FLIP ROBO

Flights Fare Prediction

Submitted by:

Akriti Kakkar

ACKNOWLEDGMENT

I have referred to data trained study material and python documentation for this model development.

INTRODUCTION

- ## Business Problem Framing

  Anyone who has booked a flight ticket knows how unexpectedly the prices vary. The cheapest available ticket on a given flight gets more and less expensive over time. This usually happens as an attempt to maximize revenue based on - 1. Time of purchase patterns (making sure last-minute purchases are expensive) 2. Keeping the flight as full as they want it (raising prices on a flight which is filling up in order to reduce sales and hold back inventory for those expensive last-minute expensive purchases)

- ## Conceptual Background of the Domain Problem

  Trip Type refers to whether it is only departure journey, only return journey or multi city journey.

  Flight From depicts from which place the flight will start its journey.

  Flight To depicts to which place the flight will land.

  Stops refers to number of stops during the journey.

  Duration refers to the time covered by the journey.

  Flight Name, includes, Indigo, Vistara, etcetera.

  Flight Id is the unique id of each flight.

  Fare Type are regular and double seater, Regular depicts fare for one seat; double seater depicts fare for booking an extra seat for maintaining distance.

  Departure Time is the time at which the flight will leave.

  Arrival Time is the time at which the flight will arrive.

  Depart Date is the date at which the flight is booked.

  Booking Date is the date on which the booking is done.

  Passenger refers to number and type of passenger booking the flight.

Class includes, economy, premium economy, business.

- Review of Literature

Follow the link to the dashboard:

https://docs.google.com/spreadsheets/d/1BcxTRkYat8TlHQ35WjviCbmuBcQ9TpuBmBA0dHoq9JE/edit?usp=sharing

Follow the link to the dashboard:

https://docs.google.com/spreadsheets/d/1zRBnzBfrn06Ip3Ztew-ifQT5xhNG-nuplFCaoHbLqIg/edit?usp=sharing

Follow the link to the dashboard:

https://docs.google.com/spreadsheets/d/11aaWnLDCOk_3OxmIxa-E9i5Q5Djidj5btNaj21m5NlY/edit?usp=sharing

Follow the link to the dashboard:

https://docs.google.com/spreadsheets/d/1d95xWuFk026-TytCxlVYUVt5SgWN91CEzRt3aQcWdf8/edit?usp=sharing

Follow the link to the dashboard:

https://docs.google.com/spreadsheets/d/1wi3l_U59TCL6ZoPGOsKqzI-SU0VrJMOfiX-SHjiisFU/edit?usp=sharing

Follow the link to the dashboard:

https://docs.google.com/spreadsheets/d/1JrB96CFH2ZVug7wGFlDakAwhRJOyN9Tcew8d_atfJTA/edit?usp=sharing

Follow the link to the dashboard:

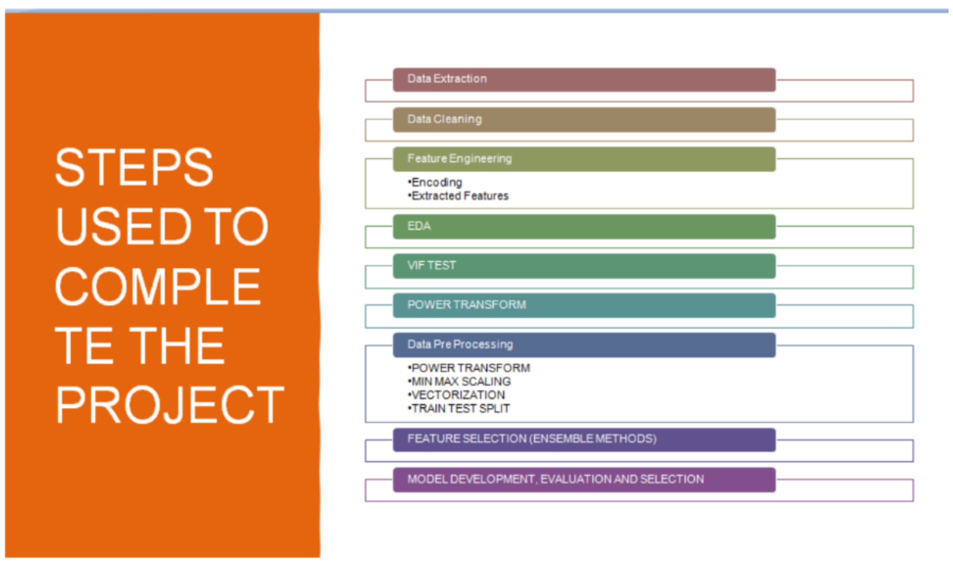
https://docs.google.com/spreadsheets/d/1Qa3IOUpyVLk-7F0LlWu2fGzcWmLuuMgGkhOumoLoWhE/edit?usp=sharing

