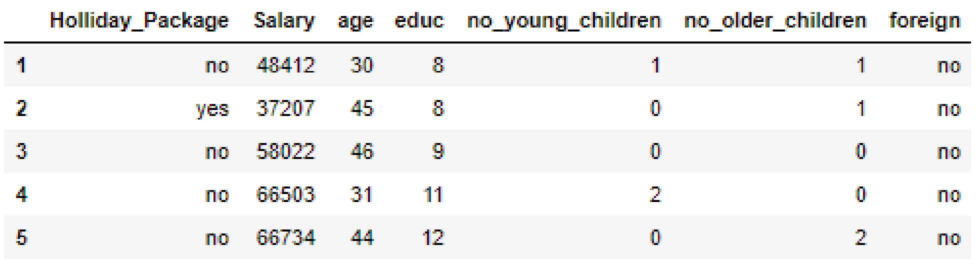
Problem 2: Logistic Regression and LDA

Problem Statement

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.

Let us import the data into a DataFrame and display the top 5 records –



	Holliday_Package	Salary	age	educ	no_young_children	no_older_children	foreign
1	no	48412	30	8	1	1	no
2	yes	37207	45	8	0	1	no
3	no	58022	46	9	0	0	no
4	no	66503	31	11	2	0	no
5	no	66734	44	12	0	2	no

Data Dictionary:

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

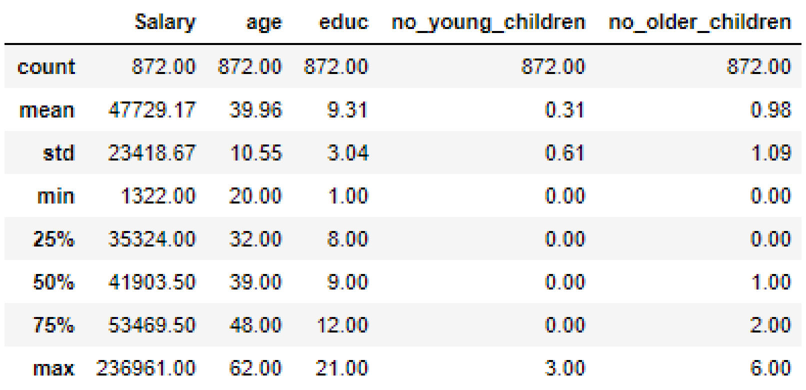
Now let us check the shape and datatypes of the variables –

```
Shape - (872, 7)
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 872 entries, 1 to 872
Data columns (total 7 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Holliday_Package   872 non-null    object
 1   Salary             872 non-null    int64
 2   age                872 non-null    int64
 3   educ               872 non-null    int64
 4   no_young_children  872 non-null    int64
 5   no_older_children  872 non-null    int64
 6   foreign            872 non-null    object
dtypes: int64(5), object(2)
memory usage: 54.5+ KB
```

The data consists of 7 columns (6 independent and 1 dependent variable). There are 2 categorical columns in the dataset. In total, there are 872 observations, with no null values.

Let us describe the dataset to get a statistical description of the data –



	Salary	age	educ	no_young_children	no_older_children
count	872.00	872.00	872.00	872.00	872.00
mean	47729.17	39.96	9.31	0.31	0.98
std	23418.67	10.55	3.04	0.61	1.09
min	1322.00	20.00	1.00	0.00	0.00
25%	35324.00	32.00	8.00	0.00	0.00
50%	41903.50	39.00	9.00	0.00	1.00
75%	53469.50	48.00	12.00	0.00	2.00
max	236961.00	62.00	21.00	3.00	6.00

Next let us also check for Null and Duplicate values –

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

dupes = df.duplicated()
print(sum(dupes))

0
```

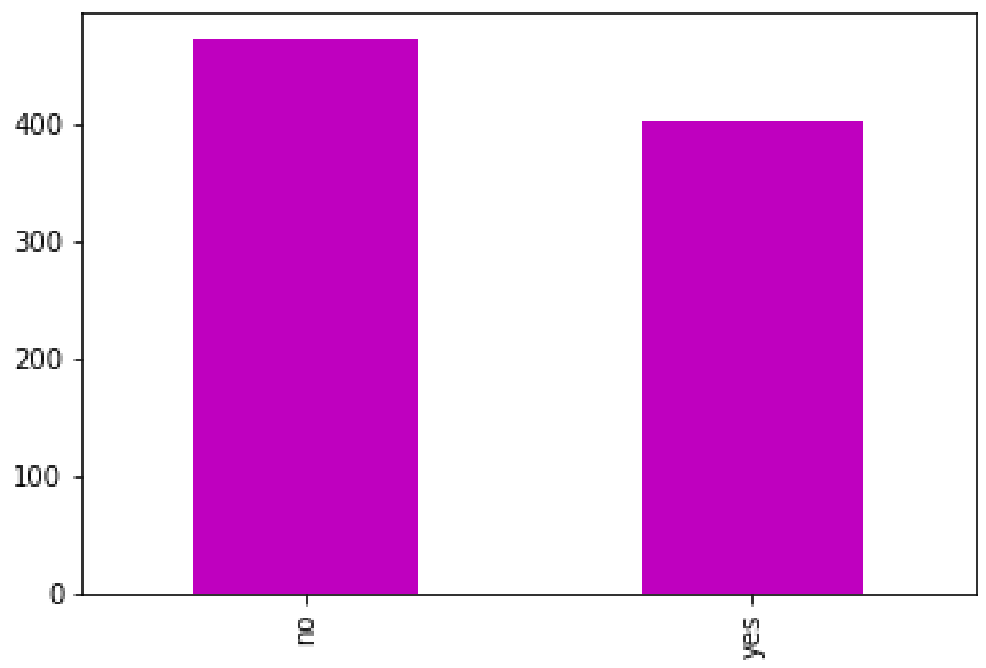
There are no Null or duplicate records in the dataset.

Now let's extract the numerical columns and save it in a separate list as it'll be easier for us to use this list for further transformations.

```
#list of numerical variables
num_vars = [var for var in df.columns if df[var].dtypes != 'O']

print('Number of numerical variables: ', len(num_vars))

#visualize the numerical variables
df[num_vars].head()

Number of numerical variables:  5
```

UNIVARIATE ANALYSIS –

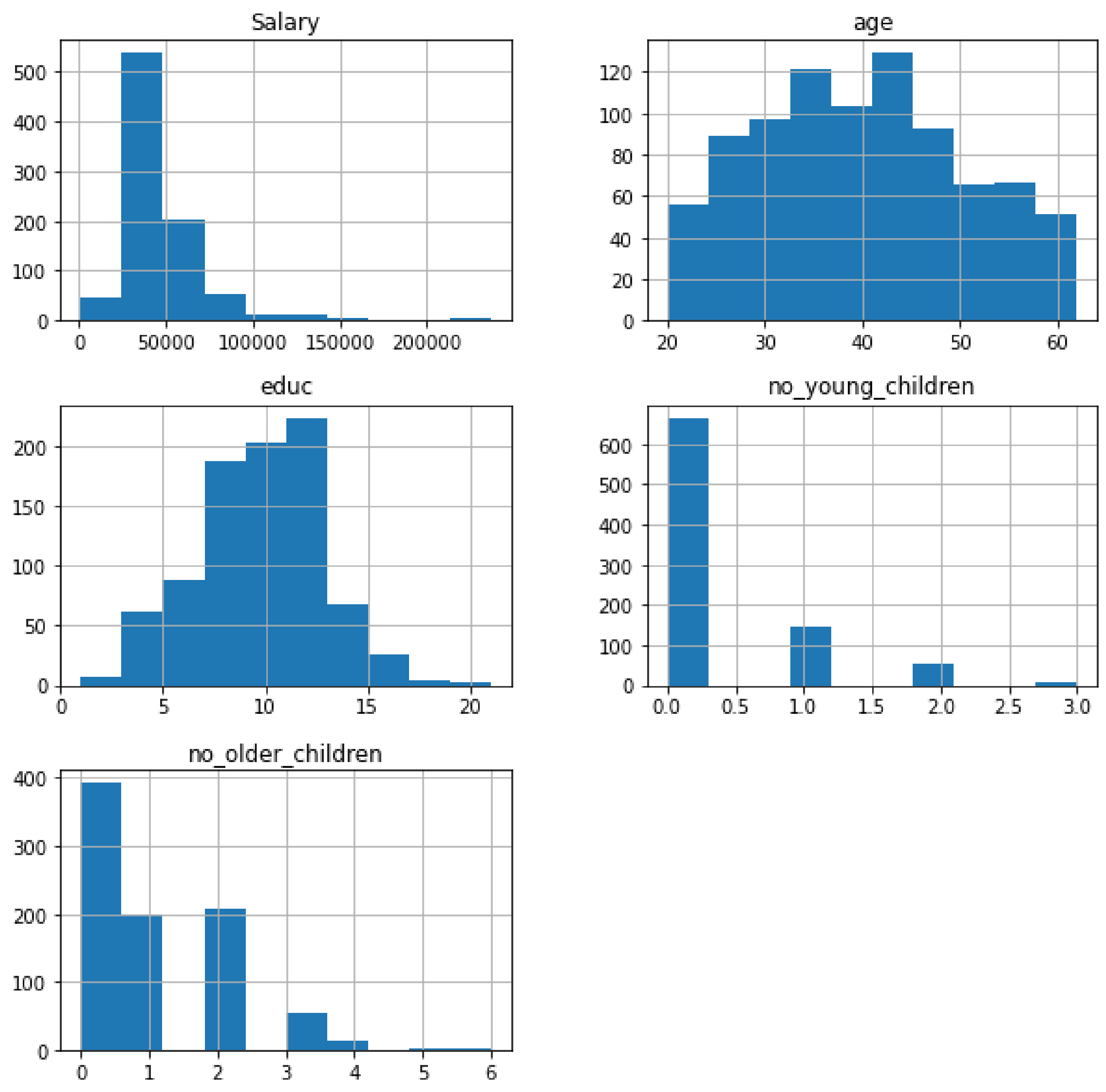
Next, let us inspect the distribution of the target variable ‘price’ –



