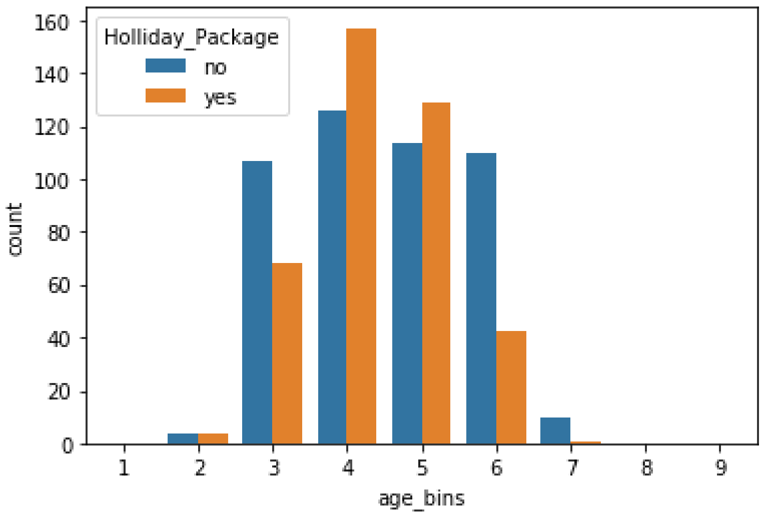
Out[110]: <matplotlib.axes._subplots.AxesSubplot at 0x18c169b0ec8>




## Dividing the age variable into number of bins for Better understanding

## Countplot between Holiday Package and age

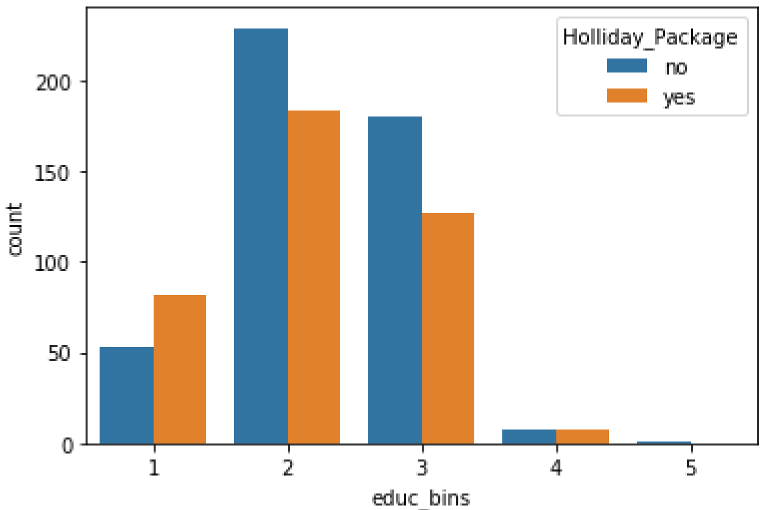
Out[112]: <matplotlib.axes._subplots.AxesSubplot at 0x18c16a52308>




## Dividing the Education variable into number of bins for Better understanding

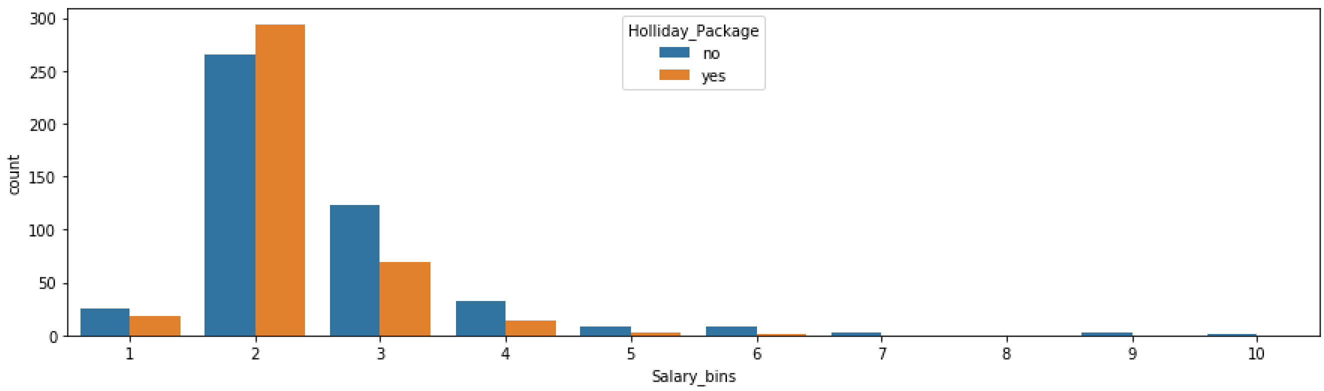
## Countplot between Holiday Package and Salary

