PGP_DSE_FT CAPSTONE PROJECT FINAL REPORT

BATCH DETAILS	DSE_FT_CHENNAI_OCT20
TEAM MEMBERS	1. ANANDRAM G 2. MANJUSHA RAJAN 3. PRIYA SINGH 4. SARAVANA ALAGAR 5. SRINATH K
DOMAIN OF PROJECT	AIRLINE INDUSTRY
PROPOSED PROJECT TITLE	ANALYSIS OF FLIGHT DELAYS USING MACHINE LEARNING MODEL
GROUP NUMBER	GROUP-2
TEAM LEADER	SRINATH K
MENTOR NAME	MS. VIDHYA KANNAIAH

1.INTRODUCTION

The airline industry itself is a major economic force, both in terms of its operations and its impacts on related industries such as aircraft manufacturing and tourism, to name but two. The worldwide aviation industry is vibrant, energetic and constantly changing as a result of competition, new technologies, global incidents, and worldwide economic factors.

In general, the marketability of the product is judged by the timeliness, accuracy, functionality, quality, and price of the service. As perceived by air transportation customers, these criteria translate into flexible schedules, on-time flights, safety, satisfactory in-flight services, proper baggage handling, and convenient ticket purchases.

To operate a flight efficiently a flight is challenging on so many different levels. Considering one major reason for its disruption is the delay of flights.

SCOPE OF STUDY:

Airlines delays are one of the most important issues which need to be considered because they incur high overhead costs for airlines on the one hand and airports on the other. The purpose of this study project is to use the most relevant machine learning methods in real data to predict flight delays for all factors such as weather, passenger delays, maintenance, and airport congestion. These can be minimized or improved by developing highly efficient flight plans.

We are going to use Python programming language to run the codes for our computational purposes. We will analyze different methods from the point of view of accuracy and propose an integrated method to improve our prediction results.

OBJECTIVES:

- To identify a root cause for the delays and also considering its impact throughout the network.
- To gain insights/inferences on how these delays can be reduced with different efficient practices

2.METHODOLOGY

We will be using the Jupyter notebook to do all the work on the dataset. We will be importing all the required libraries such as

- NumPy
- Pandas
- Matplotlib.pyplot
- Seaborn

And added to this there will be more libraries during further hands-on work of the dataset. The data is collected from the Kaggle website. The data need to be read and the initial pre-processing will be done.

Also, the various inferences will be derived from the EDA techniques, which would give the insights of the various factors with respect to the variables/features present in the dataset.

By plotting the graph, charts and computing various statistical techniques, we will be able to do an analysis with the inferential statistics concepts. Later we will build the suitable Machine learning model which would help in obtaining our objective of the project.

From the colossal dataset, we will consider the sample that would be used to implement all the above process. We will concentrate on a particular airport or the top few busiest airports and then derive the output based on our work.

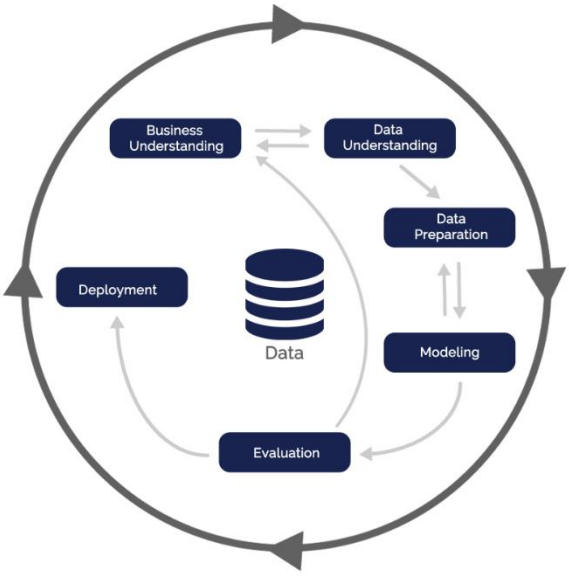


Fig1.1: CRISP-DM, Source: https://analyticsindiamag.com/crisp-dm-data-science-project/Opinions/How_The_CRISP-DM_Method_Can_Help_Manage_Your_Next_Data_Science_Project

RESEARCH DATASET:

Main Dataset

- Flight delay details in various US airports in 2018

POPULATION

- O'Hare international Airport, Chicago

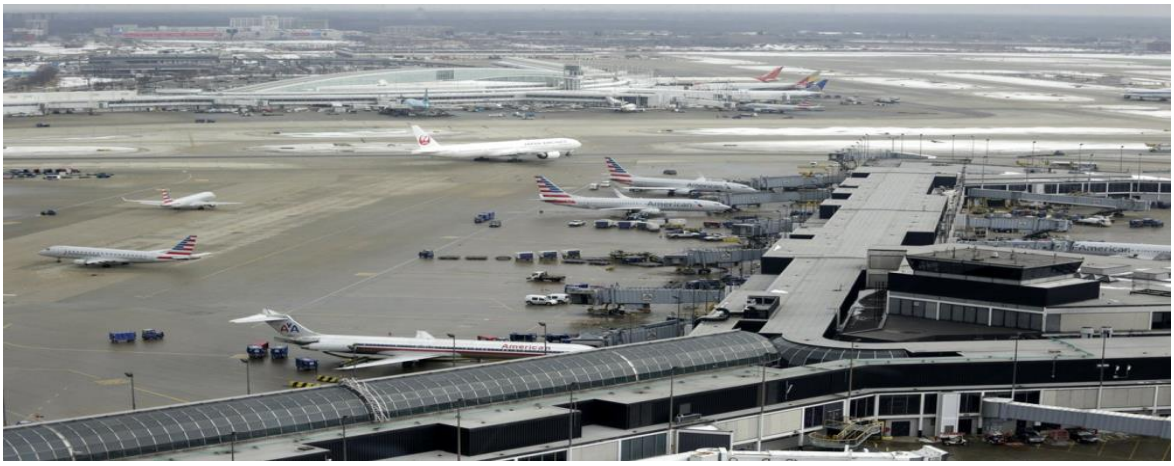
Sample Unit

- Flights details which are delayed

The source of our dataset is downloaded from the Kaggle site. The data set is for the year 2018 and consists of well over 7 Million examples with 28 features categorized as follows:

- Information about the flight (flight date, airline, flight number, tail number)
- Information about origin and destination (origin airport, destination airport)
- Information about the departure (scheduled departure, departure time, departure delay, taxi-out, wheels-off)
- Information about the flight-journey (scheduled time, elapsed time, air time, distance)
- Information about the arrival (wheels-on, taxi-in, scheduled arrival, arrival time, arrival delay)
- Information about diversion, cancellation, and reason of delay (air system delay, security delay, airline delay, late aircraft delay, weather delay)

In this research, as mentioned in the sample unit, we have considered the flight delays details of O'Hare International Airport, Chicago exclusively. Out of the existing airports we have constrained our work on an individual airport and selected the top busiest destination airports.



The details of the columns/variables/features are given below in the table:

COLUMN DETAILS	DESCRIPTION
FL_DATE	Date of the flight, yy/mm/dd
OP_CARRIER	Airline Identifier
OP_CARRIER_FL_NUM	Flight Number
ORIGIN	Starting Airport Code
DEST	Destination Airport Code
CRS_DEP_TIME	Planned Departure Time
DEP_TIME	Actual Departure Time
DEP_DELAY	Total Delay on Departure in minutes
TAXI_OUT	The time duration elapsed between departure from the origin airport gate and wheels off
WHEELS_OFF	The time point that the aircraft's wheels leave the ground
WHEELS_ON	The time point that the aircraft's wheels touch on the ground
TAXI_IN	The time duration elapsed between wheels-on and gate arrival at the destination airport
CRS_ARR_TIME	Planned arrival time
ARR_TIME	Actual Arrival Time
ARR_DELAY	Total Delay on Arrival in minutes
CANCELLED	Flight Cancelled (1 = cancelled)
CANCELLATION_CODE	Reason for Cancellation of flight: A - Airline/Carrier; B - Weather; C - National Air System; D - Security
DIVERTED	Aircraft landed on airport that out of schedule
CRS_ELAPSED_TIME	Planned time amount needed for the flight trip
ACTUAL_ELAPSED_TIME	AIR_TIME+TAXI_IN+TAXI_OUT
AIR_TIME	The time duration between wheels_off and wheels_on time
DISTANCE	Distance between two airports
CARRIER_DELAY	Delay caused by the airline in minutes
WEATHER_DELAY	Delay caused by weather
NAS_DELAY	Delay caused by air system
SECURITY_DELAY	Delay caused by security
LATE_AIRCRAFT_DELAY	Delay caused by the late aircraft
Unnamed: 27	Useless column

3.BUSINESS UNDERSTANDING

There are some factors, issues, and events that cannot be anticipated and aviation organizations must ensure they are flexible and responsive to deal with them effectively. The airline business is very unique. The product offered by airlines is represented by flights that carry passengers or cargo from various origins to various targeted destinations.