- Motivation for the Problem Undertaken

  Anyone who has booked a flight ticket knows how unexpectedly the prices vary. The cheapest available ticket on a given flight gets more and less expensive over time. This usually happens as an attempt to maximize revenue based on - 1. Time of purchase patterns (making sure last-minute purchases are expensive) 2. Keeping the flight as full as they want it (raising prices on a flight which is filling up in order to reduce sales and hold back inventory for those expensive last-minute expensive purchases)

Analytical Problem Framing

- Mathematical/ Analytical Modeling of the Problem



1. Data Extraction: Storing scraped data from make my trip in a csv. Using read_ csv function of pandas library to read the data in tabulated format and analyze it.

2. Data Cleaning for missing values detection and its handiling.

3. Feature Engineering for encoding object format data and deriving more features.

4. EDA For data visualization and biasness detection:

· HEAD VIEW OF DATA

· TAIL VIEW OF DATA

· SAMPLE VIEW OF DATA

· GROUPBY EXPLORATION

· DESCRIPTIVE STATISTICS

· SCATTER PLOTS

· CORRELATION ANALYSIS

· BOX PLOTS EXPLORATION

· DESCRIPTIVE STATISTICS

· DISTRIBUTION PLOTS

5. VIF Test for multicollinearity reduction.

6. Power Transformation for standard scaling and outliers transformation.

7. Data PreProcessing for data transformation, scaling and vectorization.

8. Feature Selection (Ensemble Methods): ANOVA Test, p value, ftest, constant threshold filter to classify features based on relevance and biasness and select the most relevant features.

9. Model Development, Evaluation And Selection (Ensemble Methods and Grid Search CV) to do best hyper parameter tuning and develop low bias and low variance with right fit and minimal difference between test metrics and train metrics.

- Data Sources and their formats
    a. **Data Collection Source: Error! Hyperlink reference not valid.**
    b. Data Collection Technique: Web Scraping
    c. Data Collection Technical Tools: Python+selenium
    d. Data Collection Code Technical Tool: Jupyer Notebook NBFormat
    e. Data Collection Source Code File: data_script1.ipynb
    f. Data Collection Output: flights_data.zip
    g. Data Format Used In Present Code: flights_data.csv
    h.
- Data Preprocessing Done

· Data Pre Processing

· POWER TRANSFORM

· MIN MAX SCALING

· VECTORIZATION

· TRAIN TEST SPLIT

The data was further used for feature selection.

Assumption made:

• Acceptable Skewness Range Is +/-0.65

• Acceptable VIF Score Is Than 6

• Acceptable P Value Is Less Than 0.05

• Variance Threshold Is 0.01

- Data Inputs- Logic- Output Relationships
- **Data Inputs include:**

  ['Depart_Date_encoded', 'Flight Name_encoded',
  'Arrival Time_encoded_pct_change', 'Depart Time_encoded']

  Data Input Type: float64, MinMaxScaling: 0 to 1.

  Impact on output:

Depart_Date_encoded	-0.471487
Flight Name_encoded	0.098114
Arrival Time_encoded_pct_change	-0.016126
Depart Time_encoded	0.008823

- State the set of assumptions (if any) related to the problem under consideration

Assumption made:

• Acceptable Skewness Range Is +/-0.65

• Acceptable VIF Score Is Than 6

• Acceptable P Value Is Less Than 0.05

• Variance Threshold Is 0.01

- Hardware and Software Requirements and Tools Used

Installation Of Anaconda Library.

Required Installations:

· Pandas (Within environment)

· Numpy (Within environment)

· Seaborn (Within environment)

· Matplotlib (Within environment)

