Flight Price Prediction Project

Submitted by:

KARISHMA YADAV.

ACKNOWLEDGMENT

I have taken efforts in this project. To solve this problem, the log transformation on the target variable is applied when it has skewed distribution and we need to apply an inverse function on the predicted values to get the actual predicted target value.

Due to this, for evaluating the model, the *RMSLE* is calculated to check the error and the *R2 Score* is also calculated to evaluate the accuracy of the model.

INTRODUCTION

Airline companies use complex algorithms to calculate flight prices given various conditions present at that particular time. These methods take financial, marketing, and various social factors into account to predict flight prices.

Nowadays, the number of people using flights has increased significantly. It is difficult for airlines to maintain prices since prices change dynamically due to different conditions. That's why we will try to use machine learning to solve this problem. This can help airlines by predicting what prices they can maintain. It can also help customers to predict future flight prices and plan their journey accordingly.

1) Regression analysis: Regression analysis techniques aim mainly to investigate and estimate the relationships among a set of features. Regression includes many models for analysing the relation between one target/response variable and a set of independent variables. Logistic Regression (LR) is the appropriate regression analysis model to use when the dependent variable is binary. LR is a predictive analysis used to explain the relationship between a dependent binary variable and a set of independent variables.

2) Decision Tree: Decision Tree (DT) is a model that generates a tree-like structure that represents set of decisions. DT returns the

probability scores of class membership. DT is composed of: a) internal Nodes: each node refers to a single variable/feature and represents a test point at feature level; b) branches, which represent the outcome of the test and are represented by lines that finally lead to c) leaf Nodes which represent the class labels. That is how decision rules are established and used to classify new instances. DT is a flexible model that supports both categorical and continuous data. Due to their flexibility they gained popularity and became one of the most commonly used models for prediction.

3) Random Forest Random forests (RF) are an ensemble learning technique that can support classification and regression. It extends the basic idea of single classification tree by growing many classification trees in the training phase. To classify an instance, each tree in the forest generates its response (vote for a class), the model choses the class that has receive the most votes over all the trees in the forest. One major advantage of RF over traditional decision trees is the protection against over fitting which makes the model able to deliver a high performance.

4) KNN

K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique. K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories. K-NN algorithm stores all the available data and classifies a new data point based on

the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm. K-NN algorithm can be used for Regression as well as

for Classification but mostly it is used for the Classification problems. K-NN is a **non-parametric algorithm**, which means it does not make any assumption on underlying data.

- R^2 Score: It is a statistical measure of how close the data are to the fitted regression line. It is also known as the coefficient of determination, or the coefficient of multiple determination for multiple regression. 0% indicates that the model explains none of the variability of the response data around its mean.

The first step before applying the selected analytical models on the dataset, explanatory data analysis for more insights into dataset was performed. Based on the observations data was pre-processed to be more suitable for analysis.

Analytical Problem Framing

Data cleaning :

As a next step, check missing values for each feature.

Missing values before data cleaning process :

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

```
Unnamed: 0         0
AirlineName        0
Arrival Time       0
Duration           0
Price              0
Total Stops        0
Destination        0
Route              0
Departure Time     0
dtype: int64
```

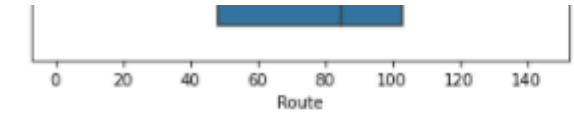
There is no null values present in this dataset.

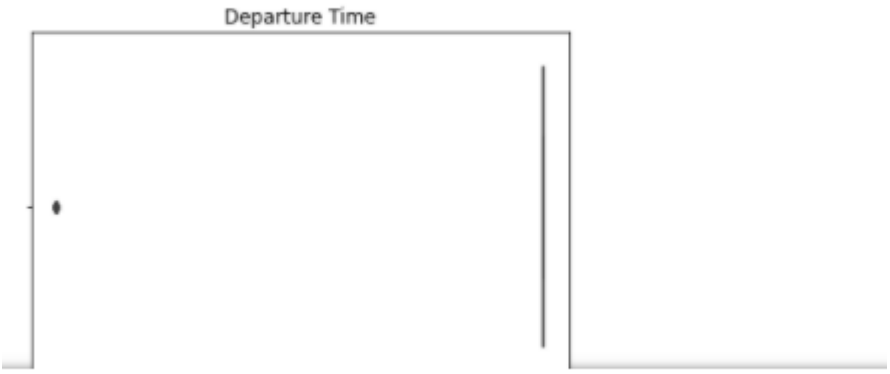
```
# example of univariate analysis
plt.figure(figsize=[16,6])
sns.heatmap(df.isnull())
plt.title("null values")
plt.show()
```

In this plot there is no null values are present.