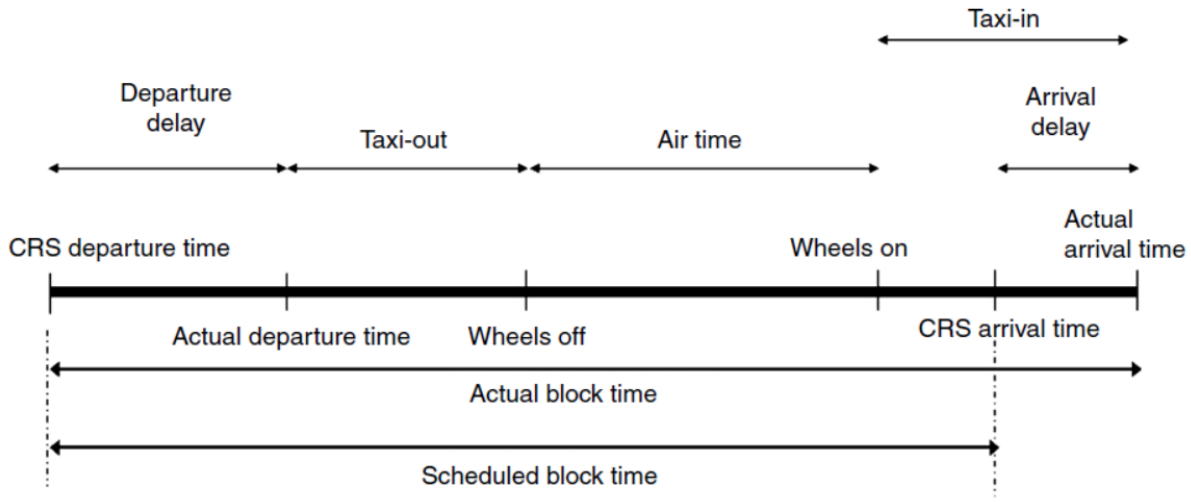


Fig3.1: Main Segments of Air travel time

[2]Airlines track and report five segments of travel time for each of their flights to the FAA: (1) departure delay, (2) taxi-out, (3) air time, (4) taxi-in, and (5) arrival delay. Figure 2 displays this segmentation. We use this information (publicly available through the BTS) to determine the scheduled block time of each flight. Based on Figure 2, we first define different terms that constitute the travel time segments: computerized reservation system (CRS) departure/arrival time is the scheduled departure/arrival time of the flight, wheels off is the time when the wheels of the aircraft leave the ground at the origin airport, and wheels on is the time when the wheels of the aircraft touch the ground at the destination airport. The departure delay of an aircraft is the difference between the actual departure time and the CRS departure time of the flight. Arrival delay equals actual arrival time minus the scheduled arrival time.

[2]The airline scheduler has to pick a scheduled block time (*Q*) for a flight, whereas the actual travel/block time (*D*) for the flight is uncertain. The scheduled block-time decision and the actual block time is defined as scheduled block time _*Q*_

=CRS arrival time−CRS departure time_ (1)

actual block time _*D*_

=actual arrival time−CRS departure time_ (2)

[2]From the above definitions, it is clear that if a flight's actual block time exceeds its scheduled block time, then the flight is delayed, whereas if its actual block time falls short of its scheduled block time, then the flight arrives early. For any scheduled flight *i*, airlines first estimate the actual block-time distribution by analysing historical (typically over the previous two to three years) travel time data. It is important to note that airline schedulers typically remove late aircraft delay (LAD) from the actual block time when estimating the distribution of the actual block time. LAD is the portion of departure delay attributed to an aircraft arriving late from its earlier flight, and is tracked in the BTS on-time data set for every flight. Thus, truncated block time, which is used for making scheduled block-time decisions, is defined as follows:

truncated block time TD

=actual block time D

−late aircraft delay LAD (3)

[2] The reason airlines typically remove LAD from the actual block time is that the scheduled block-time decisions are made in the schedule generation phase (phase 2 of the schedule development described above) at which point the aircraft rotation (i.e., the sequence of flights operated by a particular aircraft on the same day) is unknown. Hence, it is difficult for an airline scheduler to estimate the actual block time for a flight, which includes LAD. Another advantage of using truncated block time instead of actual block time is that it reduces the error term correlation between flights flown by the same aircraft in a given rotation, which may be induced by late aircraft delays. Finally, from the perspective of our research questions, it makes no difference to exclude LAD from the block time. This is because all flights look similar in terms of LAD due to unknown rotations, and our objective is to explore the drivers of heterogeneity in scheduling decisions.

There are five broad categories that the airlines officially categorize a delay;

- **Air Carrier Delay:** The reason for the delay was in the airline's control.
- **Security:** Security issues such as when required to evacuate the airline or even the airport premises.
- **Extreme Weather:** Snow, thunderstorms, fog, high winds, etc.
- **Late-arriving traffic:** Due to the late arrival of the previous flight (same aircraft), the present one departs late.
- **National Aviation System (NAS):** Heavy air traffic, traffic control, airport operations are monitored by the NAS, which can sometimes delay flights.

4. EXPLORATORY DATA ANALYSIS

Bivariate/Multivariate analysis:

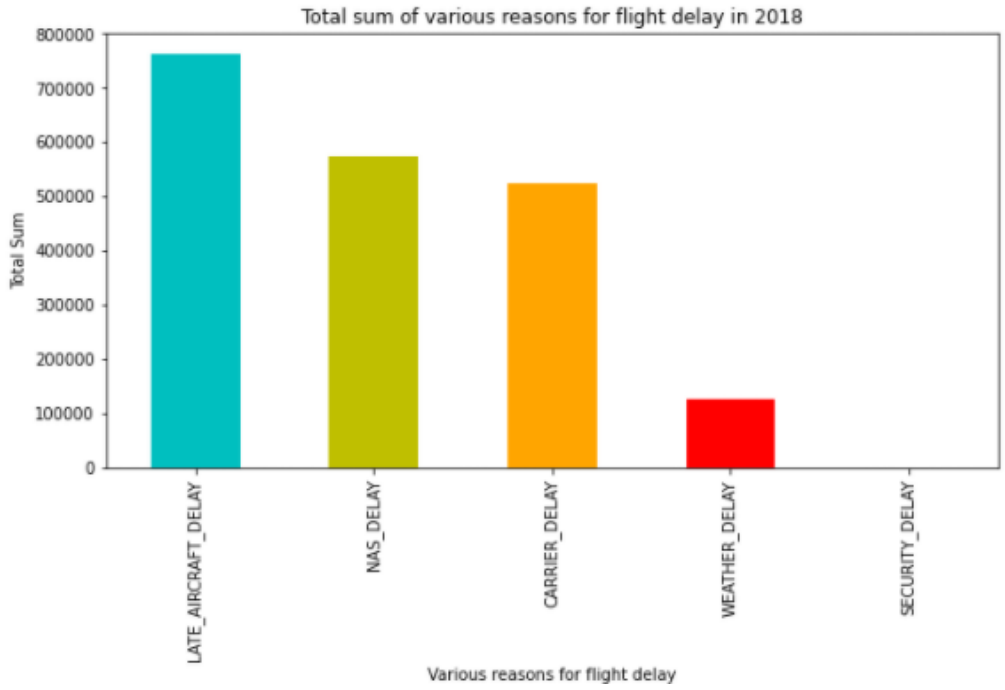


Fig 4.1: This bar graph shows the total time duration of the various delays in the flight and it is found that the late aircraft delays has the maximum time duration with a total of approximately 750000 minutes in the year.