```
no      54.01%
yes     45.99%
Name: Holiday_Package, dtype: float64
```

Around 46% employees have opted for the holiday package whereas 54% haven’t.

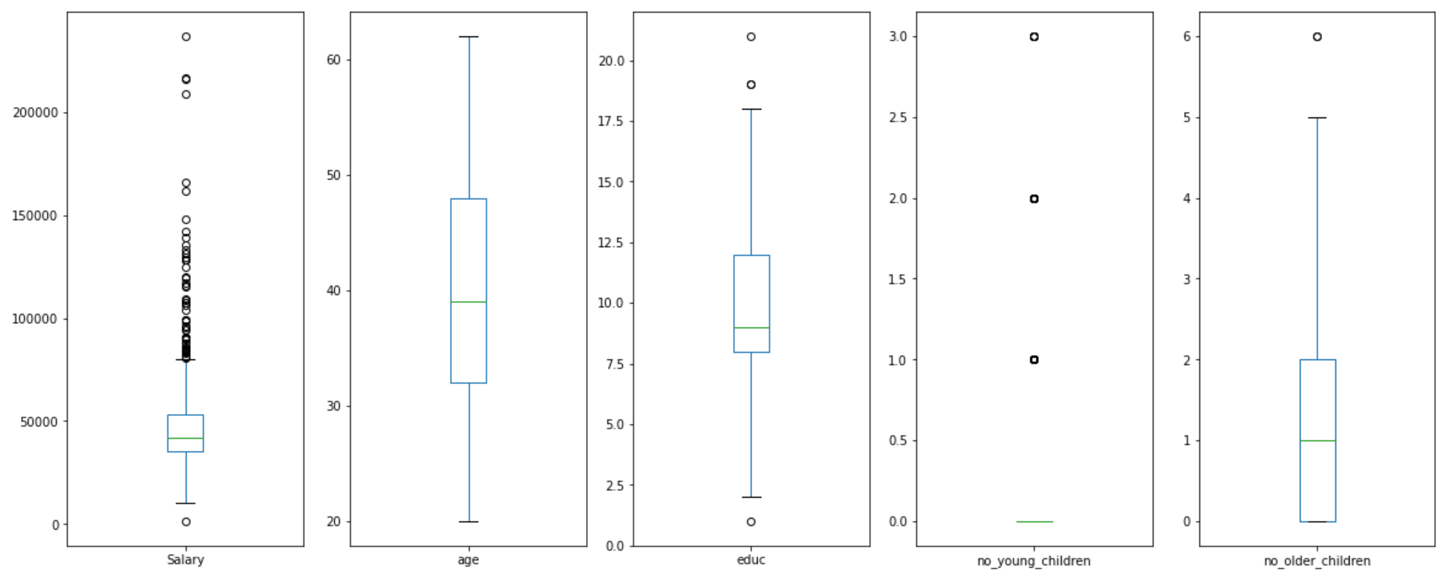
Let us also check the distribution of all numerical variables –



Observations –

 ‘Salary’ variable shows right-skewed distribution which denotes the presence of outliers. ‘age’ and ‘educ’ show almost normal distribution.

To add further clarity, lets plot boxplots for each of these variables to check the presence of outliers –



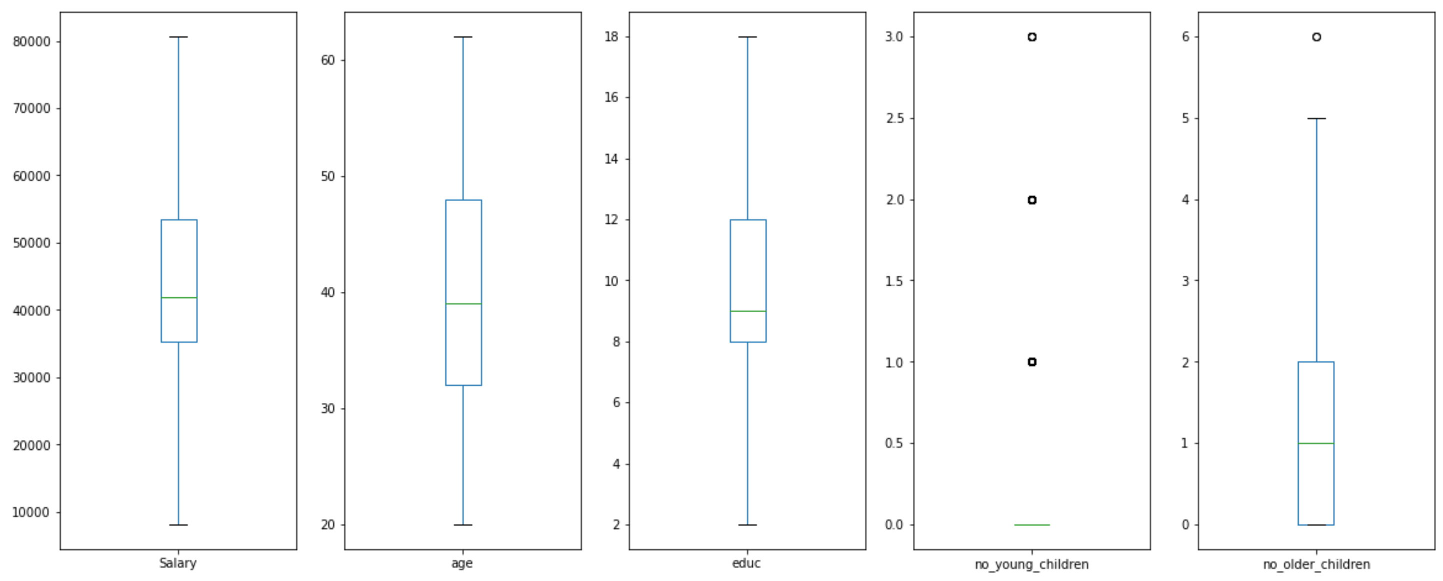
Salary shows the presence of outliers. Few outliers present for educ variable.

Let us also handle the outliers and create a separate DataFrame, so that we can try to fit the model with and without outliers to gauge the impact –

Please note we have omitted no_young_children and no_old_children from the outlier treatment as these columns would lose their value.

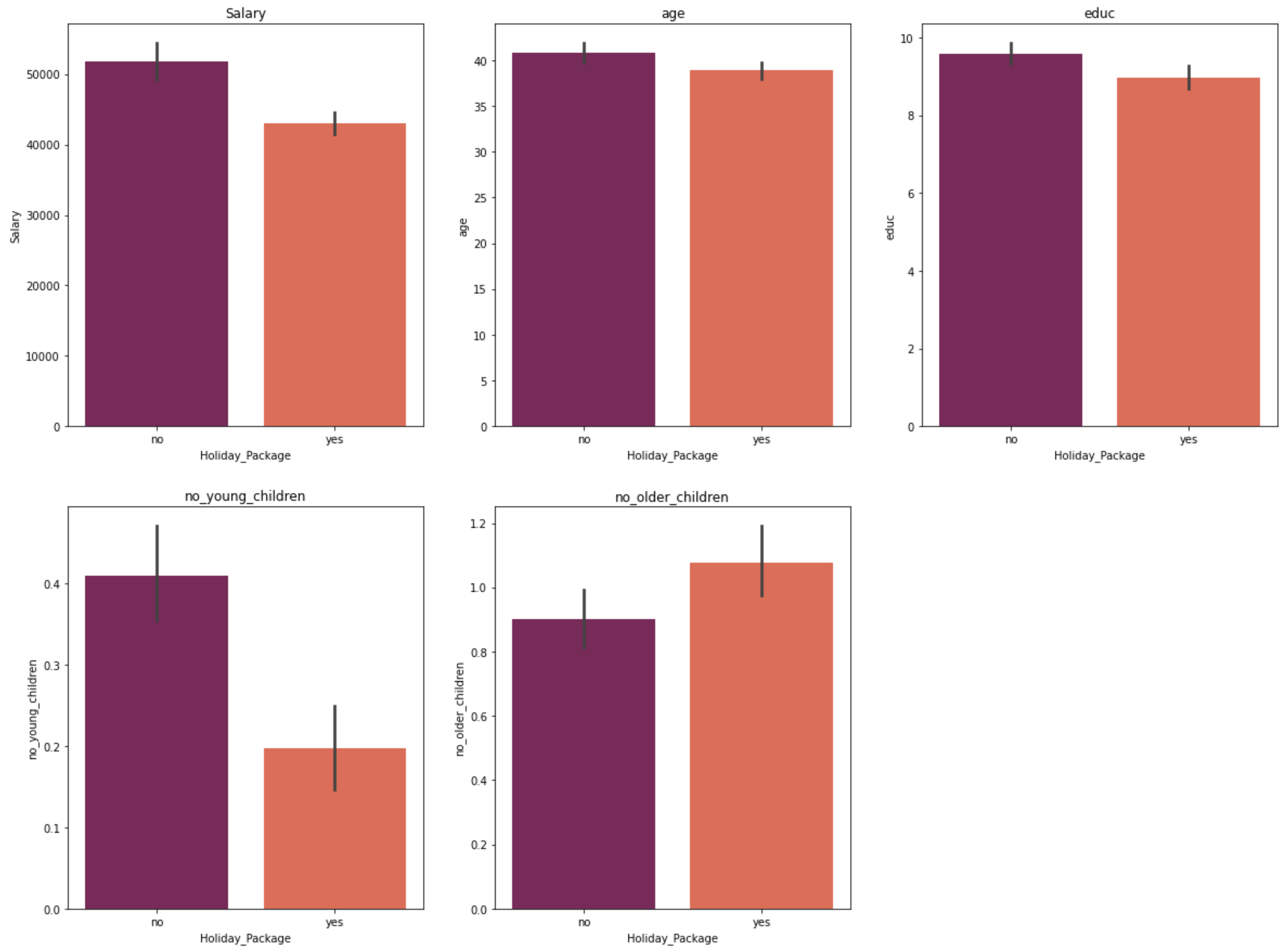
```
def find_skewed_boundaries(df, variable):
    #Let's calculate the boundary ranges for skewed distributions
    IQR = df[variable].quantile(0.75) - df[variable].quantile(0.25)