Out[114]: <matplotlib.axes._subplots.AxesSubplot at 0x18c16af4fc8>



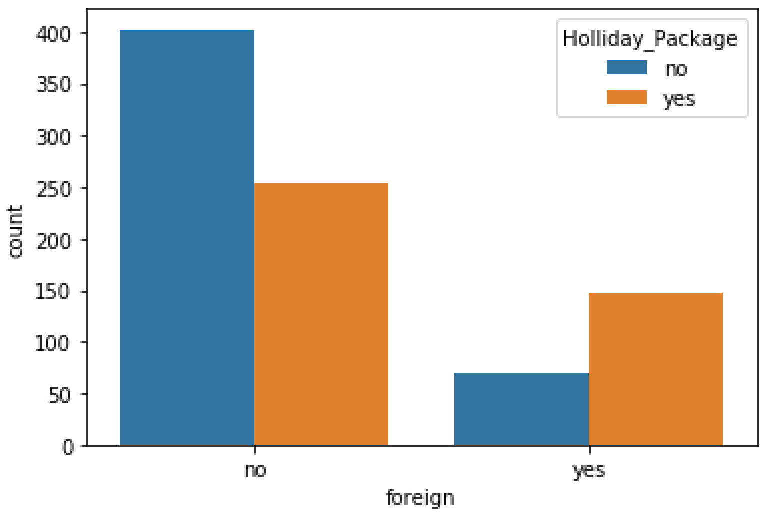

## Dividing the Salary variable into number of bins for Better understanding

## Countplot between Holiday Package and Salary




## Countplot between Holiday Package and Foreigner

Out[117]: <matplotlib.axes._subplots.AxesSubplot at 0x18c16c5ba88>

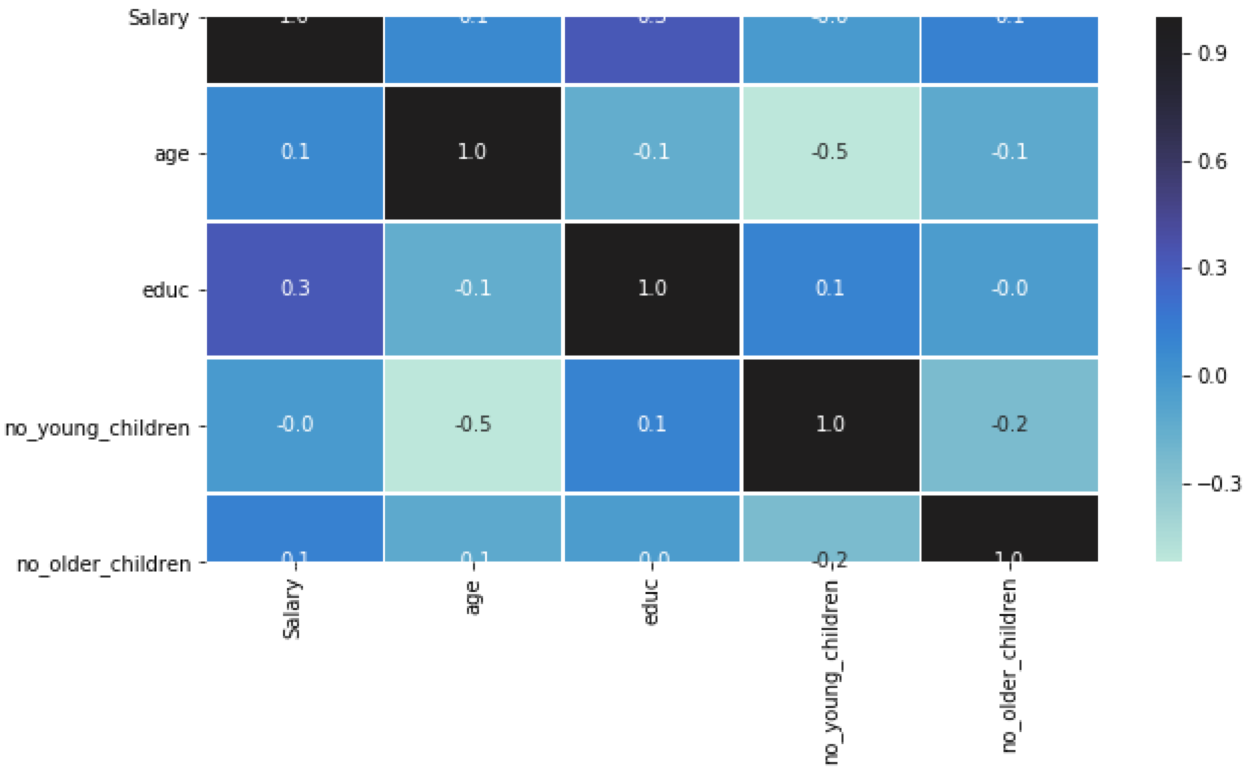



## Checking how many people are foreigner against Holiday Package

Out[118]:

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

## Correlation among pairs of continuous variables




## Exploratory Data Analysis.

- There are total 872 rows and 7 columns . There is no null values in the dataset