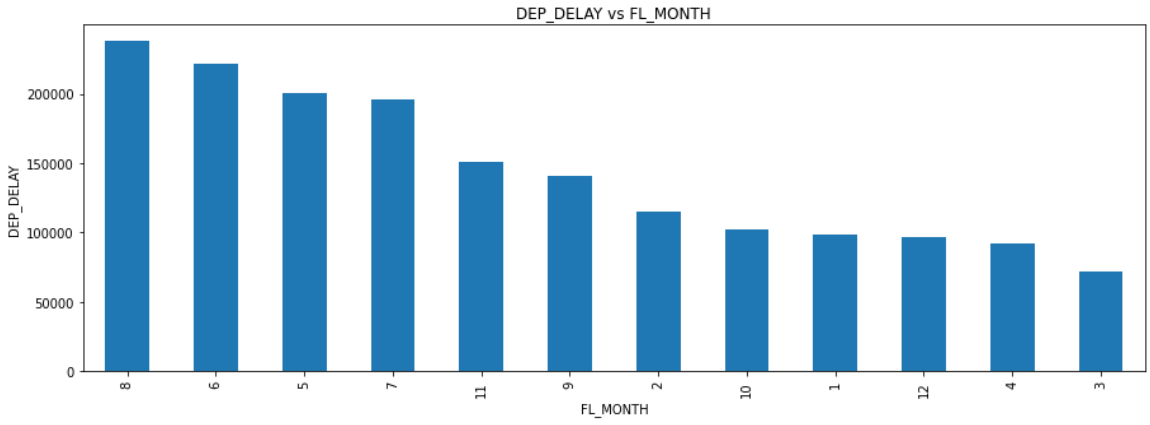


Fig 4.2: This bar graph shows the total duration of DEP delay in each month and is found that the maximum time duration of the DEP delay have occurred in August month and lowest in the March.

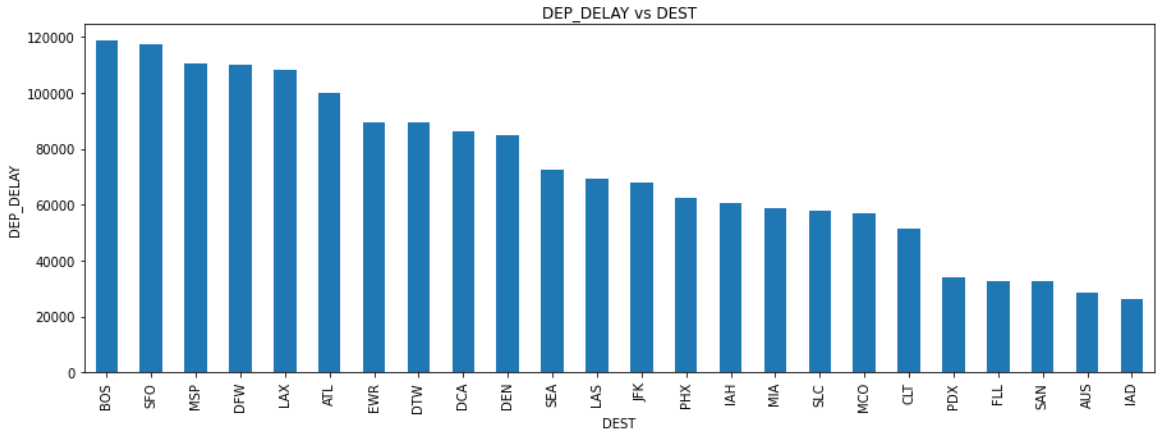


Fig 4.3: This bar graph shows the total duration of DEP delay for each location and is found that the maximum time duration of the DEP delay have occurred in Boston(BOS) and lowest in Dulles(IAD).

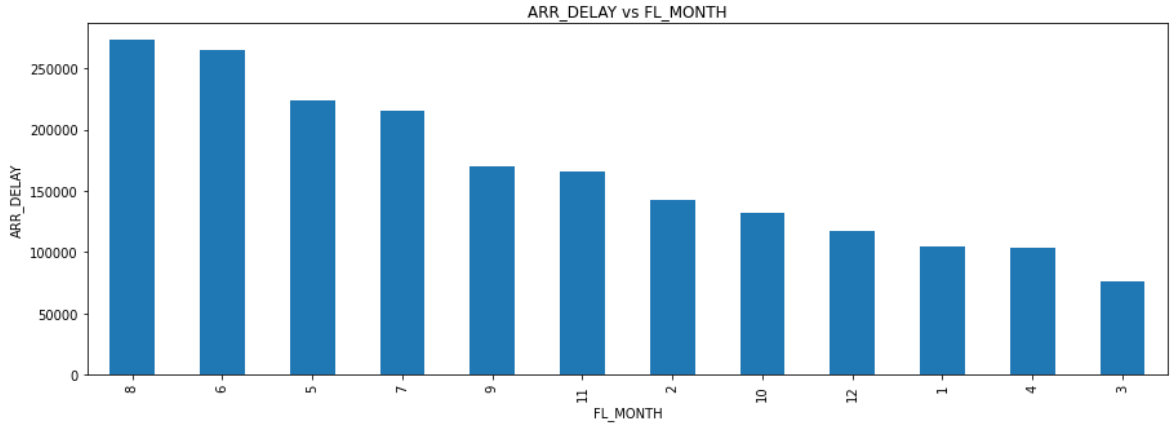


Fig 4.4: This bar graph shows the total duration of ARR delay in each month and is found that the maximum time duration of the ARR delay have occurred in August month and lowest in the March.

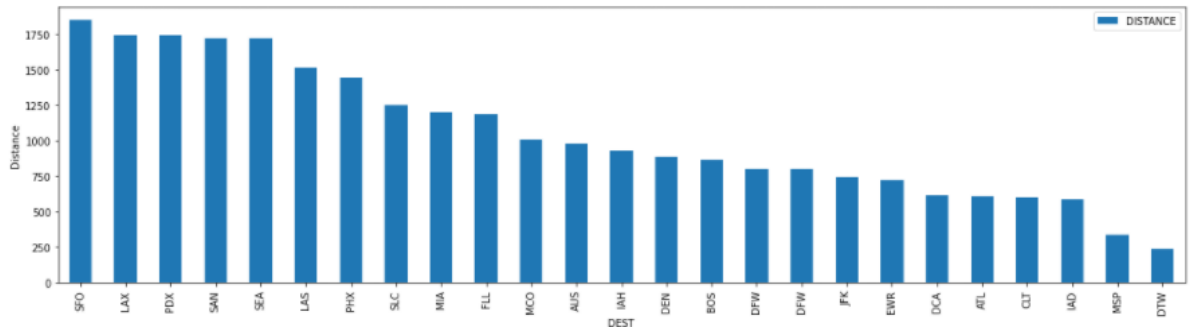


Fig 4.5: In this bar graph, we have picked multiple columns with different datatypes, and in order to get the unique values we used dropping duplicates and then created a data frame by assigning a variable.

Hence, we kept 'DEST' column as index, then used sort_values function to sort in descending order to get a bar output.