    lower_boundary = df[variable].quantile(0.25) - (IQR * 1.5)
    upper_boundary = df[variable].quantile(0.75) + (IQR * 1.5)

    return upper_boundary, lower_boundary

for var in num_vars:
    if var not in ['no_young_children', 'no_older_children']:
        upper_range, lower_range = find_skewed_boundaries(df_outlier, var)
        df_outlier[var] = np.where(df_outlier[var] > upper_range, upper_range, df_out
lier[var])
        df_outlier[var] = np.where(df_outlier[var] < lower_range, lower_range, df_out
lier[var])
```

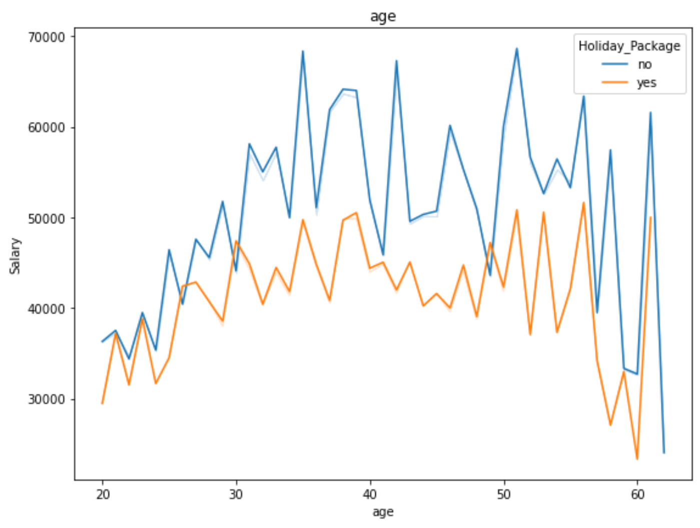
Viewing the data after treating the outliers –

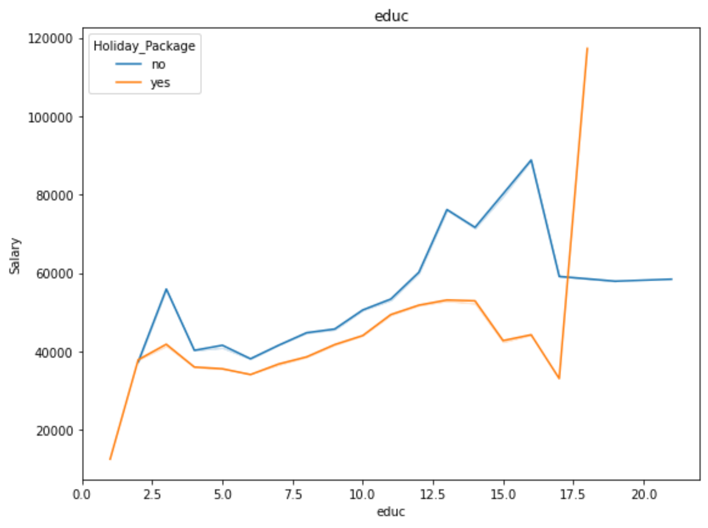


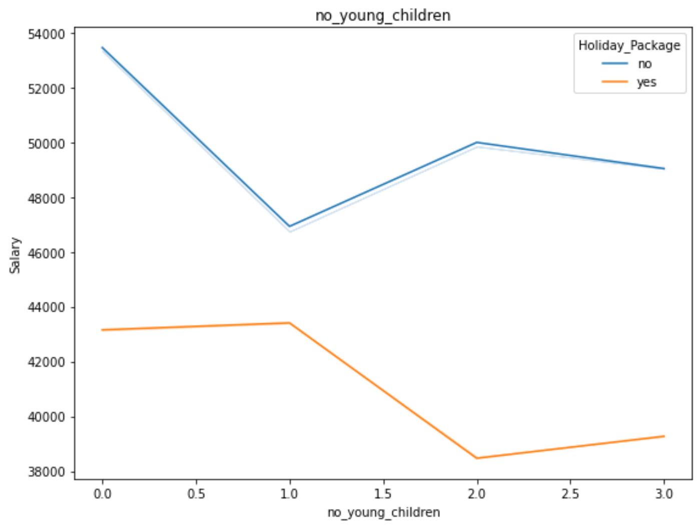
BIVARIATE ANALYSIS –

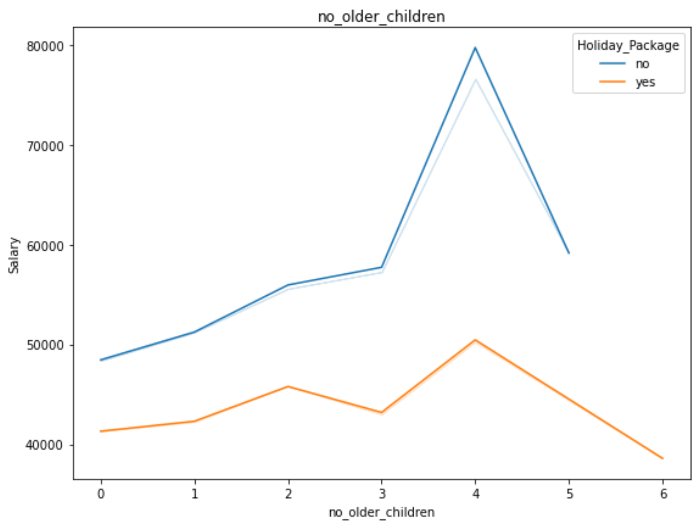
Let us now plot independent variables with the dependent to check the relationship between the data. We will use bar plots for doing this –









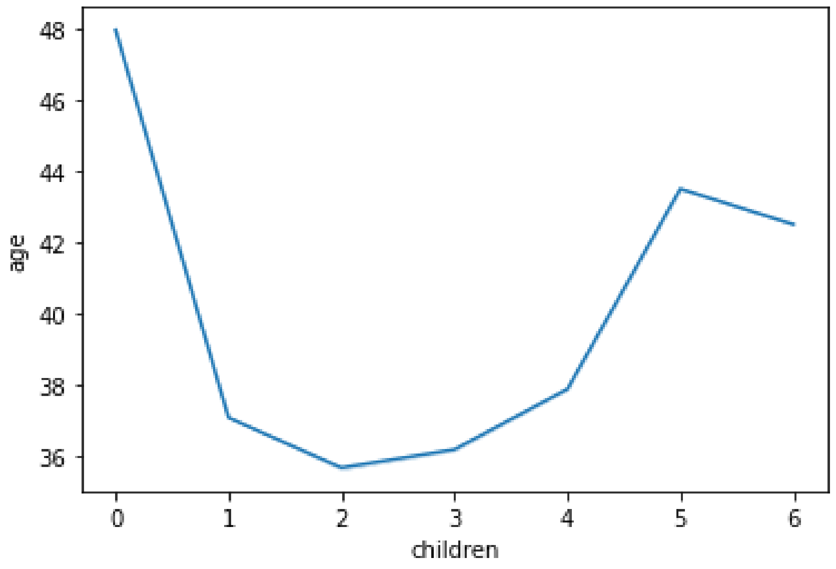


Observations –

From the plots, we can observe that employees with young children (younger than 7 years) have mostly not opted for the holiday package. There isn't much variation in the age of employees who have and have not opted for the package.

As expected, we see an increasing trend in Salary with total number of years of Education. Employees with 17 to 18 years of education seem to have opted for the package.

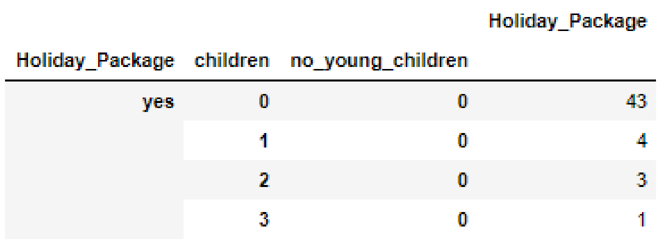
As we are not able to get more specific insights from these plots, lets do some more EDA by creating an extra column by adding the no of young and older children.



Next, we will try to divide the DataFrame in to temp DataFrames –