- Age ,Salary , Education,No of young children and no of older children are not normally distrubuted
- Age ,Salary , Education,No of young children and no of older children having outliers
- There is no duplcates values in the dataset
- There is slightly postive corelation between salary and education
- There is slightly negative corelation between age and No of young children
- There is slightly negative co relation between No of older children and No of young 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).

## Removing the unwanted column

Out[122]:

| | 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 |

## Geting unique counts of all Objects

Out[123]:

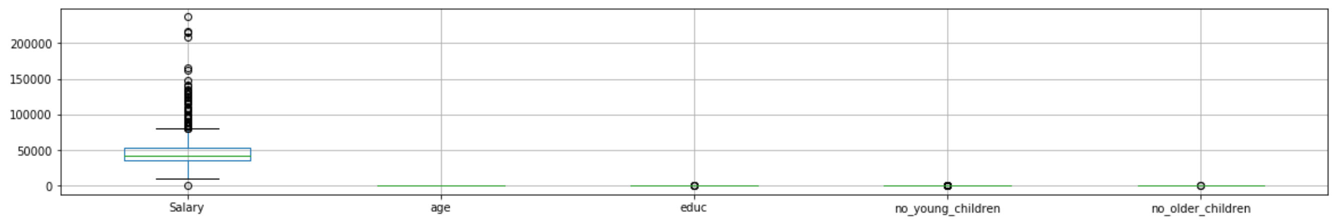
```
no     0.540138
yes    0.459862
Name: Holliday_Package, dtype: float64

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


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

## Checking the outliers through Box plot

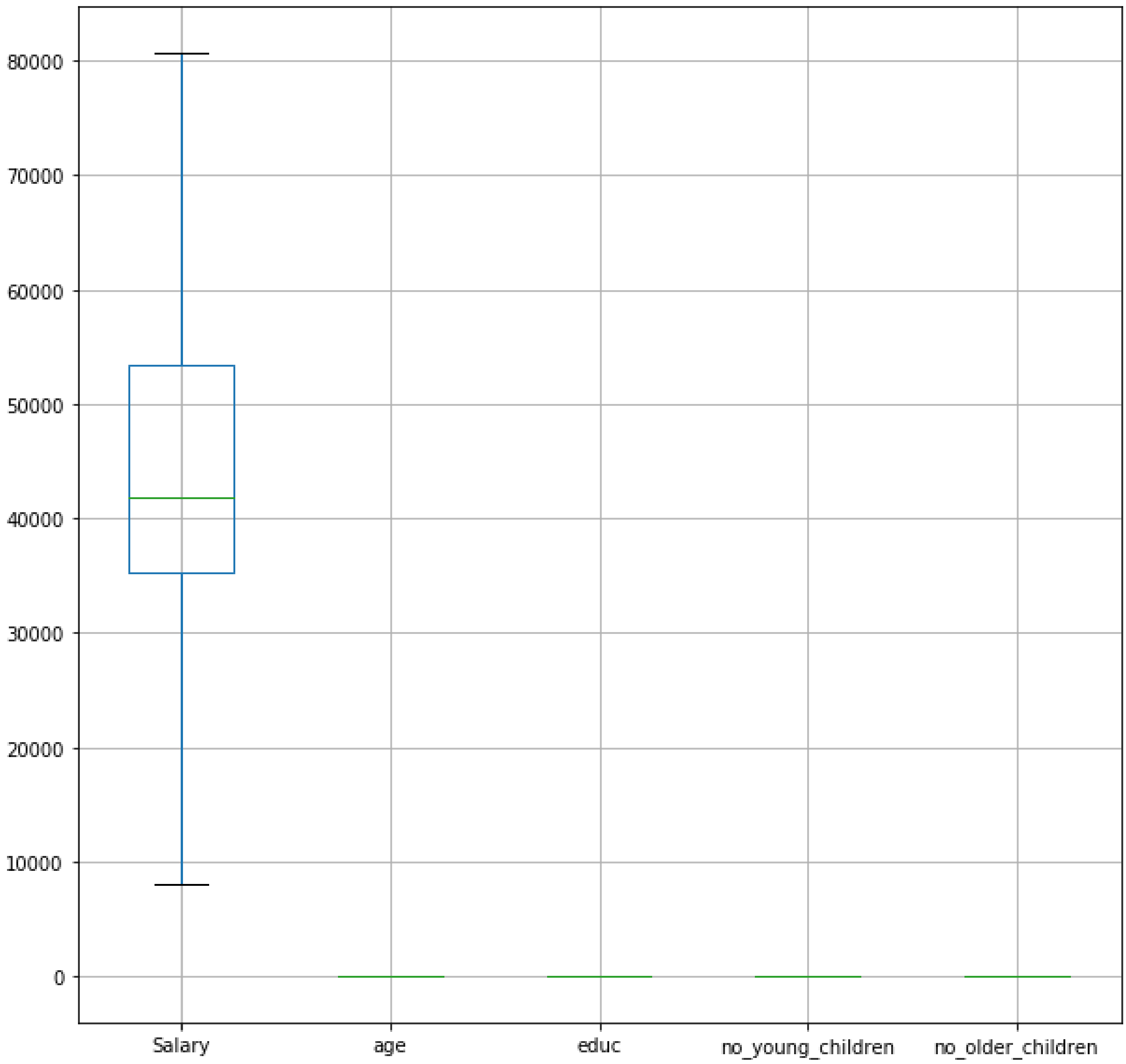
Out[125]: <matplotlib.axes._subplots.AxesSubplot at 0x18c17081748>




## Treating Outliers

## Construct box plot for continuous variables

Out[128]: <matplotlib.axes._subplots.AxesSubplot at 0x18c176f9608>




## Converting object type variables into numeric variables