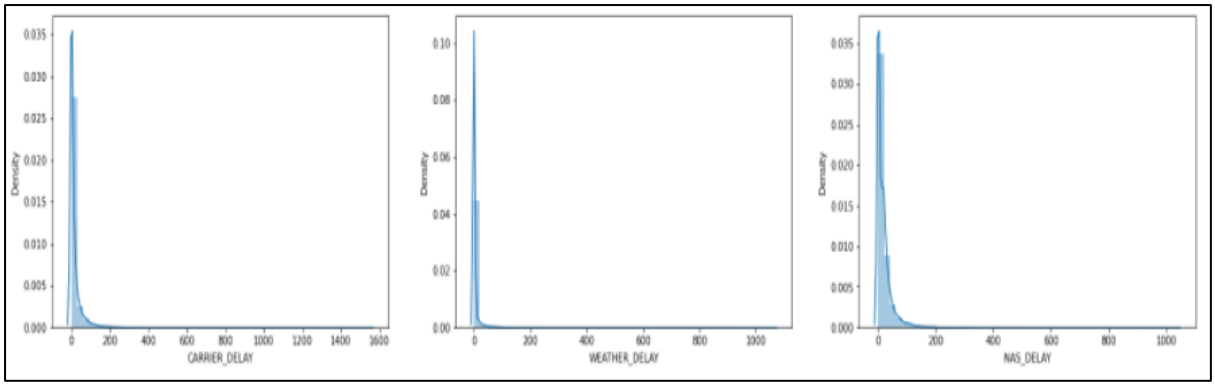


Fig 4.6: This graph checks the time duration of various delays and it is found that neither of the three delays have extended more than 2 hours

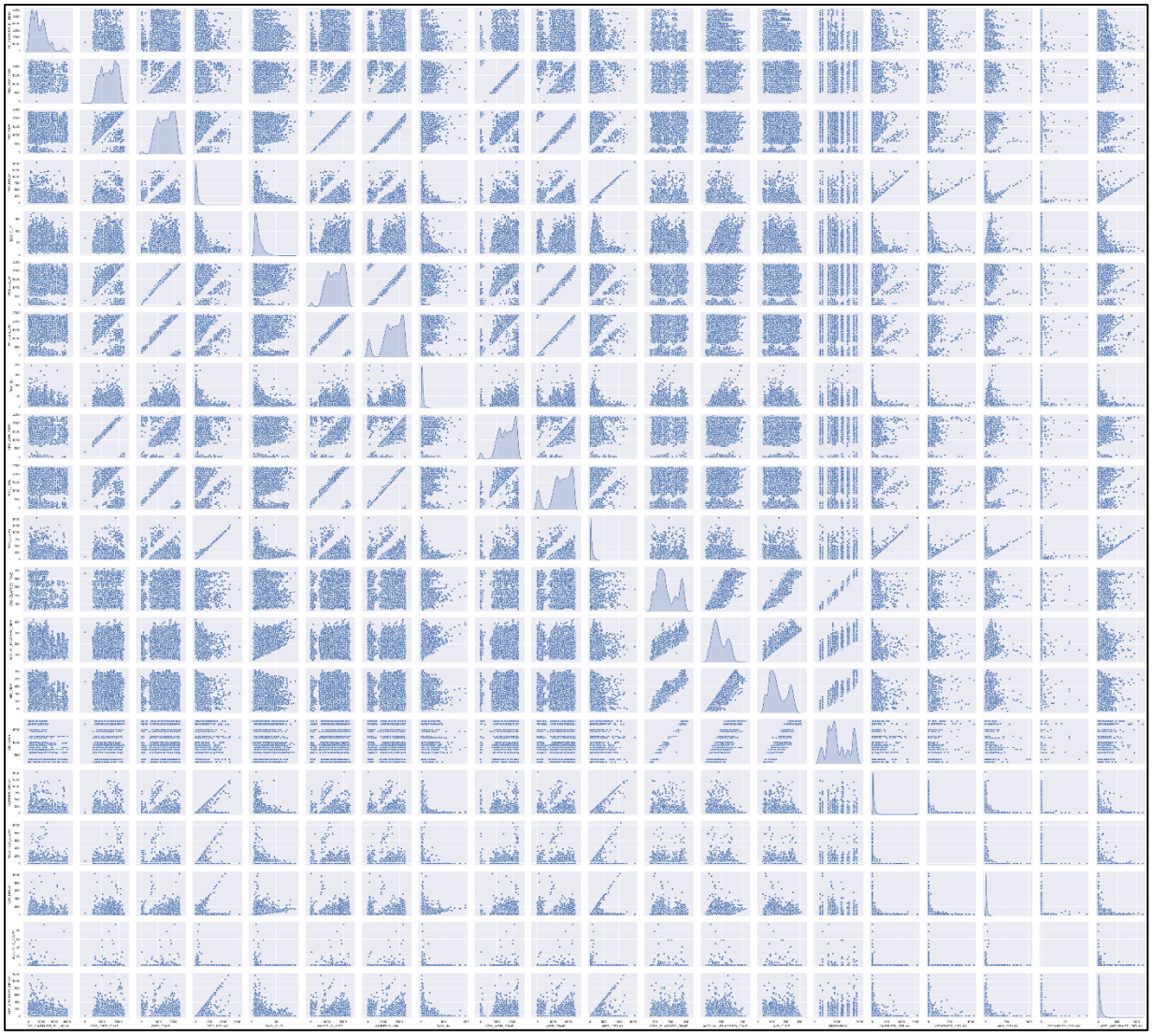


Fig 4.7 Analysing the scatterplot, it doesn't seem to be a significant correlation between the types of delay themselves. A more in-depth analysis, following the exact routes of the aircrafts and their connections and subsequent flights could help identifying the exact root cause to each delay, but that are certain linear relationship within the scope of this analysis.

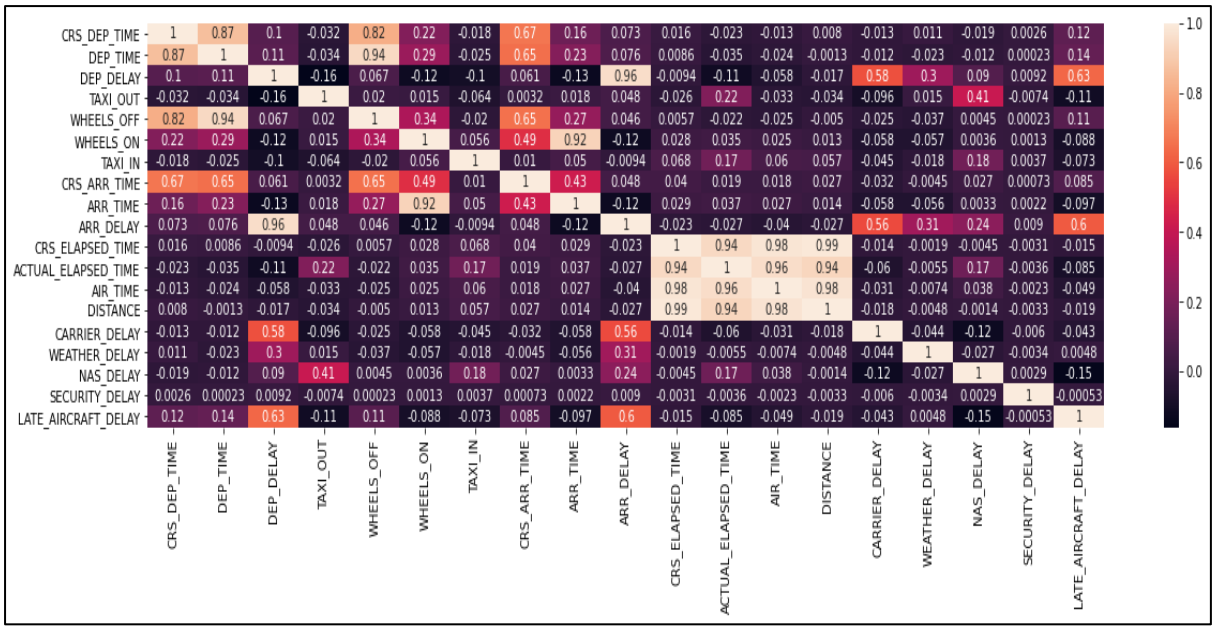


Fig 4.8: In the above heatmap, the Distance, Air_time, CRS_Elapsed_time and Actual_Elapsed_time are highly correlated to each other. The wheels off is related to departure time and the wheel on is related to the arrival time.