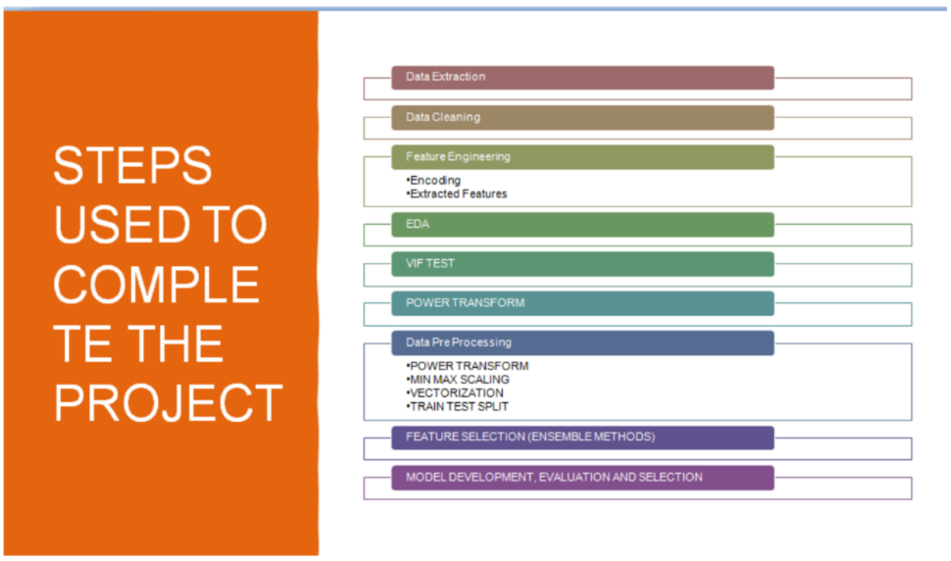
· Cufflinks

· Plotly Express

· Sklearn

Model/s Development and Evaluation

- Identification of possible problem-solving approaches (methods)



· Total Models = 6

· Selection Reasoning Of Models 1 to 6

· The goal of ensemble methods is to combine the predictions of several base estimators built with a given learning algorithm in order to improve generalizability / robustness over a single estimator.

· Two families of ensemble methods are usually distinguished:

· In averaging methods, the driving principle is to build several estimators independently and then to average their predictions. On average, the combined estimator is usually better than any of the single base estimator because its variance is reduced.

· Examples: Bagging methods, Forests of randomized trees, etcetera

· By contrast, in boosting methods, base estimators are built sequentially and one tries to reduce the bias of the combined estimator. The motivation is to combine several weak models to produce a powerful ensemble.

· Examples: AdaBoost, Gradient Tree Boosting,etcetera

· The use case assigned revolves around uneven label points hence there is high probality of not achieving a good fit. Hence, I have tried different ensembele techniques that can lower variance and bias and help achieve good scores. (Just as a

reminder, I have already applied variance threshold of 0.01 to ensure that risk of models is low).

· The theories in the above two cells explain why I have chosen Model 1, Model 2, Model 3, Model 4, Model 5 and Model 6

Total Models = 6

• Model 1: Random Forest Regressor With Grid Search CV Hyper Parameter Tuning

• Model 2: Random Forest Regressor With Default Hyper Parameter Tuning

• Model 3: Ada Boost Regressor And Random Forest Regressor With Grid Search CV Hyper Parameter Tuning and Ada Boost Boosting

• Model 4: Extra Trees Regressor With Grid Search CV Hyper Parameter Tuning

• Model 5: Linear Regression With Intuitional Hyper Parameter Tuning

• Model 6: Ada Boost Regressor With Huber Regressor As Base Estimator

• Testing of Identified Approaches (Algorithms)

Total Models = 6

• Model 1: Random Forest Regressor With Grid Search CV Hyper Parameter Tuning

• Model 2: Random Forest Regressor With Default Hyper Parameter Tuning

• Model 3: Ada Boost Regressor And Random Forest Regressor With Grid Search CV Hyper Parameter Tuning and Ada Boost Boosting

• Model 4: Extra Trees Regressor With Grid Search CV Hyper Parameter Tuning

• Model 5: Linear Regression With Intuitional Hyper Parameter Tuning

• Model 6: Ada Boost Regressor With Huber Regressor As Base Estimator

- Run and Evaluate selected models
- Follow the link to the dashboard:
  https://docs.google.com/spreadsheets/d/12dCTv2HCX_gxafSXbNRfM-K4NI_beEdblLqp4yqEGUU/edit?usp=sharing

- Key Metrics for success in solving problem under consideration
  1. Power Transform: To remove outliers from extremely spread out data.
  2. VIF Scores: To reduce multicollinearity from a highly biased dataset.
  3. Ensemble Methods: To remove over fitting in a complex dataset and finding maximum explanatory power .