```
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 Split (Logistic Regression)

## Split X and y into training and test set in 70:30 ratio (Logistic Regression)

## Fit the Logistic Regression model(Logistic Regression)

```
C:\Users\kuldipwadhwa\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:43
2: FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify a solver
to silence this warning.
  FutureWarning)
```

Out[132]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=None, solver='warn', tol=0.0001, verbose=0,
                   warm_start=False)
```

## Predicting on Training and Test dataset(Logistic Regression)

## Getting the Predicted Classes and Probs(Logistic Regression)

Out[134]:

| | 0 | 1 |
|---|---|---|
| 0 | 0.616825 | 0.383175 |
| 1 | 0.538392 | 0.461608 |
| 2 | 0.555352 | 0.444648 |
| 3 | 0.625798 | 0.374202 |
| 4 | 0.481370 | 0.518630 |

## Model Evaluation(Logistic Regression)

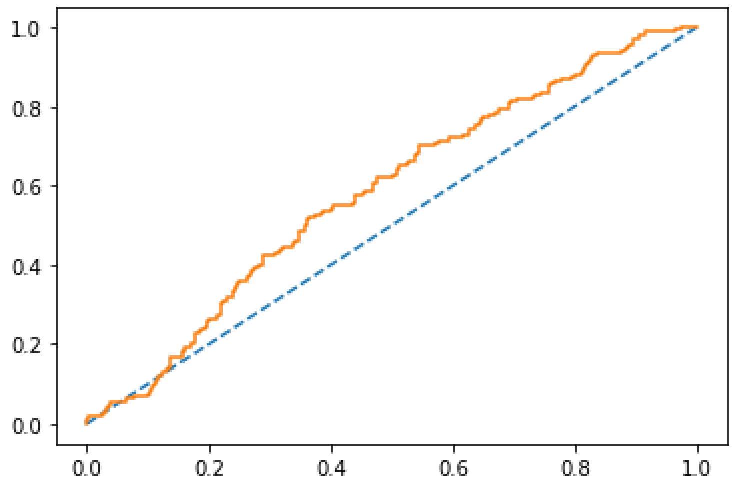
## Training Data Accuracy (Logistic Regression)

Out[135]:
```
0.5229508196721312
```

## AUC and ROC for the training data(Logistic Regression)

AUC: 0.583

Out[136]: [<matplotlib.lines.Line2D at 0x18c1cd57648>]



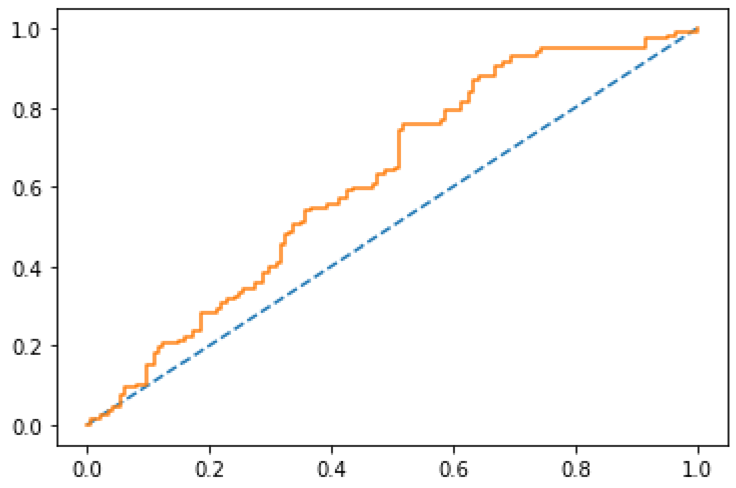

## Accuracy Test Data(Logistic Regression)

Out[137]: 0.5534351145038168

## AUC and ROC for the test data(Logistic Regression)

AUC: 0.619

Out[138]: [<matplotlib.lines.Line2D at 0x18c16714748>]




## Confusion Matrix for the training data (Logistic Regression)

```
Out[139]: array([[301,  25],
                 [266,  18]], dtype=int64)
```

## Classification Report of Training data (Logistic Regression)

```
              precision    recall  f1-score   support

           0       0.53      0.92      0.67       326
           1       0.42      0.06      0.11       284

    accuracy                           0.52       610
   macro avg       0.47      0.49      0.39       610
weighted avg       0.48      0.52      0.41       610
```

## Training Data Matrics(Logistic Regression)

```
Logistic_train_precision  0.42
Logistic_train_recall  0.06
Logistic_train_f1  0.11
```

## Confusion Matrix for test data( (Logistic Regression)

Out[142]:
```
array([[133,  12],
       [105,  12]], dtype=int64)
```

## Test data Accuracy (Logistic Regression)

Out[143]:
```
0.5534351145038168
```

## Classification Report of Test data (Logistic Regression)

```
              precision    recall  f1-score   support

           0       0.56      0.92      0.69       145
           1       0.50      0.10      0.17       117

    accuracy                           0.55       262
   macro avg       0.53      0.51      0.43       262
weighted avg       0.53      0.55      0.46       262
```

## Test Metrics (Logistic Regression)

```
Logistic_test_precision  0.5
Logistic_test_recall  0.1
Logistic_test_f1  0.17
```

## Train and Test Performance (Logistic Regression)

Out[146]:

| | Logistic Train | Logistic Test |
|---|---|---|
| Accuracy | 0.52 | 0.55 |
| AUC | 0.58 | 0.62 |
| Recall | 0.06 | 0.10 |
| Precision | 0.42 | 0.50 |
| F1 Score | 0.11 | 0.17 |

## LDA Model