Univariate Analysis:

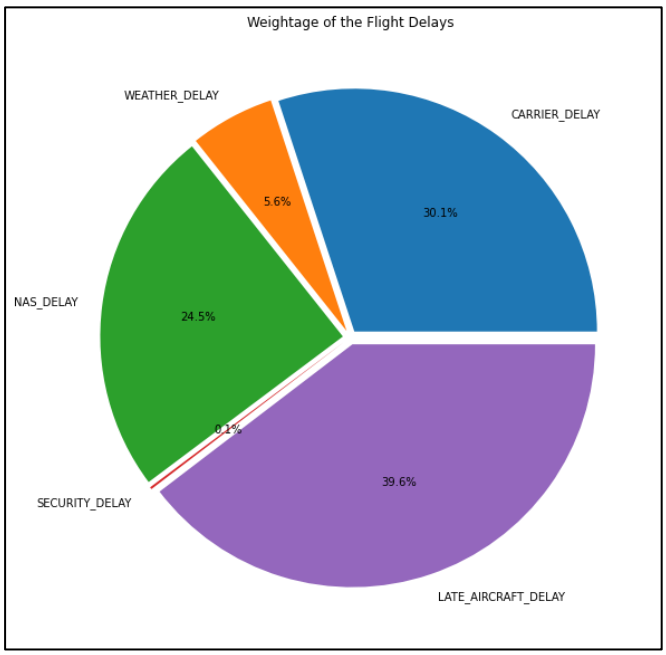


Fig 4.9: The Late_Aircraft_delay accounts to the major delays in 2018 which is 39.6%,followed by carrier delay that is 30.1%. Security delay is the lowest with 0.1%.

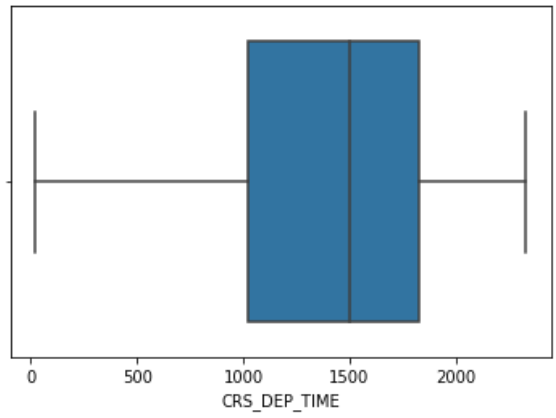


Fig 4.10:Boxplot of CRS_DEP_TIME

- The majority of flight departure is scheduled at 3 pm.

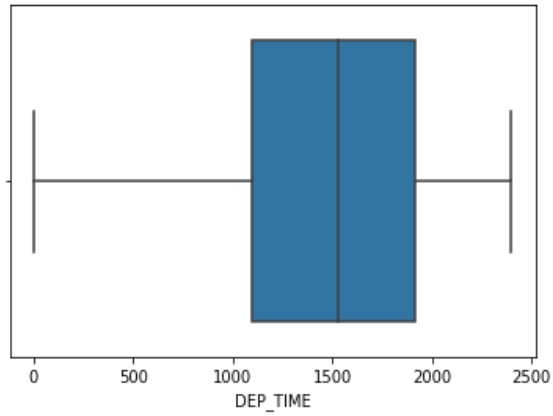


Fig 4.11:Boxplot of DEP_TIME

- The actual departure time varied from the scheduled departure time.
- The median timing is somehow around 3:00 pm, which signifies most of the flights were departure at the right time

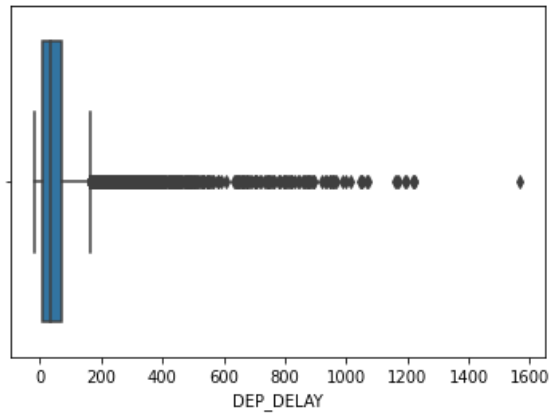


Fig 4.12:Boxplot of DEP_DELAY

- The departure delay shows the outlier since the value would differ, because the on-time flight would be considered as 0. This will consider the delay as outliers.

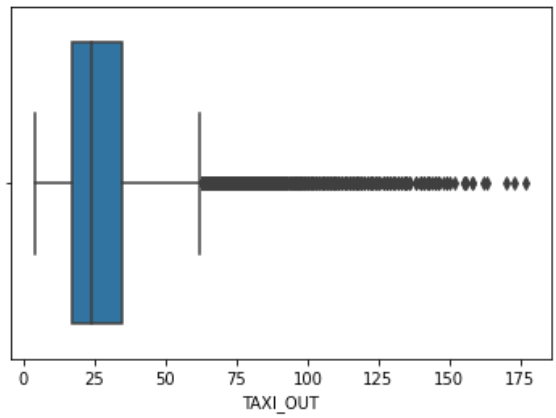


Fig 4.13:Boxplot of TAXI_OUT

- The median value is around 25 minutes.
- The outlier could be the effect of the delays

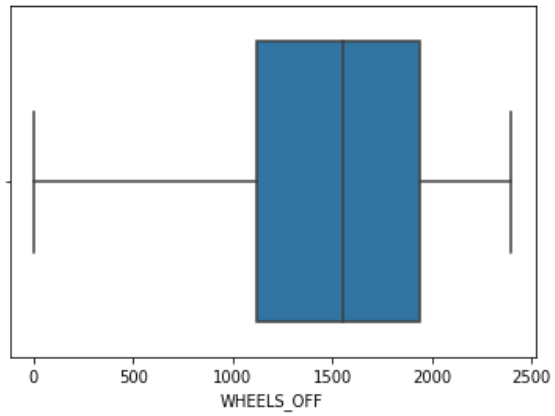


Fig 4.14:Boxplot of WHEELS_OFF

- The time will be related with the actual departure time of the flight from the airport.

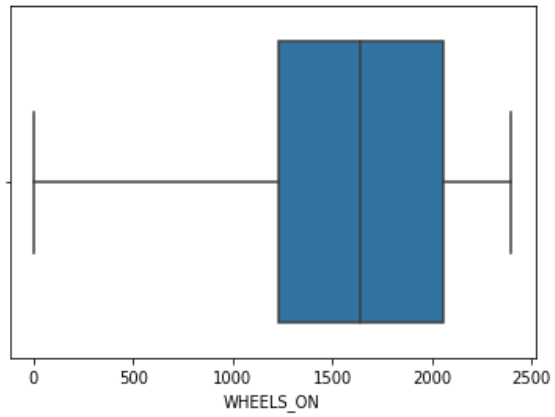


Fig 4.15:Boxplot of WHEELS_ON

- The time will be related with the actual arrival time of the flight from the airport.

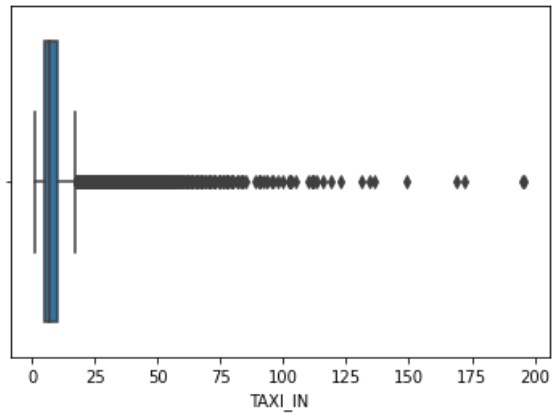


Fig 4.16:Boxplot of TAXI_IN

- The median value is around 5minutes.
- The outlier could be the effect of the delays

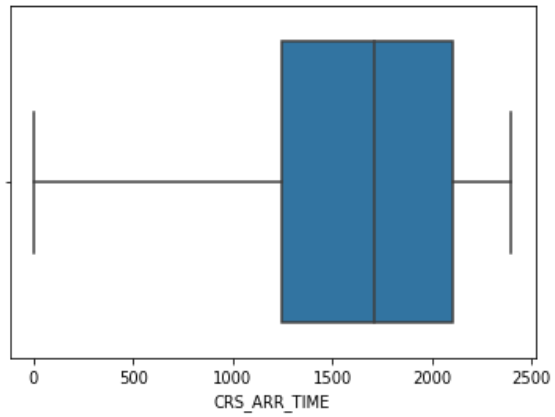


Fig 4.17:Boxplot of CRS_ARR_TIME

- The median time is around 7:00 pm.