This plot showing there is no null values are present.

Outliers: zscore method is used to remove the outliers from the data.





Duration and Departure Time this two columns has outliers are present.

Data preprocessing:

Label Encoder: In our dataset, 7 features are categorical variables and To apply the ML models, we need to transform these categorical variables into numerical variables. And sklearn library *LabelEncoder* is used to solve this problem.

Normalization: The dataset is not normally distributed. All the features have different ranges. Without normalization, the ML model will try to disregard coefficients of features that have low values because their impact will be so small compared to the big value. Hence to normalized, *sklearn library i.e. standard Scaler* is used.

Train the data. In this process, 90% of the data was split for the train data and 10% of the data was taken as test data.

Hardware and Software Requirements and Tools Used

RAM: 8GB

ROM: I3 processor

SOFTWARE: Python 3.9.6

LIBRARIES: Numpy, Pandas, Seaborn, Sklearn, Scipy .

Tools: Jupiter notebook, Matplotlib, Scikit-learn, Excel.

- ## Testing of Identified Approaches (Algorithms)

1) K-Neighbors Regressor
2) Decision Tree Regressor
3) Random Forest Regressor
4) Ridge
5) Lasso
6) Logistic Regressor

- Run and Evaluate selected models

- MODELS

RandomForestRegressor

```
rdr = RandomForestRegressor()
rdr.fit(x_train,y_train)
y_train_pred = rdr.predict(x_train)
y_test_pred = rdr.predict(x_test)
```

```
print("Train Results for Random Forest Regressor Model:")
print(50 * '-')
print('Mean_absolute_error', metrics.mean_absolute_error(y_train.values, y_train_pred))
print('Mean_squared_error', metrics.mean_squared_error(y_train.values, y_train_pred))
print('R-squared:', np.sqrt(metrics.mean_squared_error(y_train.values, y_train_pred)))
```

```
Train Results for Random Forest Regressor Model:
--------------------------------------------------
Mean_absolute_error 2.7788801261829654
Mean_squared_error 23.431633280757104
R-squared: 4.840623232679559
```

```
print("Test Results for Random Forest Regressor Model:")
print(50 * '-')
print('Mean_absolute_error', metrics.mean_absolute_error(y_test, y_test_pred))
print('Mean_squared_error', metrics.mean_squared_error(y_test, y_test_pred))
print('R-squared:', np.sqrt(metrics.mean_squared_error(y_test, y_test_pred)))
```

```
Test Results for Random Forest Regressor Model:
--------------------------------------------------
Mean_absolute_error 7.06904761904762
Mean_squared_error 146.50943553113555
R-squared: 12.104108208832882
```

Logistic Regressor

```
logreg = LogisticRegression()
logreg.fit(x_train, y_train)
y_train_pred = logreg.predict(x_train)
y_test_pred = logreg.predict(x_test)
```

```
print("Train Results for LogisticRegression Model:")
print(50 * '-')
print('Mean_absolute_error', metrics.mean_absolute_error(y_train.values, y_train_pred))
print('Mean_squared_error', metrics.mean_squared_error(y_train.values, y_train_pred))
print('R-squared:', np.sqrt(metrics.mean_squared_error(y_train.values, y_train_pred)))
```

```
Train Results for LogisticRegression Model:
--------------------------------------------------
Mean_absolute_error 14.392744479495269
Mean_squared_error 685.3170347003155
R-squared: 26.17856059259782
```

```
print("Test Results for LogisticRegression Model:")
print(50 * '-')
print('Mean_absolute_error', metrics.mean_absolute_error(y_train.values, y_train_pred))
print('Mean_squared_error', metrics.mean_squared_error(y_train.values, y_train_pred))
print('R-squared:', np.sqrt(metrics.mean_squared_error(y_train.values, y_train_pred)))
```

```
Test Results for LogisticRegression Model:
--------------------------------------------------
Mean_absolute_error 14.392744479495269
Mean_squared_error 685.3170347003155
R-squared: 26.17856059259782
```