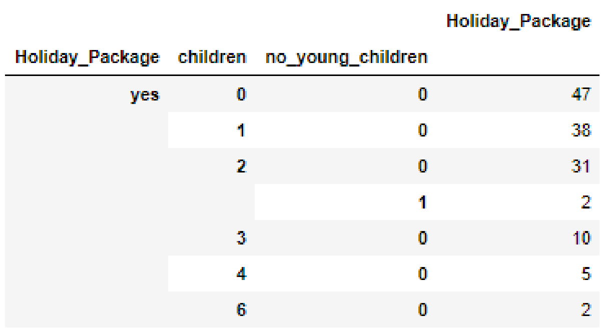
- Employees in their Fifties (Age 50 and plus)
- In their Forties (Age between 40 and 49)
- In their Thirties (Age between 30 and 39)
- In their Twenties (Age less than 30)
-

In Fifties



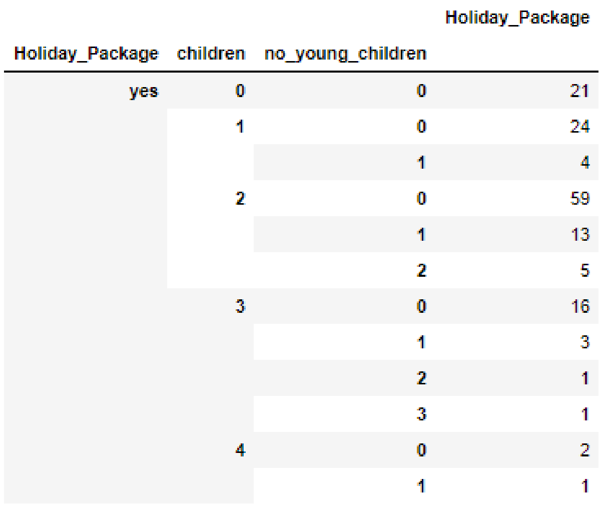
			Holiday_Package
Holiday_Package	children	no_young_children	
yes	0	0	43
	1	0	4
	2	0	3
	3	0	1

In Forties



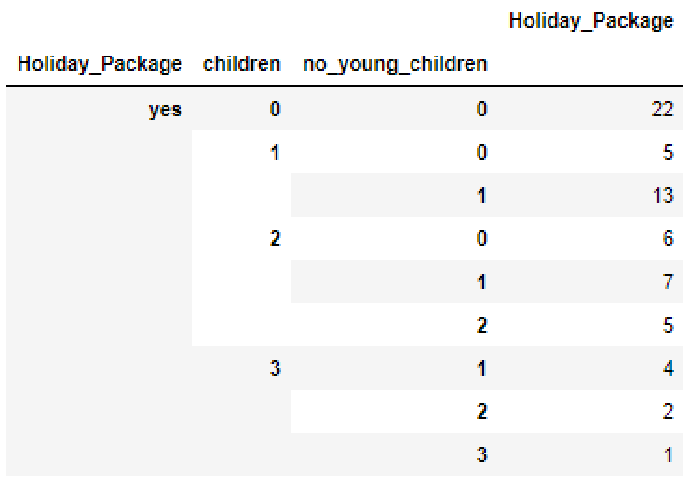
			Holiday_Package
Holiday_Package	children	no_young_children	
yes	0	0	47
	1	0	38
	2	0	31
		1	2
	3	0	10
	4	0	5
	6	0	2

In Thirties

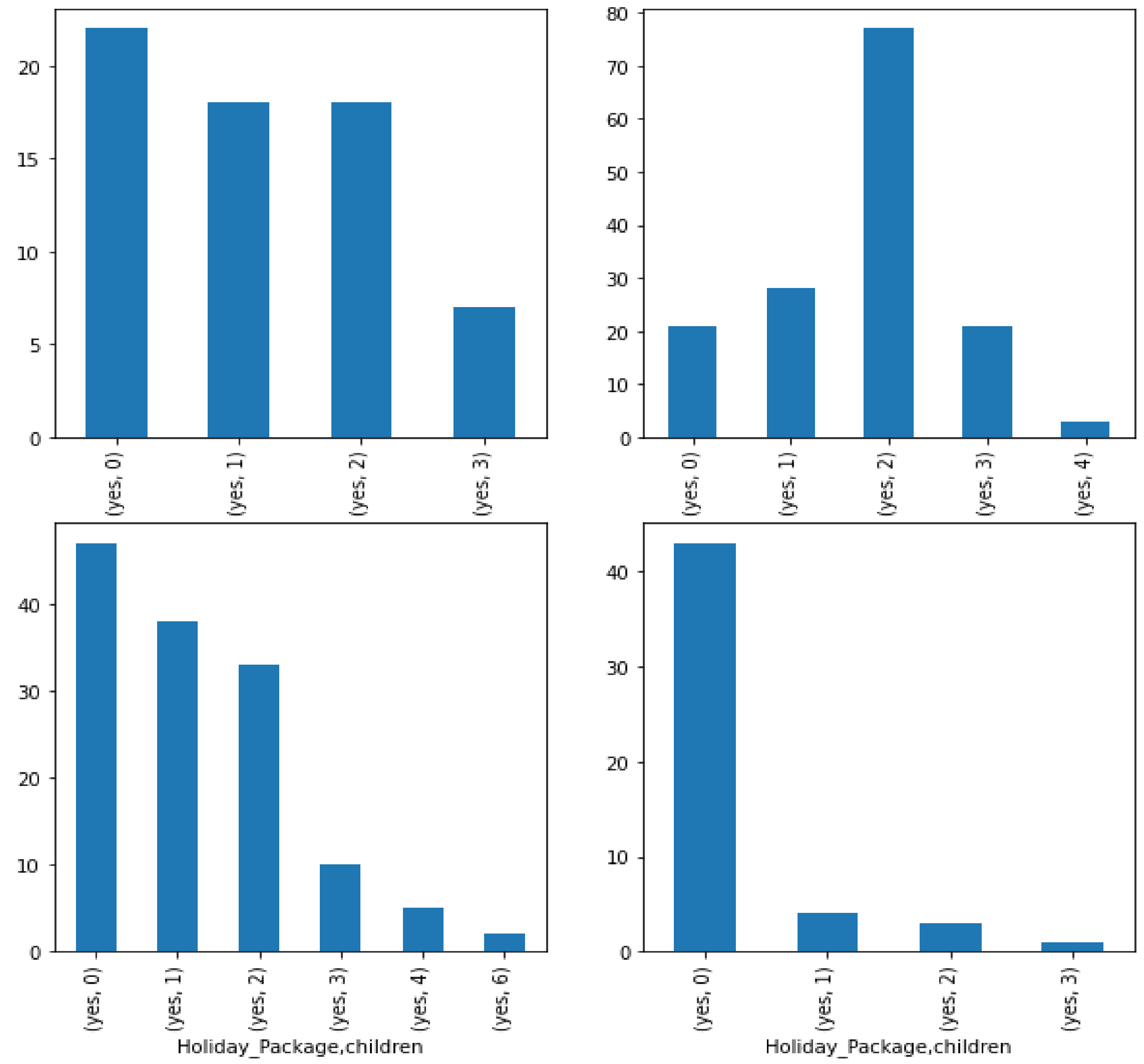


			Holiday_Package
Holiday_Package	children	no_young_children	
yes	0	0	21
	1	0	24
		1	4
	2	0	59
		1	13
		2	5
	3	0	16
		1	3
		2	1
		3	1
	4	0	2
		1	1

In Twenties



			Holiday_Package
Holiday_Package	children	no_young_children	
yes	0	0	22
	1	0	5
		1	13
	2	0	6
		1	7
		2	5
	3	1	4
		2	2
		3	1

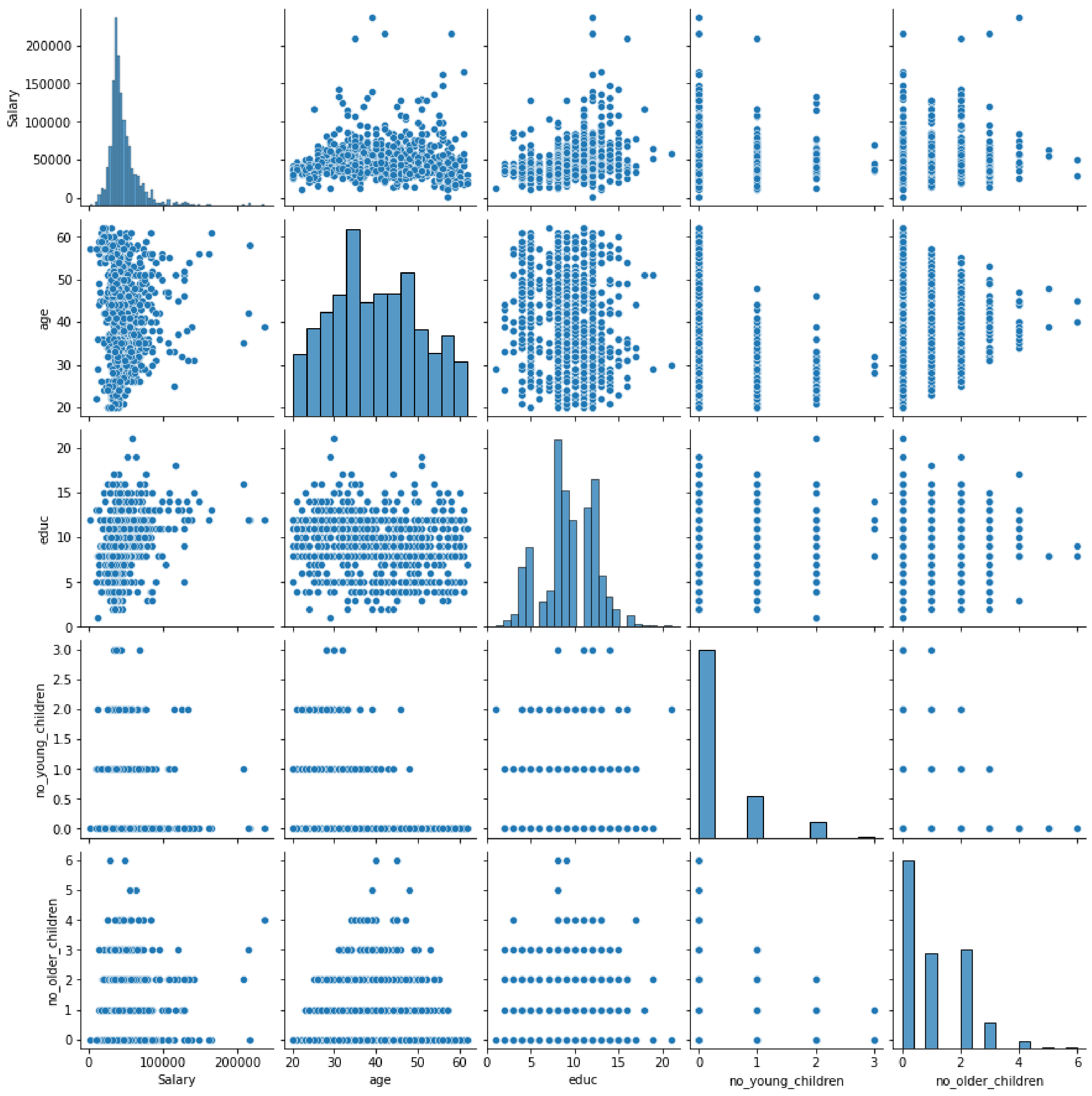


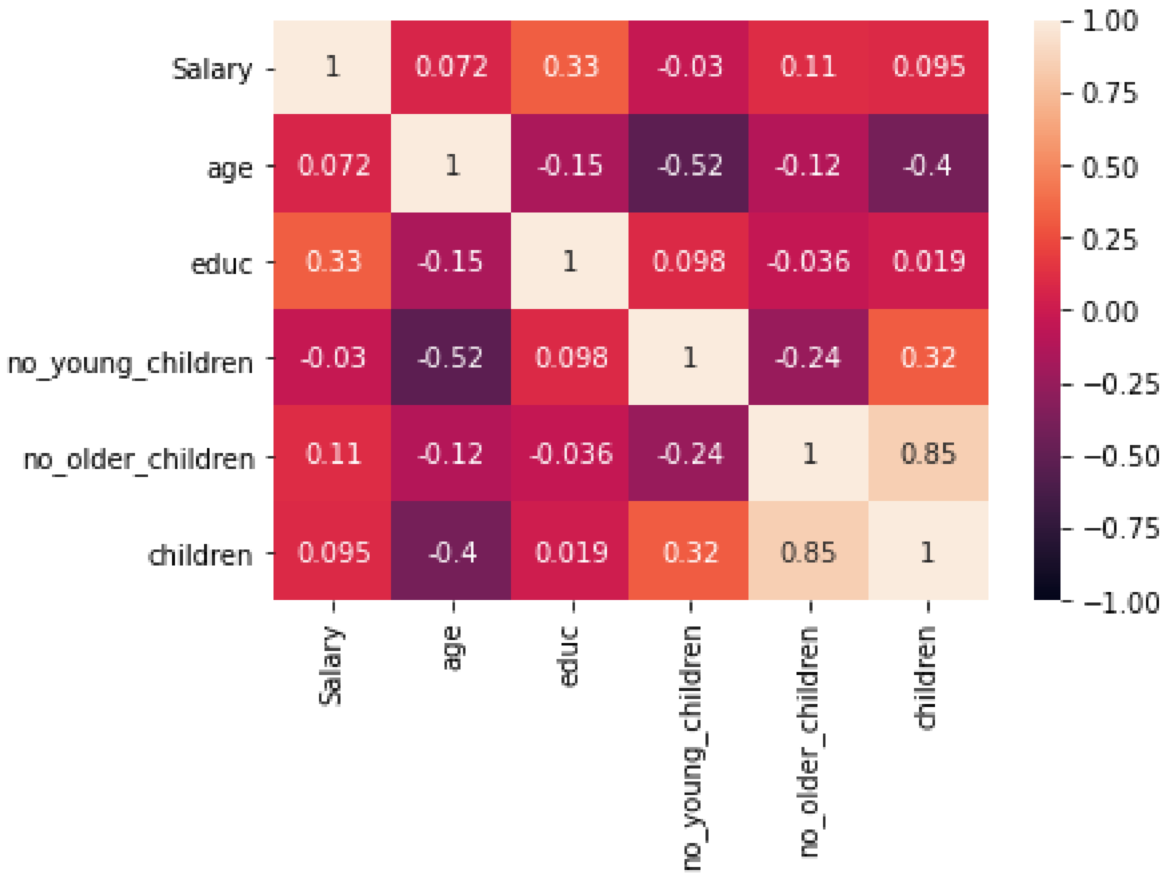
Some interesting observations –

1. Only 27% of the employees who are in their Fifties have opted for the Holiday Package. Of these, around 84% have 0 children. Remaining are the ones with older children but no young children.
2. 53% of employees in their forties seem to have opted for the holiday package. 62% from these have either 0 or 1 child, but no young children. This is also an only group with 6 children. Both who have opted for the package are not foreigners.
3. 55% of employees in their thirties have opted for the holiday package. Of these, 51% have 2 children and 23% from these who seem to have opted for the holiday package have 1 or 2 young children.