```
C:\Users\kuldipwadhwa\Anaconda3\lib\site-packages\sklearn\discriminant_analysis.py:38
8: UserWarning: Variables are collinear.
  warnings.warn("Variables are collinear.")
```

Out[147]:

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

## Predicting on Training and Test dataset(LDA)

## Getting the Predicted Classes and Probs(LDA)

Out[149]:

| | 0 | 1 |
|---|---|---|
| 0 | 0.701405 | 0.298595 |
| 1 | 0.326676 | 0.673324 |
| 2 | 0.625464 | 0.374536 |
| 3 | 0.691134 | 0.308866 |
| 4 | 0.360268 | 0.639732 |

## LDA Model Evaluation

## Training Data Accuracy (LDA)

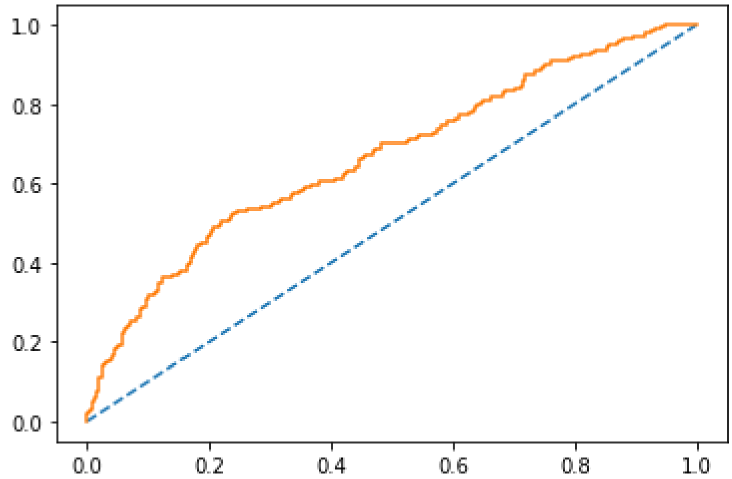
Out[150]:

```
0.6426229508196721
```

## AUC and ROC for the training data(LDA)

AUC: 0.667

Out[151]: [<matplotlib.lines.Line2D at 0x18c166bc248>]



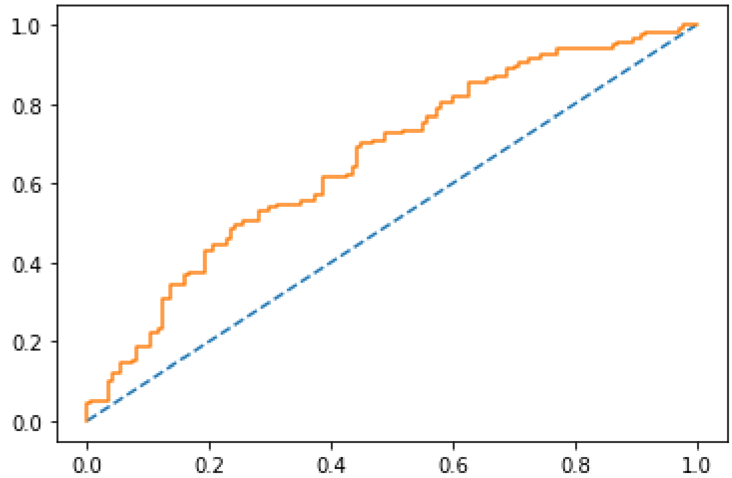

## Test Data Accuracy (LDA)

Out[152]: 0.6297709923664122

## AUC and ROC for the test data (LDA)

AUC: 0.662

Out[153]: [<matplotlib.lines.Line2D at 0x18c17046dc8>]




## Confusion Matrix for the training data(LDA)

Out[154]:
```
array([[269,  57],
       [161, 123]], dtype=int64)
```

## Classification Report of training data (LDA)

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.63 | 0.83 | 0.71 | 326 |
| 1 | 0.68 | 0.43 | 0.53 | 284 |
| accuracy | | | 0.64 | 610 |
| macro avg | 0.65 | 0.63 | 0.62 | 610 |
| weighted avg | 0.65 | 0.64 | 0.63 | 610 |

## LDA training matrics

```
LDA_train_precision  0.68
LDA_train_recall  0.43
LDA_train_f1  0.53
```

## Confusion Matrix for Test Data(LDA)

Out[157]:
```
array([[113,  32],
       [ 65,  52]], dtype=int64)
```

## Test Data Accuracy(LDA)

Out[158]: 0.6297709923664122

## Classification Report of Test Data (LDA)

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.63 | 0.78 | 0.70 | 145 |
| 1 | 0.62 | 0.44 | 0.52 | 117 |
| accuracy | | | 0.63 | 262 |
| macro avg | 0.63 | 0.61 | 0.61 | 262 |
| weighted avg | 0.63 | 0.63 | 0.62 | 262 |

## LDA Test matrics