- Visualizations

Follow the link to the dashboard:

https://docs.google.com/spreadsheets/d/1BcxTRkYat8TlHQ35WjviCbmuBcQ9TpuBmBA0dHoq9JE/edit?usp=sharing

Follow the link to the dashboard:

https://docs.google.com/spreadsheets/d/1zRBnzBfrn06Ip3Ztew-ifQT5xhNG-nuplFCaoHbLqIg/edit?usp=sharing

Follow the link to the dashboard:

https://docs.google.com/spreadsheets/d/11aaWnLDCOk_3OxmIxa-E9i5Q5Djidj5btNaj21m5NlY/edit?usp=sharing

Follow the link to the dashboard:

https://docs.google.com/spreadsheets/d/1d95xWuFk026-TytCxlVYUVt5SgWN91CEzRt3aQcWdf8/edit?usp=sharing

Follow the link to the dashboard:

https://docs.google.com/spreadsheets/d/1wi3l_U59TCL6ZoPGOsKqzI-SU0VrJMOfiX-SHjiisFU/edit?usp=sharing

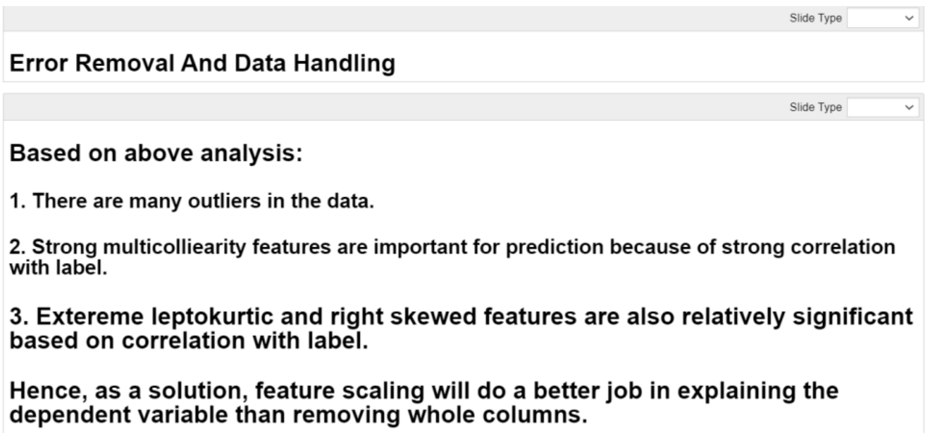
Follow the link to the dashboard:

https://docs.google.com/spreadsheets/d/1JrB96CFH2ZVug7wGFlDakAwhRJOyN9Tcew8d_atfJTA/edit?usp=sharing

Follow the link to the dashboard:

https://docs.google.com/spreadsheets/d/1Qa3IOUpyVLk-7F0LlWu2fGzcWmLuuMgGkhOumoLoWhE/edit?usp=sharing

- Interpretation of the Results



Error Removal And Data Handling

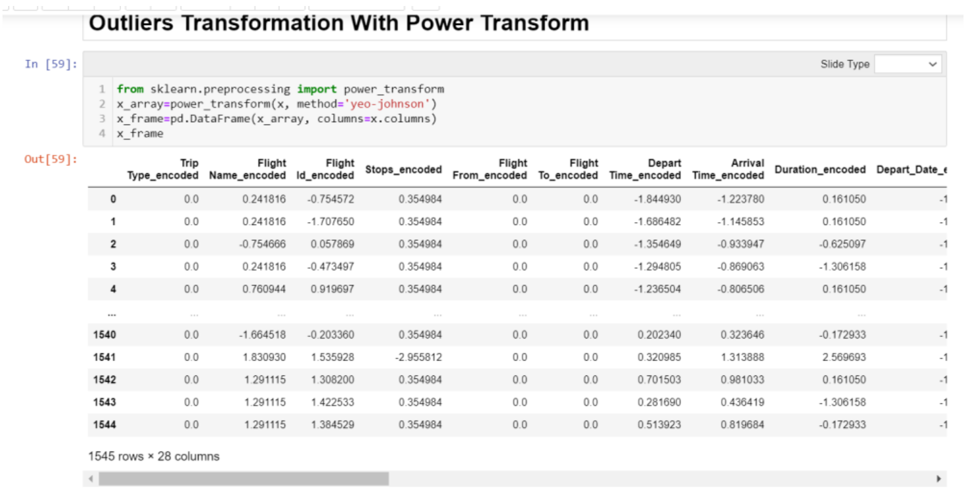
Based on above analysis:

1. There are many outliers in the data.

2. Strong multicolliearity features are important for prediction because of strong correlation with label.

3. Extereme leptokurtic and right skewed features are also relatively significant based on correlation with label.

Hence, as a solution, feature scaling will do a better job in explaining the dependent variable than removing whole columns.