4. Around 40% of the employees in their twenties have opted for the holiday package. 66% from these have either 1, 2 or 3 children and mostly young ones.

In all, it appears like mostly the employees who in their thirties and forties have opted for the package. The least adoption is from the upper age group (employees who are 50 years and above). Younger employees in their twenties seem to be a potential group that can be targeted to increase sales.

CORRELATION BETWEEN THE VARIABLES USING PAIRPLOT AND HEATMAP –





Observations –

1. There doesn’t seem to be very strong relationship between the variables
2. As one would expect, there is a negative relationship between age 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).

We will drop the no_older_children column from the DataFrame as it is now redundant.

Encoding

Since there is no inherent order in the categorical columns, we will go with One Hot encoding. We have used get_dummies function from Pandas which will convert categorical variable into dummy/indicator variables. We will also keep drop_first as True so that it will return k-1 dummies.

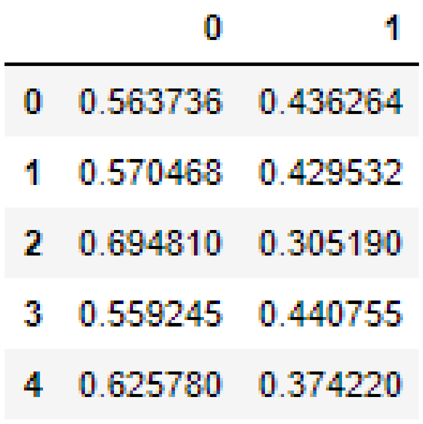
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 872 entries, 1 to 872
Data columns (total 7 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   Salary              872 non-null    int64
 1   age                 872 non-null    int64
 2   educ                872 non-null    int64
 3   no_young_children   872 non-null    int64
 4   children            872 non-null    int64
 5   Holiday_Package_yes  872 non-null    uint8
 6   foreign_yes          872 non-null    uint8
dtypes: int64(5), uint8(2)
memory usage: 82.6 KB
```