```
LDA_test_precision  0.62
LDA_test_recall  0.44
LDA_test_f1  0.52
```

## Train and Test Performance (LDA)

Out[161]:

| | LDA Train | LDA Test |
|---|---|---|
| Accuracy | 0.64 | 0.63 |
| AUC | 0.67 | 0.66 |
| Recall | 0.43 | 0.44 |
| Precision | 0.68 | 0.62 |
| F1 Score | 0.53 | 0.52 |

## 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.

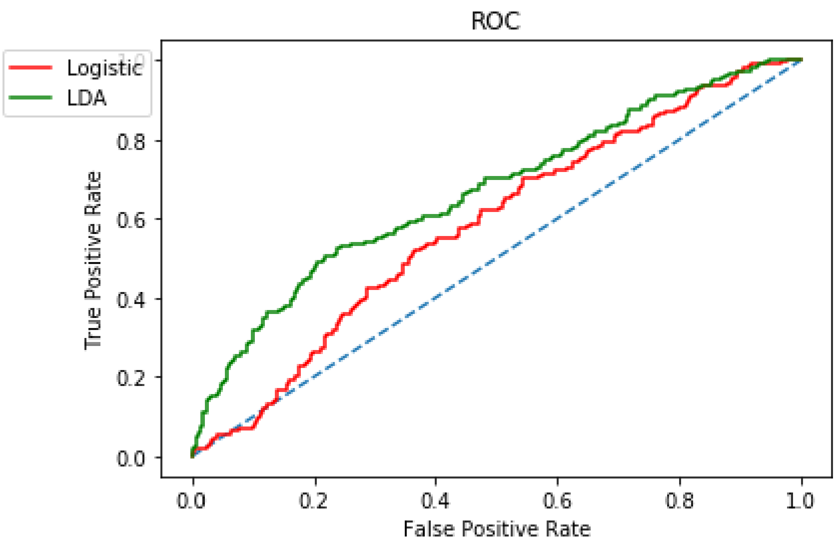
## Model comparision on the basis of Accuracy, AUC, Recall,Precision and F1 Score

Out[162]:

| | Logistic Train | Logistic Test | LDA Train | LDA Test |
|---|---|---|---|---|
| Accuracy | 0.52 | 0.55 | 0.64 | 0.63 |
| AUC | 0.58 | 0.62 | 0.67 | 0.66 |
| Recall | 0.06 | 0.10 | 0.43 | 0.44 |
| Precision | 0.42 | 0.50 | 0.68 | 0.62 |
| F1 Score | 0.11 | 0.17 | 0.53 | 0.52 |

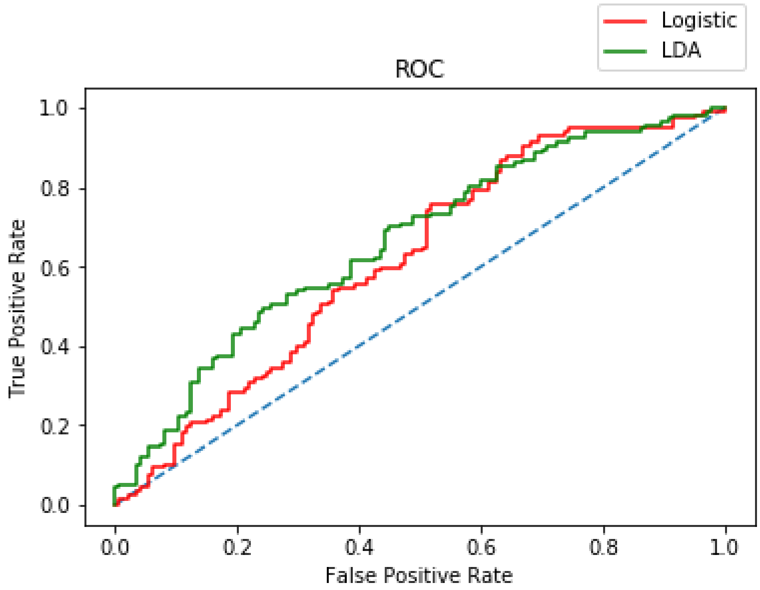
## ROC Curve for the models on the Training data

Out[163]: <matplotlib.legend.Legend at 0x18c176cdec8>




## ROC Curve for the models on the Test data

Out[164]: <matplotlib.legend.Legend at 0x18c168fbe88>




## Comparison comments

- Accuracy score of LDA is slightly better than that of Logical Regression
- AUC score of LDA is slightly better than that of Logical Regression
- Precision score of LDA is slightly better than that of Logical Regression
- F1 score of LDA is far better than that of Logical Regression
- There is significant difference in Recall Matric score of LDA and Logical Regression. LDA is better side
- Finally LDA is quite better Model when we compare with Logical Regression

## Stats Model for Holiday Package

## Concatenate X and y into a single dataframe

Out[166]:

| | Salary | age | educ | no_young_children | no_older_children | foreign | Holliday_Package |
|---|---|---|---|---|---|---|---|
| 502 | 34017.00 | 57.0 | 5.0 | 0.0 | 0.0 | 0 | 0 |
| 729 | 32197.00 | 22.0 | 6.0 | 0.0 | 0.0 | 1 | 1 |
| 604 | 80687.75 | 31.0 | 12.0 | 0.0 | 0.0 | 0 | 0 |
| 246 | 72394.00 | 50.0 | 14.0 | 0.0 | 1.0 | 0 | 0 |
| 494 | 28596.00 | 49.0 | 15.0 | 0.0 | 0.0 | 0 | 1 |

Out[167]:
```
Index(['Salary', 'age', 'educ', 'no_young_children', 'no_older_children',
       'foreign', 'Holliday_Package'],
      dtype='object')
```

## Coefficients of the variables