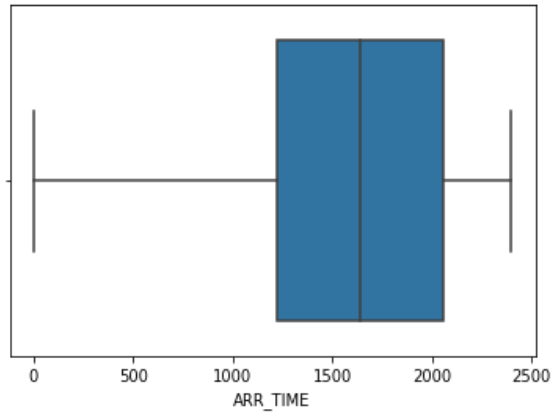


Fig 4.18:Boxplot of ARR_TIME

- The actual arrival time varied from the scheduled departure time at few data points or instances.

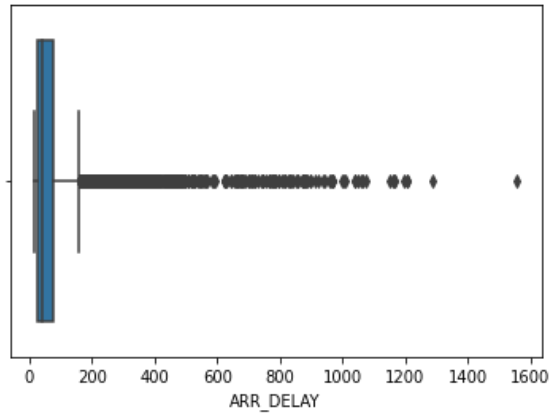


Fig 4.19:Boxplot of ARR_DELAY

- The arrival delay shows the outlier since the value would differ, because the on-time flight would be considered as 0. This will consider the delay as outliers.

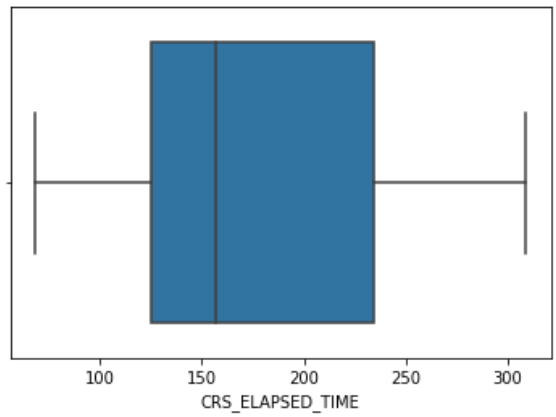


Fig 4.20:Boxplot of CRS_ELAPSED_TIME

- The median value for the total planned time amount of the flights is at 160 minutes.

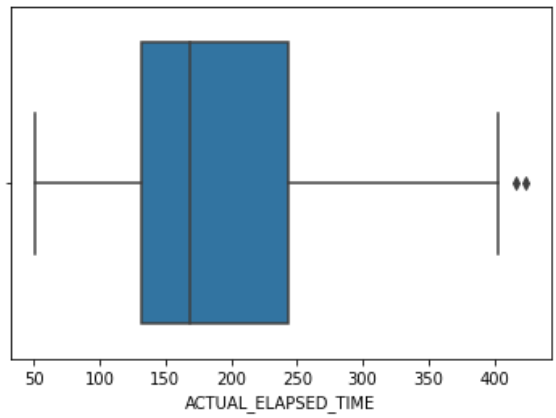


Fig 4.21:Boxplot of ACTUAL_ELAPSED_TIME

- The median value for the actual planned time amount of the flights is nearby to the planned or expected elapsed time.
- But there are few outliers which would also account to the reason for the delay of the flight.

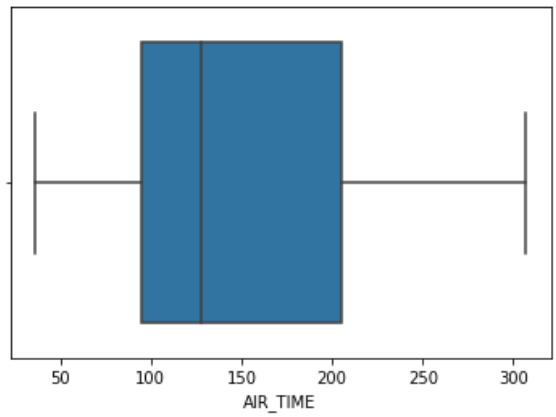


Fig 4.22:Boxplot of AIR_TIME

- The minimum airtime would be from 50 and the maximum airtime would be around 300.
- The median value of air time is 130.

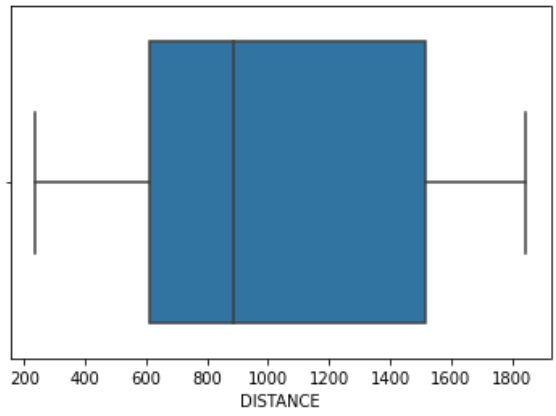


Fig 4.23:Boxplot of DISTANCE

- The minimum distance range is from 200 miles and the maximum distance about 1800 miles.
- The median distance is around 900 miles.

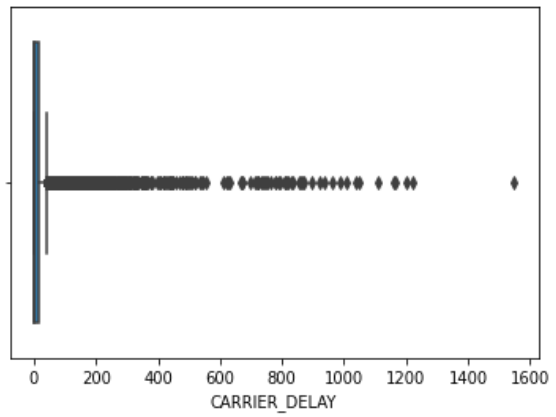


Fig 4.24:Boxplot of CARRIER_DELAY

- The outlier is the delay of caused due to carrier air control.