Ridge

```
rd = Ridge()
rd.fit(x_train,y_train)
y_train_pred = rd.predict(x_train)
y_test_pred = rd.predict(x_test)
```

```
print("Train Results for Ridge Model:")
print(50 * '-')
print('Mean_absolute_error', metrics.mean_absolute_error(y_train.values, y_train_pred))
print('Mean_squared_error', metrics.mean_squared_error(y_train.values, y_train_pred))
print('R-squared:', np.sqrt(metrics.mean_squared_error(y_train.values, y_train_pred)))
```

```
Train Results for Ridge Model:
--------------------------------------------------
Mean_absolute_error 23.23947383556145
Mean_squared_error 806.262982760407
R-squared: 28.394770341744394
```

```
print("Test Results for Ridge Model:")
print(50 * '-')
print('Mean_absolute_error', metrics.mean_absolute_error(y_train.values, y_train_pred))
print('Mean_squared_error', metrics.mean_squared_error(y_train.values, y_train_pred))
print('R-squared:', np.sqrt(metrics.mean_squared_error(y_train.values, y_train_pred)))
```

```
Test Results for Ridge Model:
--------------------------------------------------
Mean_absolute_error 23.23947383556145
Mean_squared_error 806.262982760407
R-squared: 28.394770341744394
```

Lasso

```
lss = Lasso()
lss.fit(x_train,y_train)
y_train_pred = lss.predict(x_train)
y_test_pred = lss.predict(x_test)
```

```
print("Train Results for Lasso Model:")
print(50 * '-')
print('Mean_absolute_error', metrics.mean_absolute_error(y_train.values, y_train_pred))
print('Mean_squared_error', metrics.mean_squared_error(y_train.values, y_train_pred))
print('R-squared:', np.sqrt(metrics.mean_squared_error(y_train.values, y_train_pred)))
```

```
Train Results for Lasso Model:
--------------------------------------------------
Mean_absolute_error 23.260223320119565
Mean_squared_error 808.3898642877767
R-squared: 28.432197668976922
```

```
print("Test Results for Lasso Model:")
print(50 * '-')
print('Mean_absolute_error', metrics.mean_absolute_error(y_test, y_test_pred))
print('Mean_squared_error', metrics.mean_squared_error(y_test, y_test_pred))
print('R-squared:', np.sqrt(metrics.mean_squared_error(y_test, y_test_pred)))
```

```
Test Results for Lasso Model:
--------------------------------------------------
Mean_absolute_error 23.155256128690844
Mean_squared_error 821.7937228702859
R-squared: 28.666944777396246
```

Decision Tree Regressor

```
dtc = DecisionTreeRegressor()
dtc.fit(x_train,y_train)
y_train_pred = dtc.predict(x_train)
y_test_pred = dtc.predict(x_test)
```

```
print("Train Results for Decision Tree Regressor Model:")
print(50 * '-')
print('Mean_absolute_error', metrics.mean_absolute_error(y_train.values, y_train_pred))
print('Mean_squared_error', metrics.mean_squared_error(y_train.values, y_train_pred))
print('R-squared:', np.sqrt(metrics.mean_squared_error(y_train.values, y_train_pred)))
```

```
Train Results for Decision Tree Regressor Model:
--------------------------------------------------
Mean_absolute_error 0.0
Mean_squared_error 0.0
R-squared: 0.0
```

```
print("Test Results for Decision Tree Regressor Model:")
print(50 * '-')
print('Mean_absolute_error', metrics.mean_absolute_error(y_test, y_test_pred))
print('Mean_squared_error', metrics.mean_squared_error(y_test, y_test_pred))
print('R-squared:', np.sqrt(metrics.mean_squared_error(y_test, y_test_pred)))
```

```
Test Results for Decision Tree Regressor Model:
--------------------------------------------------
Mean_absolute_error 4.641025641025641
Mean_squared_error 162.3040293040293
R-squared: 12.739859862024751
```

K Neighbors Regressor

```
knn = KNeighborsRegressor()
knn.fit(x_train,y_train)
y_train_pred = knn.predict(x_train)
y_test_pred = knn.predict(x_test)
```

```
print("Train Results for K Neighbors Regressor Model:")
print(50 * '-')
print('Mean_absolute_error', metrics.mean_absolute_error(y_train.values, y_train_pred))
print('Mean_squared_error', metrics.mean_squared_error(y_train.values, y_train_pred))
print('R-squared:', np.sqrt(metrics.mean_squared_error(y_train.values, y_train_pred)))
```

```
Train Results for K Neighbors Regressor Model:
--------------------------------------------------
Mean_absolute_error 8.902523659305995
Mean_squared_error 208.2780441640379
R-squared: 14.431841329644596
```

```
print("Test Results for K Neighbors Regressor Model:")
print(50 * '-')
print('Mean_absolute_error', metrics.mean_absolute_error(y_test, y_test_pred))
print('Mean_squared_error', metrics.mean_squared_error(y_test, y_test_pred))
print('R-squared:', np.sqrt(metrics.mean_squared_error(y_test, y_test_pred)))
```