```
Out[169]: Intercept            4.834336e-01
          Salary              -4.135560e-06
          age                 -2.032565e-03
          educ                 1.470457e-02
          no_young_children    2.519695e-17
          no_older_children    4.232062e-02
          foreign              3.107410e-01
          dtype: float64
```

## Stats Summary of Holiday Package

```
                            OLS Regression Results
==============================================================================
Dep. Variable:        Holliday_Package   R-squared:                       0.096
Model:                             OLS   Adj. R-squared:                  0.088
Method:                  Least Squares   F-statistic:                     12.78
Date:                 Sun, 05 Jul 2020   Prob (F-statistic):           7.90e-12
Time:                         20:24:27   Log-Likelihood:                -410.60
No. Observations:                  610   AIC:                             833.2
Df Residuals:                      604   BIC:                             859.7
Df Model:                            5
Covariance Type:             nonrobust
=====================================================================================
                        coef    std err          t      P>|t|      [0.025      0.975]
-------------------------------------------------------------------------------------
Intercept             0.4834      0.131      3.682      0.000       0.226       0.741
Salary            -4.136e-06   1.37e-06     -3.017      0.003   -6.83e-06   -1.44e-06
age                  -0.0020      0.002     -1.049      0.295      -0.006       0.002
educ                  0.0147      0.008      1.896      0.058      -0.001       0.030
no_young_children   2.52e-17   6.44e-18      3.912      0.000    1.25e-17    3.78e-17
no_older_children     0.0423      0.018      2.299      0.022       0.006       0.078
foreign               0.3107      0.052      5.983      0.000       0.209       0.413
==============================================================================
Omnibus:                     3739.957   Durbin-Watson:                   1.938
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               67.605
Skew:                           0.156   Prob(JB):                     2.09e-15
Kurtosis:                       1.399   Cond. No.                     1.51e+21
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly speci
fied.
[2] The smallest eigenvalue is 6.29e-31. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.
```

## Final Equation

```
(0.48) * Intercept + (-0.0) * Salary + (-0.0) * age + (0.01) * educ + (0.0) * no_young
_children + (0.04) * no_older_children + (0.31) * foreign +
```

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

## Business Insights and Recommendations.

- Final Equation

- (0.48) * Intercept + (-0.0) * Salary + (-0.0) * age + (0.01) * educ + (0.0) * no_young_children + (0.04) * no_older_children + (0.31) * foreign +
- Important point :- foreign ,educ,Number of older children, Number of young children playing the key role in descending order in positive way and Salary is playing slightly Negative role.
- Should launch the discount scheme in the holiday Package if people having more then 2 number of older children.
- Should plan something of very young children in the holiday package like In some of malls children walker is given for young children during shoping in the malls
- there is more probability , foreigner will avail the Holiday Package. it is one most positve influencing factor
- Note :- pvalue of educ variable is slightly higher than .05 , I have considered it