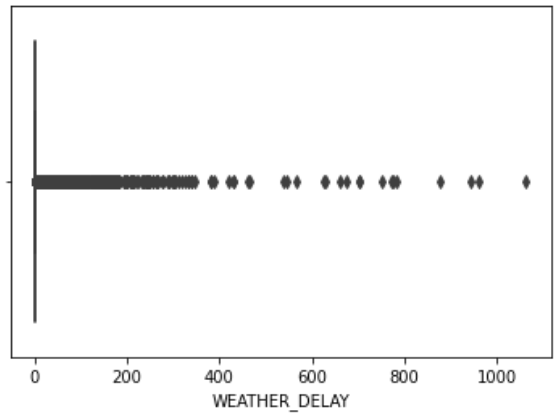


Fig 4.25:Boxplot of WEATHER_DELAY

- The outlier is the delay of caused due to weather condition

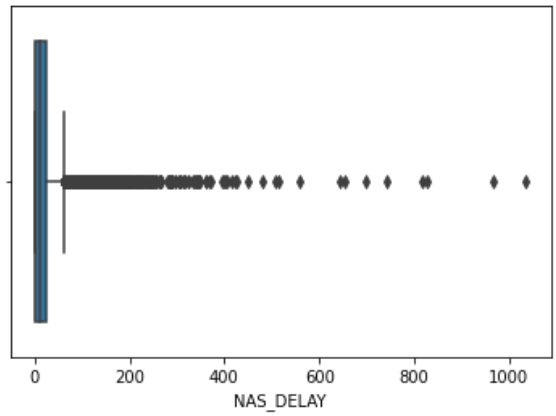


Fig 4.26:Boxplot of NAS_DELAY

- The outlier is the delay of caused due to National Aviation System faults or problems

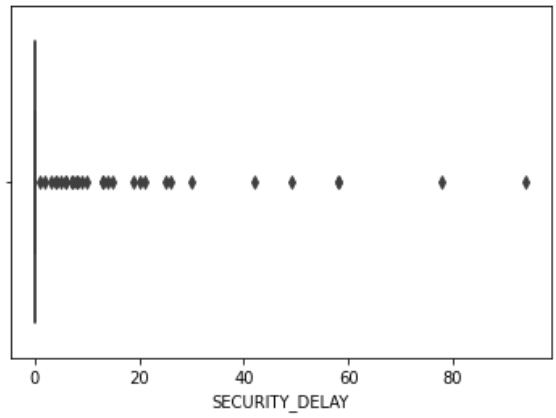


Fig 4.27:Boxplot of SECURITY_DELAY

- The outlier is the delay of caused due to security concern or issues

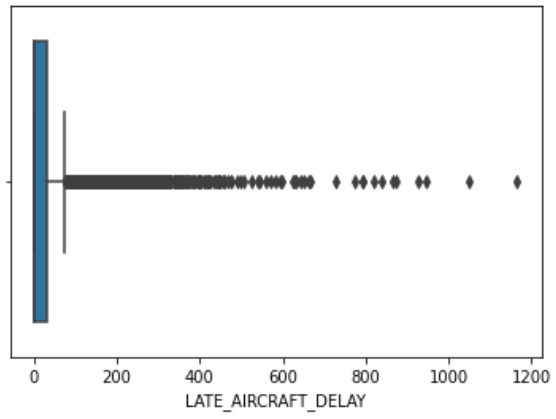


Fig 4.28:Boxplot of LATE_AIRCRAFT_DELAY

- The outlier is the delay of caused due to late arrival of previous aircraft

Winter weather is one of the most reliable heralds of flight delays. And Avery, the first winter storm of 2018, was no exception. Weather is one cause of flight delays categorized by the U.S. Department of Transportation's Bureau of Transportation Statistics. It's part of a broader set of conditions known as the National Aviation System which also includes airport operations, heavy traffic volume, and air traffic control. Extreme weather is in its own category; it covers real and forecasted blizzards, tornadoes, and hurricanes.

Delays also are caused by aircraft carriers and include problems with maintenance and crews, baggage loading, fuelling, and aircraft cleaning. Security and safety issues that many people complain about—waiting in security lines for more than 29 minutes, broken equipment, and evacuation of aircraft—are others. Late-arriving aircraft scheduled for a trip later is the last type of delay the government tracks.

O’Hare Airport was the most-delayed airport in the United States, according to new rankings.

A report from Finance Buzz looked at data from the U.S. Department of Transportation to rank the country’s 25 most-delayed airports in 2018, O’Hare was the worst.

In all, passengers at O’Hare were delayed for a combined 1,133 years just in 2018, according to the report. That’s 9,925,080 hours.

About a quarter of flights were delayed or cancelled at O’Hare Airport, according to the report, with the average delayed departure taking 72 minutes.

About 37 percent of O’Hare’s delays were attributed to National Aviation System delays, which can refer to “non-extreme weather conditions, airport operations, heavy traffic volume and traffic control,” according to the report. Just 3 percent of the airport’s delays were caused by weather.

5. HYPOTHESIS TESTING

- This section deals with the hypothesis testing and we will be explaining here the hypothesis tests and the values we got to conclude on the assumption that the arrival and delay time of the flights at the USA airports is same.
- The t-test is used best for the sample of data which is unbiased and is normally distributed.
- The data collected here for the completion of this project is basically from the open source Kaggle and the source of this dataset is basically department of air transportation USA.
- It is assumed that the data collected is unbiased as for the departure and arrival delay in the flights of plane at different airports there in USA.
- In order to check our assumption of the dataset is homogenous or not we performed levene test prior to the t-test.
- Levene test is used to find the homogeneity of variance between the samples which are under observation (Here the samples are flight departure and arrival delay.)
- It works in such a way that it computes the absolute means of the samples under observation.

HYPOTHESIS:

* Null hypothesis(H0): The average mean of the two samples (arrival and departure delay timing) is zero.

* Alternate hypothesis (Ha): The average mean of the two samples (arrival and departure delay timing) is not zero.

```
In [153]:  1  stats.levene(df1['DEP_DELAY'],df1['ARR_DELAY'],center='mean')
Out[153]: LeveneResult(statistic=42.75025580377444, pvalue=6.26793884957566e-11)
```

- As this dataset has huge number of entries, so by doing the levene test we got a significance of 6.26e-11.
- After statistically analysing our samples, we saw the p-value which is less than alpha 0.05 so we reject the null hypothesis.
- Here for this project, we are using the significance level of 0.05 and the value we got after performing the levene test is very small or less than 0.05 so it says that the variance cannot be assumed to be equal.

```
In [154]:  1  stats.ttest_rel(df1['DEP_DELAY'],df1['ARR_DELAY'])
Out[154]: Ttest_relResult(statistic=-67.51606206440103, pvalue=0.0)
```

- After completing hypothesis testing which includes levene test and t-test on the sample data under observation, we got enough statistical evidence to reject the null hypothesis which was that the arrival and departure flight timing are same in a course of one year.

6. FEATURE ENGINEERING

The objective of feature engineering is to create new features (also called explanatory variables or predictors) to represent as much information from an entire dataset in one table. Typically, this process is done by hand using pandas' operations such as groupby, agg, or merge and can be very tedious. Moreover, manual feature engineering is limited both by human time constraints and imagination: we simply cannot conceive of every possible feature that will be useful.

The importance of creating the proper features cannot be overstated because a machine learning model can only learn from the data, we give to it. Extracting as much information as possible from the available datasets is crucial to creating an effective solution.

It's a good practice to create a copy of the original dataset in order to retrieve the details form the original dataset. While doing the feature engineering, certain details might be changed/removed. It's better to work on the copied dataset.

```
In [47]: 1 df_dummy=df1.copy()
         2 df_dummy.head()
```

Out[47]:

	FL_DATE	OP_CARRIER	OP_CARRIER_FL_NUM	ORIGIN	DEST	CRS_DEP_TIME	DEP_TIME	DEP_DELAY	TAXI_OUT	WHEELS_OFF	...	ARR_DELAY	CRS
0	2018-01-01	UA	2106	ORD	FLL	1350	1512.0	82.0	12.0	1524.0	...	72.0	
1	2018-01-01	UA	2072	ORD	FLL	1935	2020.0	45.0	17.0	2037.0	...	31.0	
2	2018-01-01	UA	2001	ORD	LAX	900	1044.0	104.0	16.0	1100.0	...	75.0	
3	2018-01-01	UA	1988	ORD	SFO	705	754.0	49.0	14.0	808.0	...	57.0	
4	2018-01-01	UA	1936	ORD	DTW	1345	1413.0	28.0	24.0	1437.0	...	24.0	

5 rows × 24 columns

The flight date is in datetime format, which would be problem while applying the technique during machine learning concepts. We need the month which could be helpful in analysing its effect on the other features.

```
In [48]: 1 df_dummy["MONTH"]=df_dummy["FL_DATE"].dt.month
         2 df_dummy.head()
```

Out[48]:

	FL_DATE	OP_CARRIER	OP_CARRIER_FL_NUM	ORIGIN	DEST	CRS_DEP_TIME	DEP_TIME	DEP_DELAY	TAXI_OUT	WHEELS_OFF	...	CRS_ELAPSED_TI
0	2018-01-01	UA	2106	ORD	FLL	1350	1512.0	82.0	12.0	1524.0	...	18
1	2018-01-01	UA	2072	ORD	FLL	1935	2020.0	45.0	17.0	2037.0	...	18
2	2018-01-01	UA	2001	ORD	LAX	900	1044.0	104.0	16.0	1100.0	...	28
3	2018-01-01	UA	1988	ORD	SFO	705	754.0	49.0	14.0	808.0	...	28
4	2018-01-01	UA	1936	ORD	DTW	1345	1413.0	28.0	24.0	1437.0	...	8

5 rows × 25 columns

In machine learning, we usually deal with a variety of labels. These can be either numbers or words. If they are numbers, then the algorithm can use them directly. However, labels often need to be in a human-readable form. So, people usually label the training data with words.