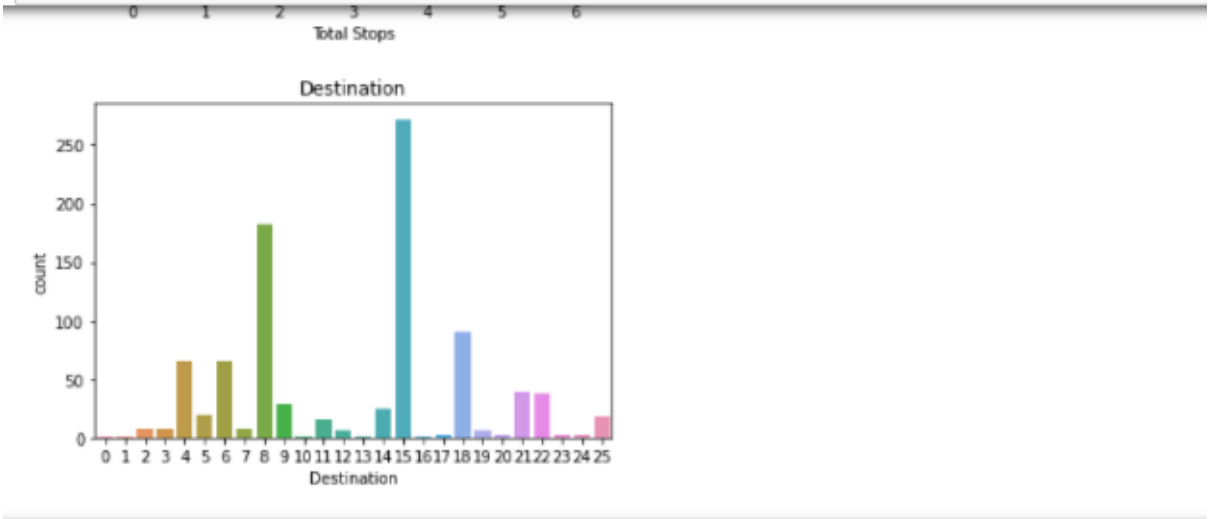
```
Test Results for K Neighbors Regressor Model:
--------------------------------------------------
Mean_absolute_error 11.927472527472526
Mean_squared_error 367.09318681318683
R-squared: 19.159676062323882
```

- Visualizations

    1. Count plot
    2. Boxplot
    3. Distplot
    4. Scatterplot
    5. Heatmap

1) Countplot

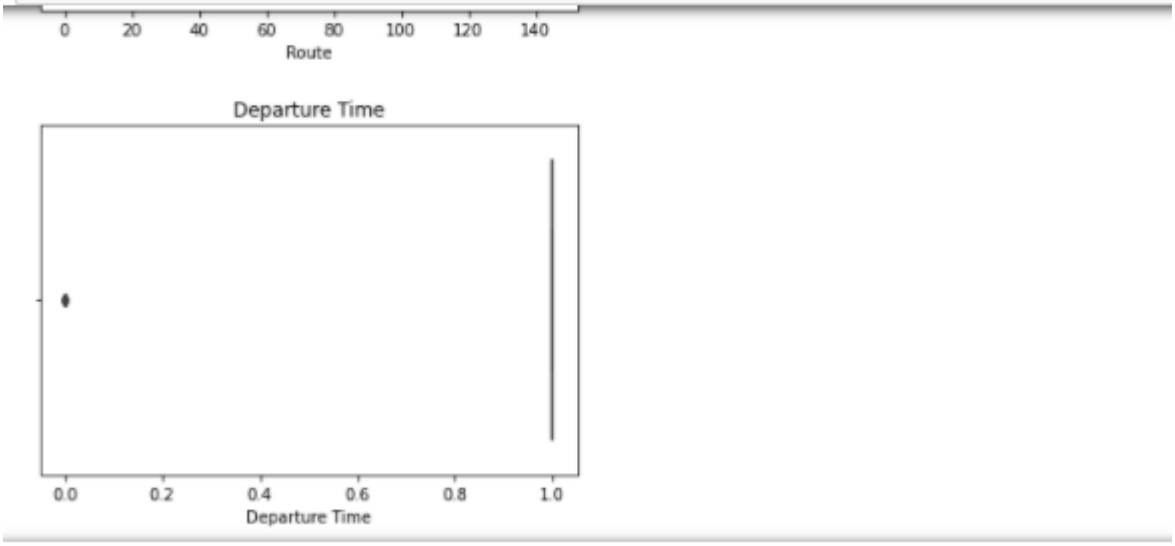
```
for col in ['AirlineName', 'Arrival Time', 'Duration', 'Price',
       'Total Stops', 'Destination', 'Route', 'Departure Time']:
    sns.countplot(df[col].dropna(),orient='v')
    plt.title(col)
    plt.show()
```