There are no categorical columns in the dataset now!

Let’s proceed and create a Training and Test Data split. Then we will create a Logistic Regression Model and try to fit it on DataFrame to see the impact –

```
#Copy all the predictor variables into X dataframe
X = df_encoded.drop('Holiday_Package_yes', axis=1)

#Copy target into the y dataframe.
y = df_encoded['Holiday_Package_yes']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state
=0, stratify=y)

log_model = LogisticRegression()
log_model.fit(X_train, y_train)

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

Let’s look at the probability estimates –



	0	1
0	0.563736	0.436264
1	0.570468	0.429532
2	0.694810	0.305190
3	0.559245	0.440755
4	0.625780	0.374220

Predict y_train and y_test –

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

Now let’s build another model using LDA (Linear Discriminant Analysis) and check –

#Build LDA Model and fit the data

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

Let’s look at the probability estimates of this model –

	0	1
0	0.563902	0.436098
1	0.419695	0.580305
2	0.870244	0.129756
3	0.281443	0.718557
4	0.438711	0.561289

Predict y_train and y_test –

```
ytrain_pred_lda = lda_model.predict(X_train)
ytest_pred_lda = lda_model.predict(X_test)
```

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.

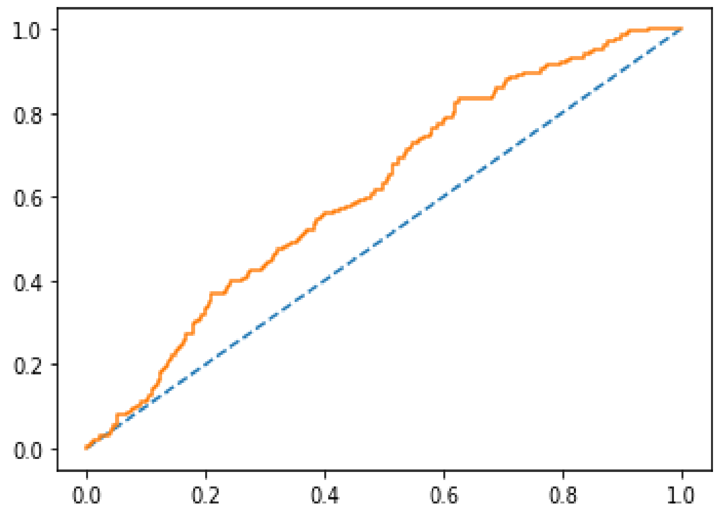
Let's check the Accuracy, ROC_AUC, Confusion matrix and Classification report for **LogisticRegression** model –

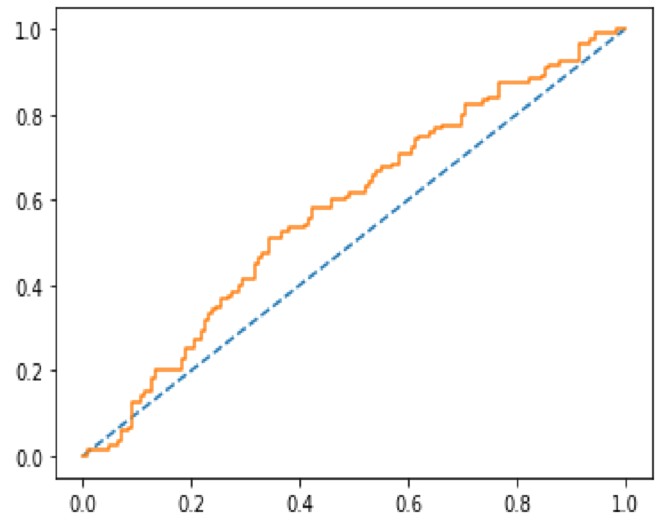
ACCURACY FOR TRAINING AND TEST DATA