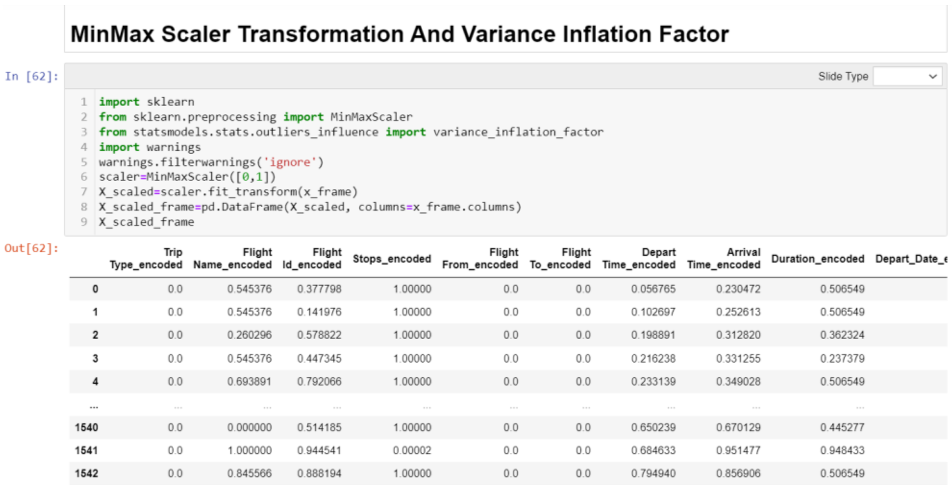
Outliers Transformation With Power Transform

```
In [59]:
from sklearn.preprocessing import power_transform
x_array=power_transform(x, method='yeo-johnson')
x_frame=pd.DataFrame(x_array, columns=x.columns)
x_frame
```

Out[59]:

	Trip Type_encoded	Flight Name_encoded	Flight Id_encoded	Stops_encoded	Flight From_encoded	Flight To_encoded	Depart Time_encoded	Arrival Time_encoded	Duration_encoded	Depart_Date_e
0	0.0	0.241816	-0.754572	0.354984	0.0	0.0	-1.844930	-1.223780	0.161050	-1
1	0.0	0.241816	-1.707650	0.354984	0.0	0.0	-1.686482	-1.145853	0.161050	-1
2	0.0	-0.754666	0.057869	0.354984	0.0	0.0	-1.354649	-0.933947	-0.625097	-1
3	0.0	0.241816	-0.473497	0.354984	0.0	0.0	-1.294805	-0.869063	-1.306158	-1
4	0.0	0.760944	0.919697	0.354984	0.0	0.0	-1.236504	-0.806506	0.161050	-1
...	...	...	...	...	...	...	...	...	...	
1540	0.0	-1.664518	-0.203360	0.354984	0.0	0.0	0.202340	0.323646	-0.172933	-1
1541	0.0	1.830930	1.535928	-2.955812	0.0	0.0	0.320985	1.313888	2.569693	-1
1542	0.0	1.291115	1.308200	0.354984	0.0	0.0	0.701503	0.981033	0.161050	-1
1543	0.0	1.291115	1.422533	0.354984	0.0	0.0	0.281690	0.436419	-1.306158	-1
1544	0.0	1.291115	1.384529	0.354984	0.0	0.0	0.513923	0.819684	-0.172933	-1

1545 rows × 28 columns

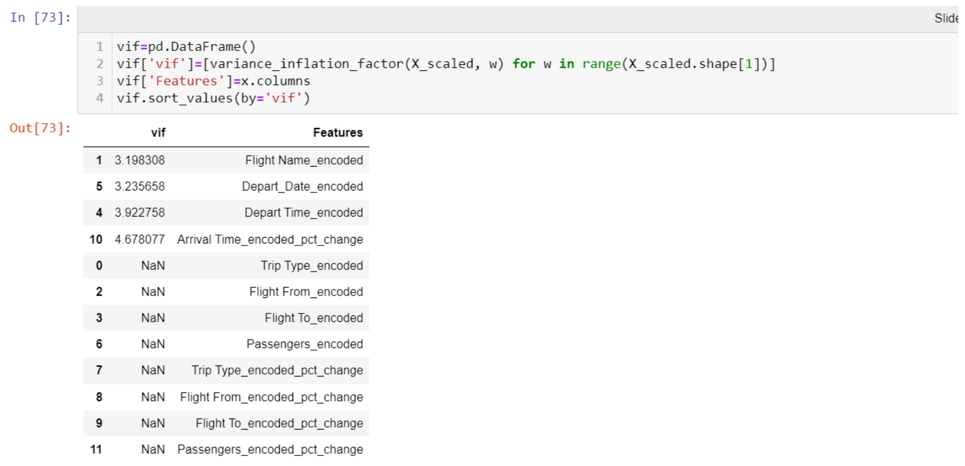


MinMax Scaler Transformation And Variance Inflation Factor

```
In [62]:
import sklearn
from sklearn.preprocessing import MinMaxScaler
from statsmodels.stats.outliers_influence import variance_inflation_factor
import warnings
warnings.filterwarnings('ignore')
scaler=MinMaxScaler([0,1])
X_scaled=scaler.fit_transform(x_frame)
X_scaled_frame=pd.DataFrame(X_scaled, columns=x_frame.columns)
X_scaled_frame
```