countplot is used to represent the counts of the observation present in the categorical variable.

2) Boxplot

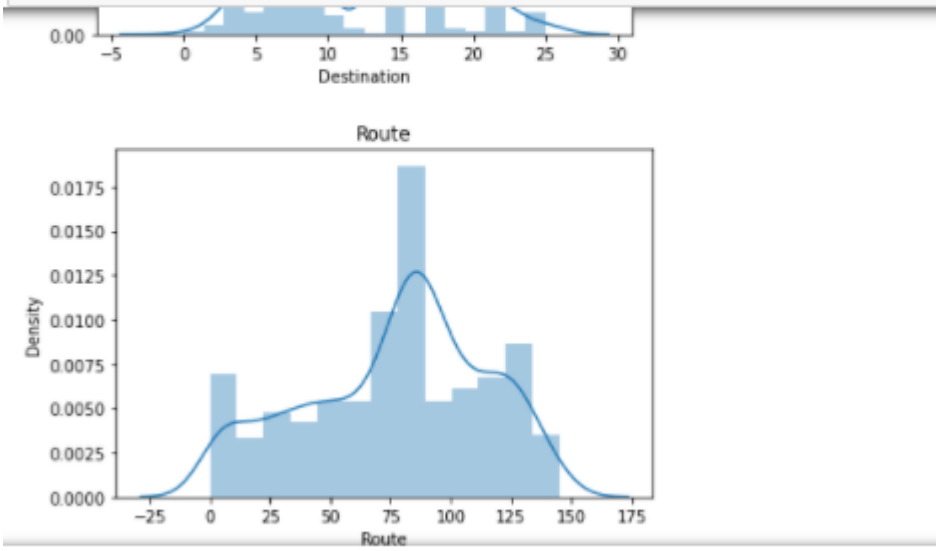
```
for col in ['AirlineName', 'Arrival Time', 'Duration', 'Price',
       'Total Stops', 'Destination', 'Route', 'Departure Time']:
    sns.boxplot(df[col].dropna(),orient='v')
    plt.title(col)
    plt.show()
```



Duration and Departure Time this two columns has outliers are present.

3) Distplot

```
for col in ['AirlineName', 'Arrival Time', 'Duration', 'Price',
       'Total Stops', 'Destination', 'Route', 'Departure Time']:
    sns.distplot(df[col])
    plt.title(col)
    plt.show()
```

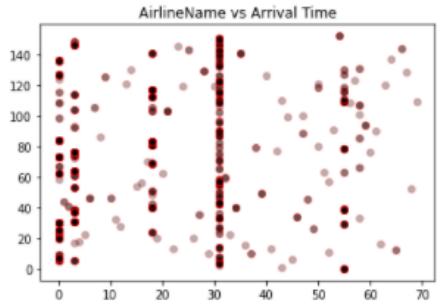


In given plots shows the data is normally distributed or not.