```
log_model.score(X_train, y_train)
0.5393442622950819

log_model.score(X_test, y_test)
0.5419847328244275
```

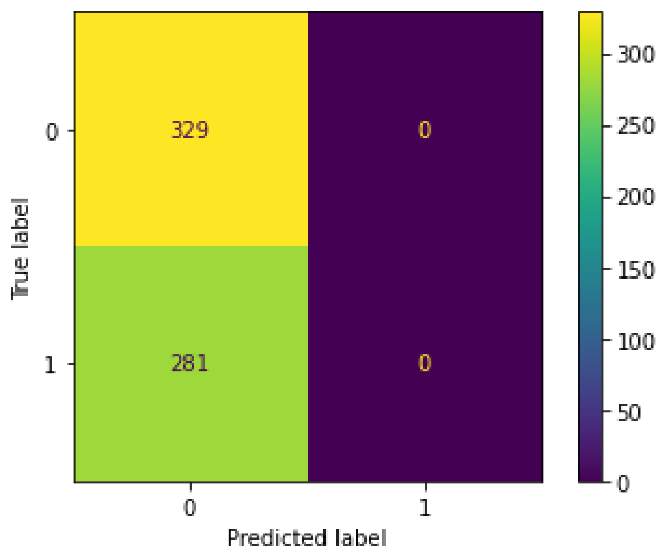
AUC SCORE AND ROC AUC CURVE FOR TRAINING AND TEST DATA

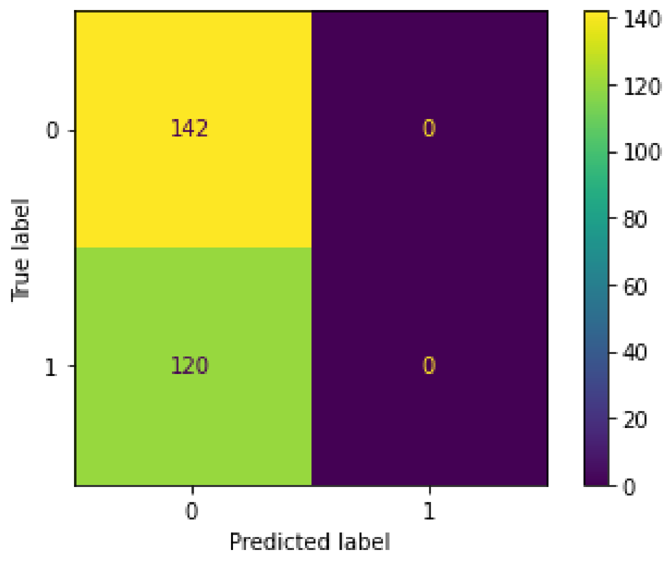
```
AUC: 0.614 AUC: 0.578
```





CONFUSION MATRIX FOR TRAINING AND TEST DATA





CLASSIFICATION REPORT FOR TRAINING AND TEST DATA

```
            precision    recall  f1-score   support

           0       0.54      1.00      0.70       329
           1       0.00      0.00      0.00       281

    accuracy                           0.54       610
   macro avg       0.27      0.50      0.35       610
weighted avg       0.29      0.54      0.38       610


             precision    recall  f1-score   support

           0       0.54      1.00      0.70       142
           1       0.00      0.00      0.00       120

    accuracy                           0.54       262
   macro avg       0.27      0.50      0.35       262
weighted avg       0.29      0.54      0.38       262
```

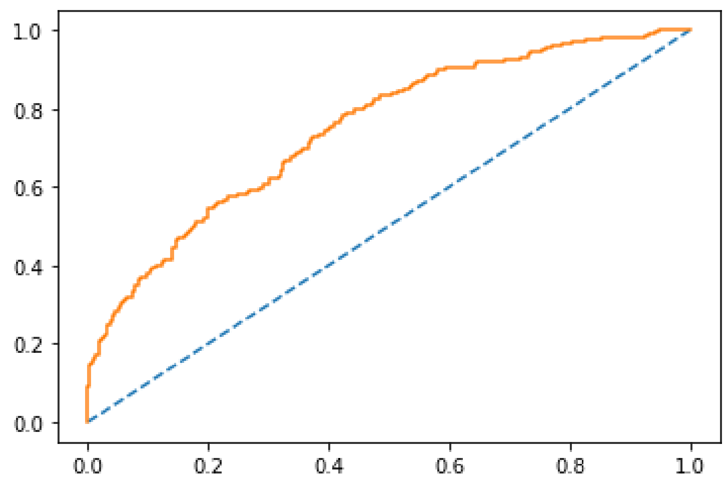
From the metrics, model performance looks pretty bad as recall score is 0. This model has been applied with it’s default parameters. Maybe we can perform GridSearchCV and tune the hyperparameters. But first lets see the accuracy of our **LDA model**.

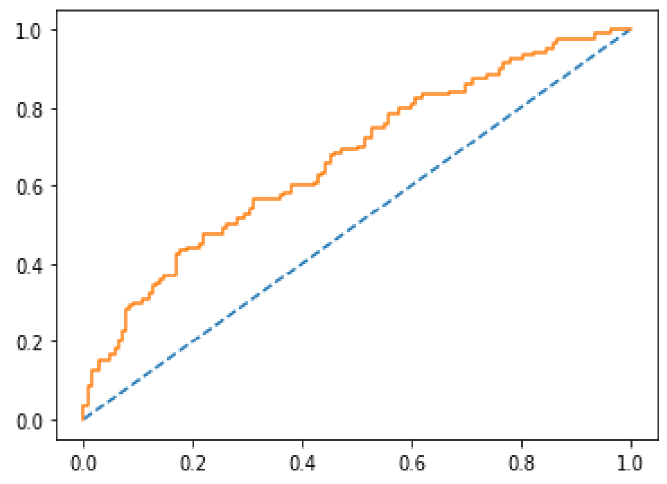
ACCURACY FOR TRAINING AND TEST DATA

```
lda_model.score(X_train, y_train)
0.6786885245901639

lda_model.score(X_test, y_test)
0.6297709923664122
```

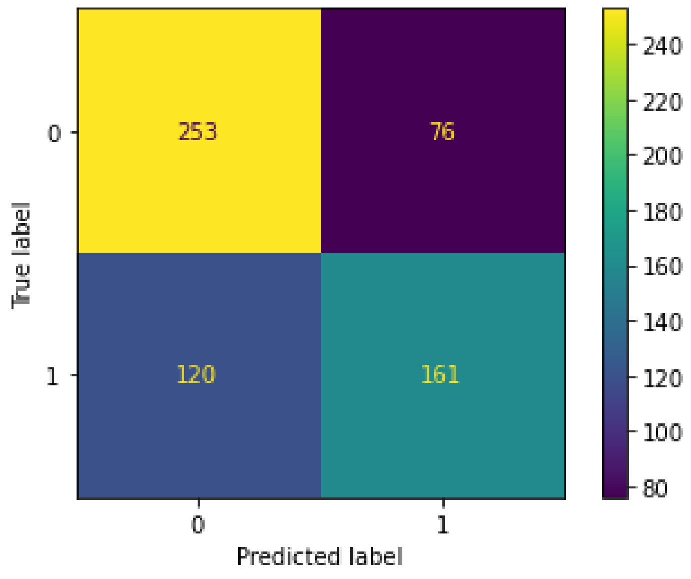
AUC SCORE AND ROC AUC CURVE FOR TRAINING AND TEST DATA

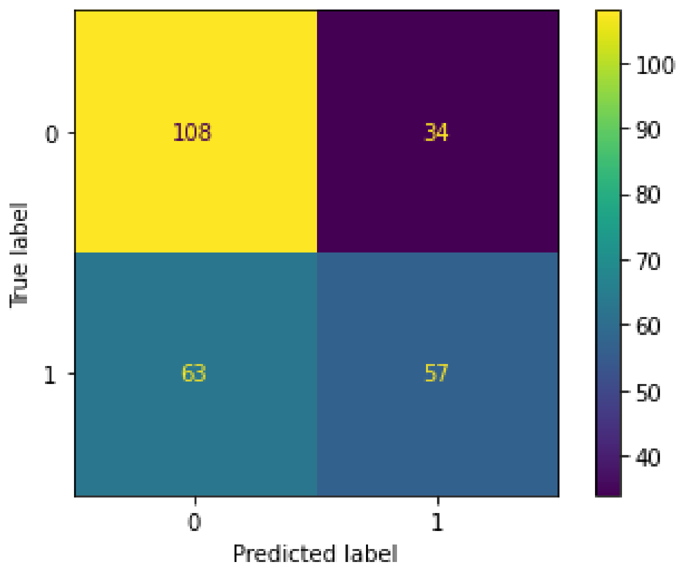
```
AUC: 0.748 AUC: 0.668
```





CONFUSION MATRIX FOR TRAINING AND TEST DATA





CLASSIFICATION REPORT FOR TRAINING AND TEST DATA