Out[62]:

	Trip Type_encoded	Flight Name_encoded	Flight Id_encoded	Stops_encoded	Flight From_encoded	Flight To_encoded	Depart Time_encoded	Arrival Time_encoded	Duration_encoded	Depart_Date_e
0	0.0	0.545376	0.377798	1.00000	0.0	0.0	0.056765	0.230472	0.506549	
1	0.0	0.545376	0.141976	1.00000	0.0	0.0	0.102697	0.252613	0.506549	
2	0.0	0.260296	0.578822	1.00000	0.0	0.0	0.198891	0.312820	0.362324	
3	0.0	0.545376	0.447345	1.00000	0.0	0.0	0.216238	0.331255	0.237379	
4	0.0	0.693891	0.792066	1.00000	0.0	0.0	0.233139	0.349028	0.506549	
...	...	...	...	...	...	...	...	...	...	
1540	0.0	0.000000	0.514185	1.00000	0.0	0.0	0.650239	0.670129	0.445277	
1541	0.0	1.000000	0.944541	0.00002	0.0	0.0	0.684633	0.951477	0.948433	
1542	0.0	0.845566	0.888194	1.00000	0.0	0.0	0.794940	0.856906	0.506549	



In [73]:

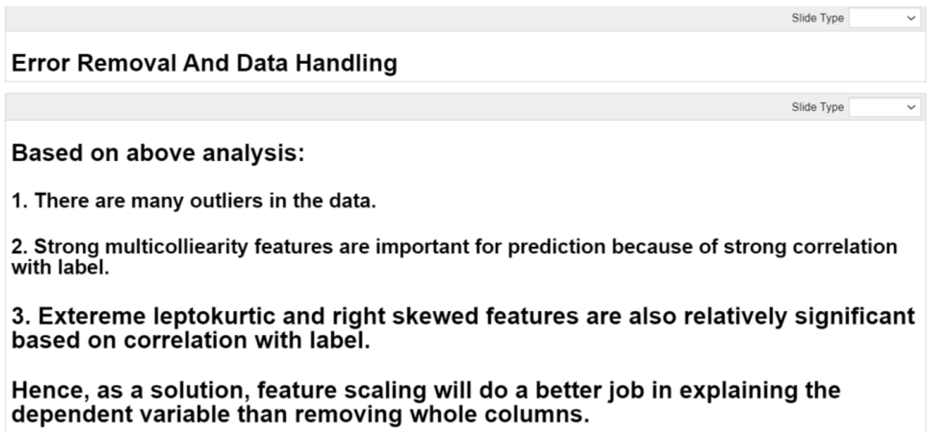
```
vif=pd.DataFrame()
vif['vif']=[variance_inflation_factor(X_scaled, w) for w in range(X_scaled.shape[1])]
vif['Features']=x.columns
vif.sort_values(by='vif')
```

Out[73]:

	vif	Features
1	3.198308	Flight Name_encoded
5	3.235658	Depart_Date_encoded
4	3.922758	Depart Time_encoded
10	4.678077	Arrival Time_encoded_pct_change
0	NaN	Trip Type_encoded
2	NaN	Flight From_encoded
3	NaN	Flight To_encoded
6	NaN	Passengers_encoded
7	NaN	Trip Type_encoded_pct_change
8	NaN	Flight From_encoded_pct_change
9	NaN	Flight To_encoded_pct_change
11	NaN	Passengers_encoded_pct_change

CONCLUSION

- Key Findings and Conclusions of the Study



Error Removal And Data Handling

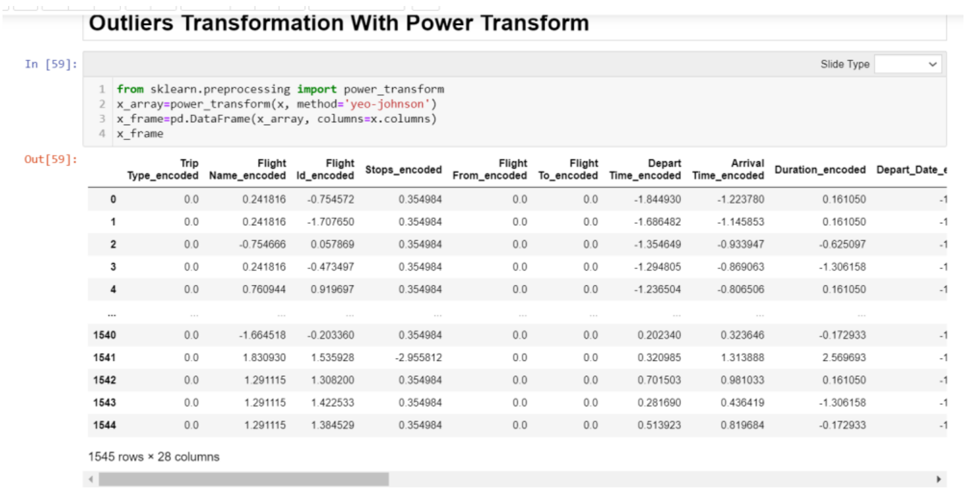
Based on above analysis:

1. There are many outliers in the data.

2. Strong multicolliearity features are important for prediction because of strong correlation with label.

3. Extereme leptokurtic and right skewed features are also relatively significant based on correlation with label.

Hence, as a solution, feature scaling will do a better job in explaining the dependent variable than removing whole columns.



Outliers Transformation With Power Transform

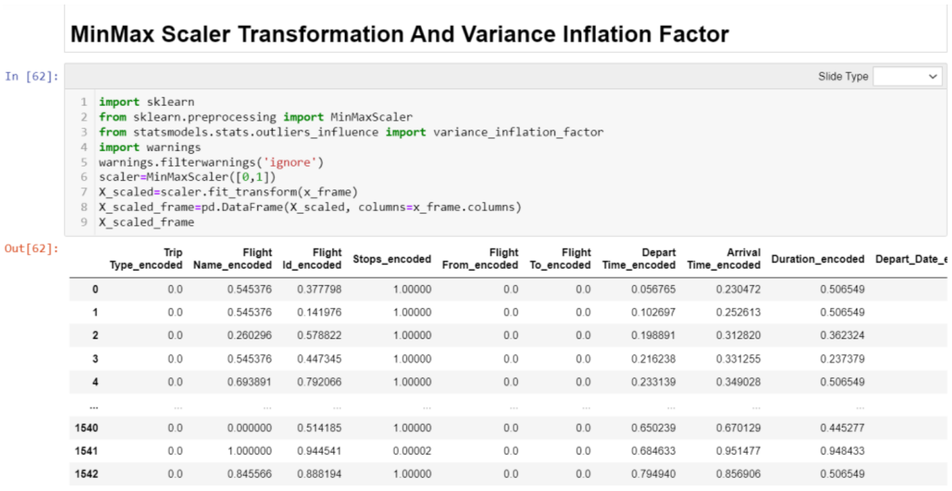
In [59]:

```python
from sklearn.preprocessing import power_transform
x_array=power_transform(x, method='yeo-johnson')
x_frame=pd.DataFrame(x_array, columns=x.columns)
x_frame
```

Out[59]:

```
              Trip         Flight       Flight                        Flight          Flight       Depart        Arrival
      Type_encoded   Name_encoded   Id_encoded   Stops_encoded   From_encoded   To_encoded   Time_encoded   Time_encoded   Duration_encoded   Depart_Date_e
   0           0.0       0.241816    -0.754572        0.354984            0.0          0.0      -1.844930      -1.223780           0.161050              -1
   1           0.0       0.241816    -1.707650        0.354984            0.0          0.0      -1.686482      -1.145853           0.161050              -1
   2           0.0      -0.754666     0.057869        0.354984            0.0          0.0      -1.354649      -0.933947          -0.625097              -1
   3           0.0       0.241816    -0.473497        0.354984            0.0          0.0      -1.294805      -0.869063          -1.306158              -1
   4           0.0       0.760944     0.919697        0.354984            0.0          0.0      -1.236504      -0.806506           0.161050              -1
 ...           ...            ...          ...             ...            ...          ...            ...            ...                ...
1540           0.0      -1.664518    -0.203360        0.354984            0.0          0.0       0.202340       0.323646          -0.172933              -1
1541           0.0       1.830930     1.535928       -2.955812            0.0          0.0       0.320985       1.313888           2.569693              -1
1542           0.0       1.291115     1.308200        0.354984            0.0          0.0       0.701503       0.981033           0.161050              -1
1543           0.0       1.291115     1.422533        0.354984            0.0          0.0       0.281690       0.436419          -1.306158              -1
1544           0.0       1.291115     1.384529        0.354984            0.0          0.0       0.513923       0.819684          -0.172933              -1

1545 rows × 28 columns
```