4) Scatterplot

```
# example of Bivariate analysis
plt.scatter(df["AirlineName"],df["Arrival Time"],alpha=0.3,c=(0,0,0),edgecolors='r')
plt.title("AirlineName vs Arrival Time")
plt.show()
```

```
*c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapping will have precedence in
case its length matches with *x* & *y*.  Please use the *color* keyword-argument or provide a 2-D array with a single row if yo
u intend to specify the same RGB or RGBA value for all points.
```

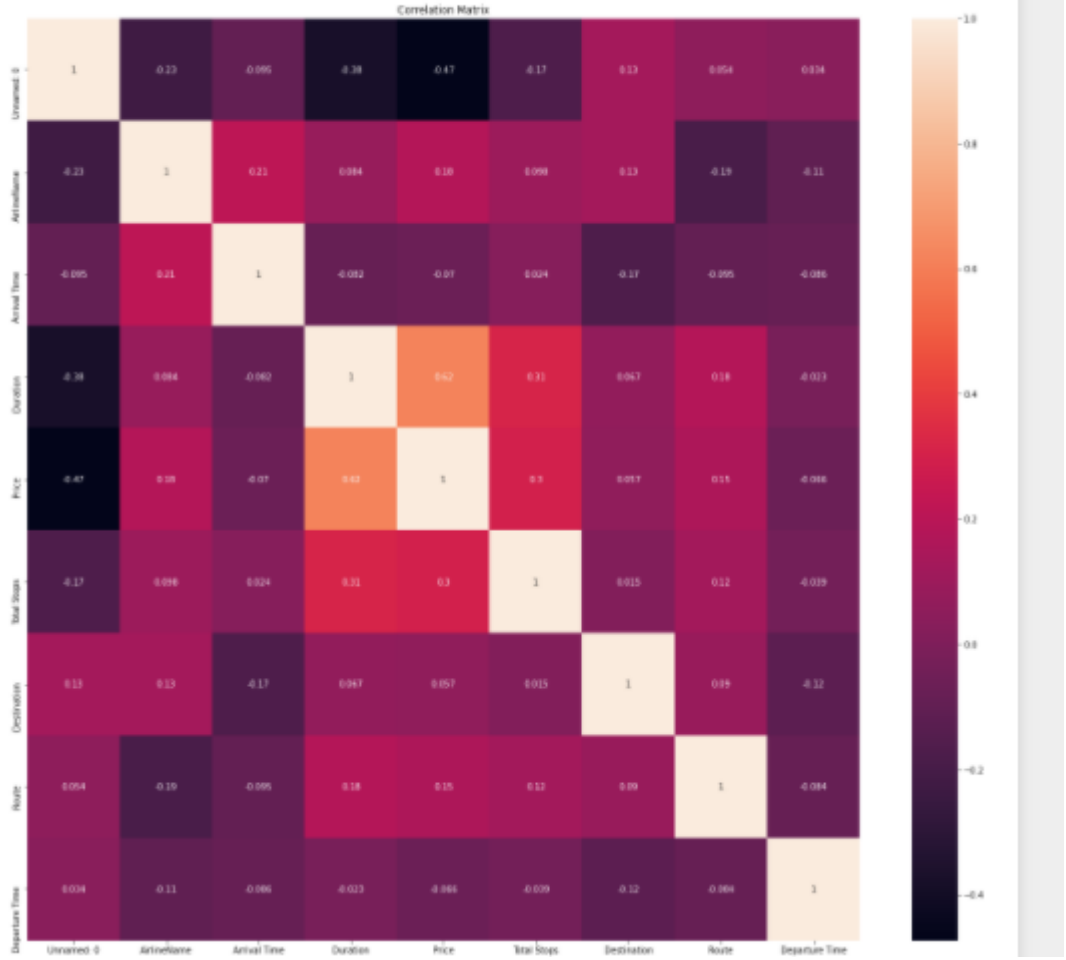


In this plot outliers are present.

5) Heatmap

```
# example of Multivariate  analysis

plt.figure(figsize=[22,20])
sns.heatmap(cor,annot=True)
plt.title("Correlation Matrix")
plt.show()
```



CONCLUSION

In these project, Decision Tree Regressor, Random Forest Regressor, K-Neighbors Regressor, Ridge, Lasso these models are used.

Random Forest Regressor is r2 score ,Mean absolute error, Mean_squared_error and R-squared is good than other models, Hence Decision Tree Regressor is performing is good.

Random Forest Regressor is best model for this dataset.

```
conclusion
```

	0	1	2	3	4	5	6	7	8	9	...	263	264	265	266	267	268	269	270	271	272
predicted	51.36	64.56	94.12	28.0	52.81	92.0	21.37	48.71	28.09	53.0	...	69.95	55.8	77.36	28.13	130.01	13.88	21.42	84.46	22.08	42.73
original	50.20	72.40	76.20	28.0	91.80	92.0	22.80	38.80	40.60	53.0	...	73.20	47.6	73.20	33.60	125.00	26.60	24.60	70.40	51.60	54.40

2 rows × 273 columns