```
              precision    recall  f1-score   support

           0       0.68      0.77      0.72       329
           1       0.68      0.57      0.62       281

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

```
          precision    recall  f1-score   support

           0       0.63      0.76      0.69       142
           1       0.63      0.47      0.54       120

    accuracy                           0.63       262
   macro avg       0.63      0.62      0.62       262
weighted avg       0.63      0.63      0.62       262
```

Observations –

There is definitely an improvement in Accuracy and Recall scores with LDA model.

A confusion matrix is a summary of prediction results. From the confusion matrix of the test data, the result is being telling us that we have 165 (57+108) correct predictions and 97 (63+34) incorrect predictions.

Recall = (TP) / (TP+FN)

where, TP is the number of true positives, and FP is the number of false positives.

On the Testing dataset, the model is correctly identifying 76% in class 0s and only 47% in class 1s.

Next let's also see the effect of outlier treatment on the model performance. We will use the separate DataFrame that we had created earlier with outliers treated.

```
X_train_out, X_test_out, y_train_out, y_test_out = train_test_split(X_out, y_out, tes
t_size=0.3, random_state=0, stratify=y)

#LogisticRegression
log_model.fit(X_train_out, y_train_out)

#LDA
lda_model.fit(X_train_out, y_train_out)
```

ACCURACY FOR TRAINING AND TEST DATA FOR LOGISTIC REGRESSION

```
log_model.score(X_train, y_train)
0.5393442622950819

log_model.score(X_test, y_test)
0.5419847328244275
```

ACCURACY FOR TRAINING AND TEST DATA FOR LDA

```
lda_model.score(X_train, y_train)
0.6770491803278689

lda_model.score(X_test, y_test)
0.6335877862595419
```

Observations – We see there is no major difference in any of the models' performance after outlier treatment.

As a final step, lets try to tune the hyper parameters of Logistic Regression model using **GridSearchCV** –

```
grid={'penalty':['l1', 'l2', 'elasticnet', 'none'],
      'solver':['sag','lbfgs','newton-cg','liblinear'],
      'C':[1, 5, 10],
      'max_iter':[10000,100000],
      'tol':[0.001, 0.0001,0.00001]}ridge = Ridge(alpha=0.001, normalize=True)


model = LogisticRegression()


grid_search = GridSearchCV(estimator = model, param_grid = grid, cv=10,n_jobs=-1)
```

```
grid_search.fit(X_train, y_train)
```

Print the best parameters –

```
{'C': 5, 'max_iter': 10000, 'penalty': 'l2', 'solver': 'liblinear', 'tol': 1e-05}

LogisticRegression(C=5, max_iter=10000, solver='liblinear', tol=1e-05)
```

Let's create a model with these params –

```
best_model = grid_search.best_estimator_
best_model


LogisticRegression(C=5, max_iter=10000, solver='liblinear', tol=1e-05)
```

#Prediction on the training set

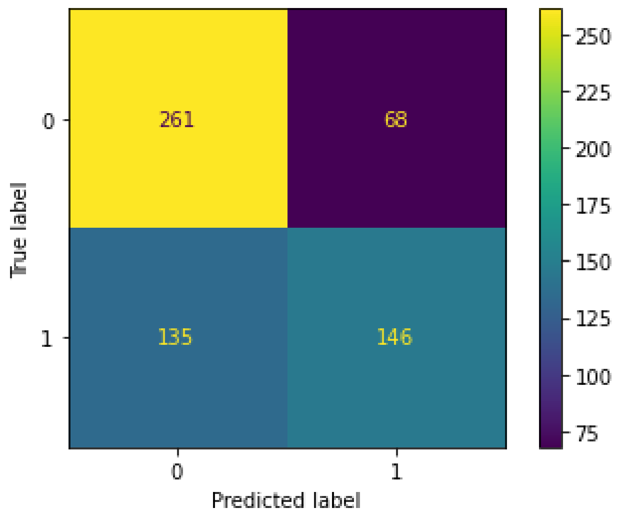
```
ytrain_predict = best_model.predict(X_train)
ytest_predict = best_model.predict(X_test)lasso = Lasso(alpha=0.08, normalize=True)
lasso.fit(X_train,y_train)
```

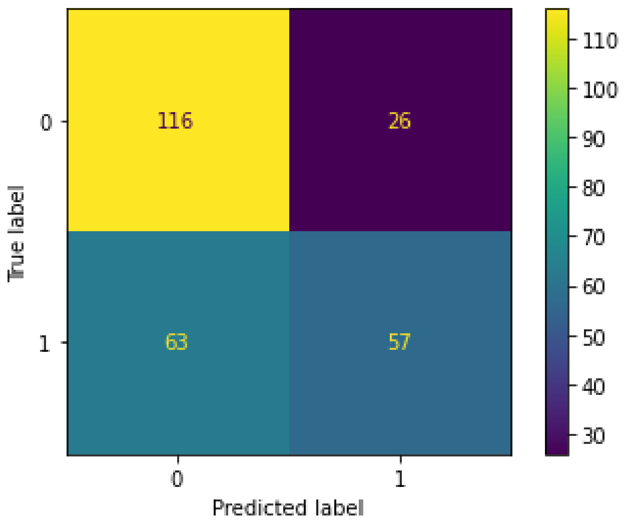
Get model scores –

ACCURACY FOR TRAINING AND TEST DATA

```
best_model.score(X_train, y_train)
0.6672131147540984

best_model.score(X_test, y_test)
0.6603053435114504
```

CONFUSION MATRIX FOR TRAINING AND TEST DATA





CLASSIFICATION REPORT FOR TRAINING AND TEST DATA

```
              precision    recall  f1-score   support

           0       0.66      0.79      0.72       329
           1       0.68      0.52      0.59       281

    accuracy                           0.67       610
   macro avg       0.67      0.66      0.65       610
weighted avg       0.67      0.67      0.66       610


           precision    recall  f1-score   support

           0       0.65      0.82      0.72       142
           1       0.69      0.47      0.56       120

    accuracy                           0.66       262
   macro avg       0.67      0.65      0.64       262
weighted avg       0.67      0.66      0.65       262
```

We see that now the model performance has improved after tuning the hyperparameters and is similar to the results we achieved with LDA.

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

INFERENCES –

From the EDA performed we can recommend the travel agency to probably target the employees in the following age clusters –

Employees with age 50 and above –

Only 27% of the employees who are in this this age group have opted for the Holiday Package. Of these, around 84% have 0 children. The agency can try to increase Sales in this age group by providing some senior citizen discount or even aid while travelling. Packages which are more age-friendly.

Employees between 30 to 50 age group –

Around 50% of the employees in this age group have opted for the holiday package and this is the age group that also seem to travel with children, so increase sale the agency can propose more child-friendly programs. Also, more payment schemes or EMI options.

Employees in twenties age group –

Around 40% of the employees in their twenties have opted for the holiday package. 66% from these have either 1, 2 or 3 children and mostly young ones. So, to target this potential agency group the agency can provide some schemes that are friendly or can provide support to young children likes nanny services etc. Payment options, EMI-schemes can also be explored here.

We have also seen that Outlier treatment has no effect on model performance, so it may be avoided in Production.

Overall, both the models can give us around 60% accuracy. So, we can try out other classification models like RandomForest or ANN and compare the model performance.