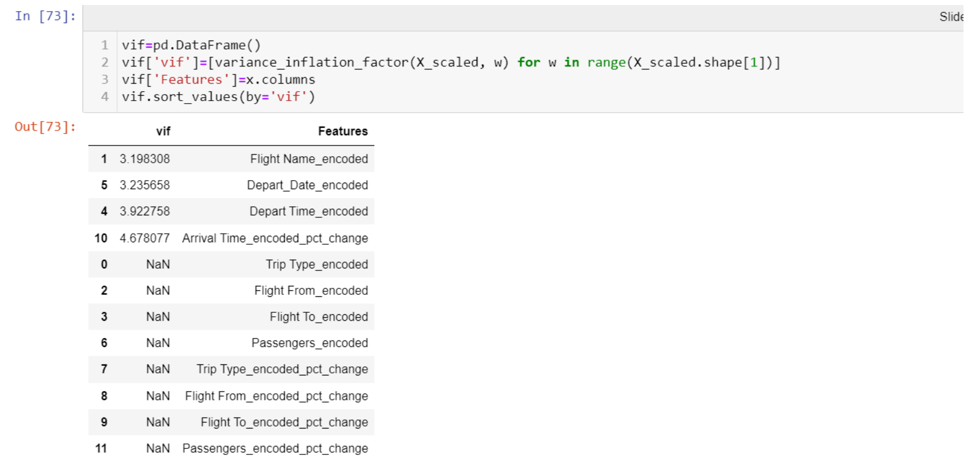
MinMax Scaler Transformation And Variance Inflation Factor

In [62]:

```python
import sklearn
from sklearn.preprocessing import MinMaxScaler
from statsmodels.stats.outliers_influence import variance_inflation_factor
import warnings
warnings.filterwarnings('ignore')
scaler=MinMaxScaler([0,1])
X_scaled=scaler.fit_transform(x_frame)
X_scaled_frame=pd.DataFrame(X_scaled, columns=x_frame.columns)
X_scaled_frame
```

Out[62]:

```
              Trip         Flight       Flight                        Flight          Flight       Depart        Arrival
      Type_encoded   Name_encoded   Id_encoded   Stops_encoded   From_encoded   To_encoded   Time_encoded   Time_encoded   Duration_encoded   Depart_Date_e
   0           0.0       0.545376     0.377798         1.00000            0.0          0.0       0.056765       0.230472           0.506549
   1           0.0       0.545376     0.141976         1.00000            0.0          0.0       0.102697       0.252613           0.506549
   2           0.0       0.260296     0.578822         1.00000            0.0          0.0       0.198891       0.312820           0.362324
   3           0.0       0.545376     0.447345         1.00000            0.0          0.0       0.216238       0.331255           0.237379
   4           0.0       0.693891     0.792066         1.00000            0.0          0.0       0.233139       0.349028           0.506549
 ...           ...            ...          ...             ...            ...          ...            ...            ...                ...
1540           0.0       0.000000     0.514185         1.00000            0.0          0.0       0.650239       0.670129           0.445277
1541           0.0       1.000000     0.944541         0.00002            0.0          0.0       0.684633       0.951477           0.948433
1542           0.0       0.845566     0.888194         1.00000            0.0          0.0       0.794940       0.856906           0.506549
```



In [73]:

```python
vif=pd.DataFrame()
vif['vif']=[variance_inflation_factor(X_scaled, w) for w in range(X_scaled.shape[1])]
vif['Features']=x.columns
vif.sort_values(by='vif')
```

Out[73]:

```
         vif                       Features
 1  3.198308            Flight Name_encoded
 5  3.235658            Depart_Date_encoded
 4  3.922758            Depart Time_encoded
10  4.678077  Arrival Time_encoded_pct_change
 0       NaN               Trip Type_encoded
 2       NaN             Flight From_encoded
 3       NaN               Flight To_encoded
 6       NaN              Passengers_encoded
 7       NaN    Trip Type_encoded_pct_change
 8       NaN  Flight From_encoded_pct_change
 9       NaN    Flight To_encoded_pct_change
11       NaN   Passengers_encoded_pct_change
```

- ## Learning Outcomes of the Study in respect of Data Science

Visualizations and data cleaning convert a whole complex and messy dataset into insightful and interesting representation, which make it easier to reach the core of the problem and solve it.

The best model is Ada Boost Regressor With Huber Regressor As Base Estimator, the most challenging part in models development process was to reduce overfitting and that is why I have applied ensemble methods on base estimators... Huber Regressor provided the best framework to reduce overfitting.

- Limitations of this work and Scope for Future Work

Further optimatization can be obtained by applying deep learning solutions. Since, it requires very high RAM capacity, it could not be displayed in jupyter notebook... I would like to update Google Colab Notebook for future projects, if acceptable... That can help me to submit a completely optimized model.