Label encoding refers to transforming word labels into a numerical form so that algorithms can understand how to operate on them. we used the *preprocessing.LabelEncoder()*

```
In [50]:
from sklearn import preprocessing
label_encoder = preprocessing.LabelEncoder()
df_dummy['DEST']= label_encoder.fit_transform(df1['DEST'])
df_dummy['ORIGIN']= label_encoder.fit_transform(df1['ORIGIN'])
df_dummy["OP_CARRIER"]=label_encoder.fit_transform(df1["OP_CARRIER"])

In [51]:
df_dummy["OP_CARRIER"].unique()

Out[51]:
array([11,  0,  3,  6,  7,  8, 10, 13,  1,  4, 12,  9,  2,  5])

In [52]:
df_dummy.head()

Out[52]:
```

	FL_DATE	OP_CARRIER	OP_CARRIER_FL_NUM	ORIGIN	DEST	CRS_DEP_TIME	DEP_TIME	DEP_DELAY	TAXI_OUT	WHEELS_OFF	...	CRS_ELAPSED_TIM
0	2018-01-01	11	2106	0	9	1350	1512.0	82.0	12.0	1524.0	...	188
1	2018-01-01	11	2072	0	9	1935	2020.0	45.0	17.0	2037.0	...	188
2	2018-01-01	11	2001	0	14	900	1044.0	104.0	16.0	1100.0	...	287
3	2018-01-01	11	1988	0	22	705	754.0	49.0	14.0	808.0	...	285
4	2018-01-01	11	1936	0	7	1345	1413.0	28.0	24.0	1437.0	...	80

5 rows × 25 columns

function to transform word labels into numerical form.

Label encoding can transform categorical data into numeric data, but the imposed ordinality creates problems if the obtained values are submitted to mathematical operations. One-hot encoding has the advantage that the result is binary rather than ordinal, and that everything is in an orthogonal vector space. The disadvantage is that for high cardinality, the feature space can explode.

```
In [53]:
len(df_dummy['OP_CARRIER_FL_NUM'].unique())

Out[53]:
1323

In [54]:
df_dummy.drop("FL_DATE",axis=1,inplace=True)

In [55]:
df_dummy.drop("OP_CARRIER_FL_NUM",axis=1,inplace=True)

In [56]:
df_dummy.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30718 entries, 0 to 30717
Data columns (total 23 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   OP_CARRIER            30718 non-null  int32
 1   ORIGIN                30718 non-null  int32
 2   DEST                  30718 non-null  int32
 3   CRS_DEP_TIME          30718 non-null  int64
 4   DEP_TIME              30718 non-null  float64
 5   DEP_DELAY             30718 non-null  float64
 6   TAXI_OUT              30718 non-null  float64
 7   WHEELS_OFF            30718 non-null  float64
 8   WHEELS_ON             30718 non-null  float64
 9   TAXI_IN               30718 non-null  float64
 10  CRS_ARR_TIME          30718 non-null  int64
 11  ARR_TIME              30718 non-null  float64
 12  ARR_DELAY             30718 non-null  float64
 13  CRS_ELAPSED_TIME      30718 non-null  float64
 14  ACTUAL_ELAPSED_TIME   30718 non-null  float64
 15  AIR_TIME              30718 non-null  float64
 16  DISTANCE              30718 non-null  float64
 17  CARRIER_DELAY         30718 non-null  float64
 18  WEATHER_DELAY         30718 non-null  float64
 19  NAS_DELAY             30718 non-null  float64
 20  SECURITY_DELAY        30718 non-null  float64
 21  LATE_AIRCRAFT_DELAY   30718 non-null  float64
 22  MONTH                 30718 non-null  int64
dtypes: float64(17), int32(3), int64(3)
memory usage: 5.0 MB
```

The dropping columns based on the feature importance might get rid of collinear features.

Getting rid of collinear features will increase the weight of their collinear counterparts which are left in the model. Since both sets of features were giving similar signals, this was causing weights on that direction to be divided among them.

Here in this dataset the OP_CARRIER_FL_NUM and FL_DATE was dropped, since they had quite a lot unique value.

7. PREPROCESSING & MACHINE LEARNING TECHNIQUES

Unsupervised learning involves creating a model that is able to extract patterns from unlabelled data. In other words, the computer analyses the input features and determines for itself what the most important features and patterns are. Unsupervised learning tries to find the inherent similarities between different instances. If a supervised learning algorithm aims to place data points into known classes, unsupervised learning algorithms will examine the features common to the object instances and place them into groups based on these features, essentially creating its own classes.

In [59]:
```
from sklearn.cluster import KMeans, AgglomerativeClustering
from scipy.cluster.hierarchy import cophenet,dendrogram,linkage
from sklearn.preprocessing import PowerTransformer
from sklearn.decomposition import PCA
from sklearn.experimental import enable_iterative_imputer
from sklearn.impute import IterativeImputer
from sklearn.metrics import silhouette_score
```

All necessary modules are imported from the respective libraries, in order to apply and execute the required methods/steps for obtaining the consecutive results.

Standardizing a dataset involves rescaling the distribution of values so that the mean of observed values is 0 and the standard deviation is 1. This can be thought of as subtracting the mean value or centring the data.

Like normalization, standardization can be useful, and even required in some machine learning algorithms when your data has input values with differing scales.

Standardization assumes that your observations fit a Gaussian distribution (bell curve) with a well-behaved mean and standard deviation. You can still standardize your data if this expectation is not met, but you may not get reliable results.

7.1 Preprocessing using StandardScaler:

In [57]:
```
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
df_scale=pd.DataFrame(sc.fit_transform(df_dummy),columns=df_dummy.columns)b
```

In [78]:
```
df_scale.head()
```

Out[78]:

	OP_CARRIER	ORIGIN	DEST	CRS_DEP_TIME	DEP_TIME	DEP_DELAY	TAXI_OUT	WHEELS_OFF	WHEELS_ON	TAXI_IN	...	CRS_ELAPSED_TIME	A
0	1.038692	0.0	-0.291205	-0.186816	0.035782	0.327585	-0.970631	-0.011577	0.548216	-0.169984	...	0.205596	
1	1.038692	0.0	-0.291205	1.101675	1.070021	-0.141923	-0.680615	1.015701	-2.440067	-0.396197	...	0.221360	
2	1.038692	0.0	0.418273	-1.177963	-0.917021	0.606753	-0.738619	-0.860634	-0.473136	-0.283091	...	1.766222	
3	1.038692	0.0	1.553437	-1.607459	-1.507433	-0.091165	-0.854625	-1.445362	-0.840949	1.752829	...	1.734694	
4	1.038692	0.0	-0.574996	-0.197829	-0.165772	-0.357643	-0.274593	-0.185794	0.106209	-0.283091	...	-1.496905	

5 rows × 23 columns

StandardScaler is used to perform Feature Scaling. Feature Scaling is a phase in Data preprocessing. Basically, we use Standard Scalar in order to scale the magnitude of the feature in a certain range. Generally, what data we get from the real world, they have a great difference between them and that have direct impact over the performance of the model. So,

it's always a best practice to scale the data before processing it. There are various techniques in Unsupervised Learning. We are primarily going to use two of them;

1. KMean Clustering (Top to bottom approach)
2. Agglomerative Clustering (Bottom to top approach)

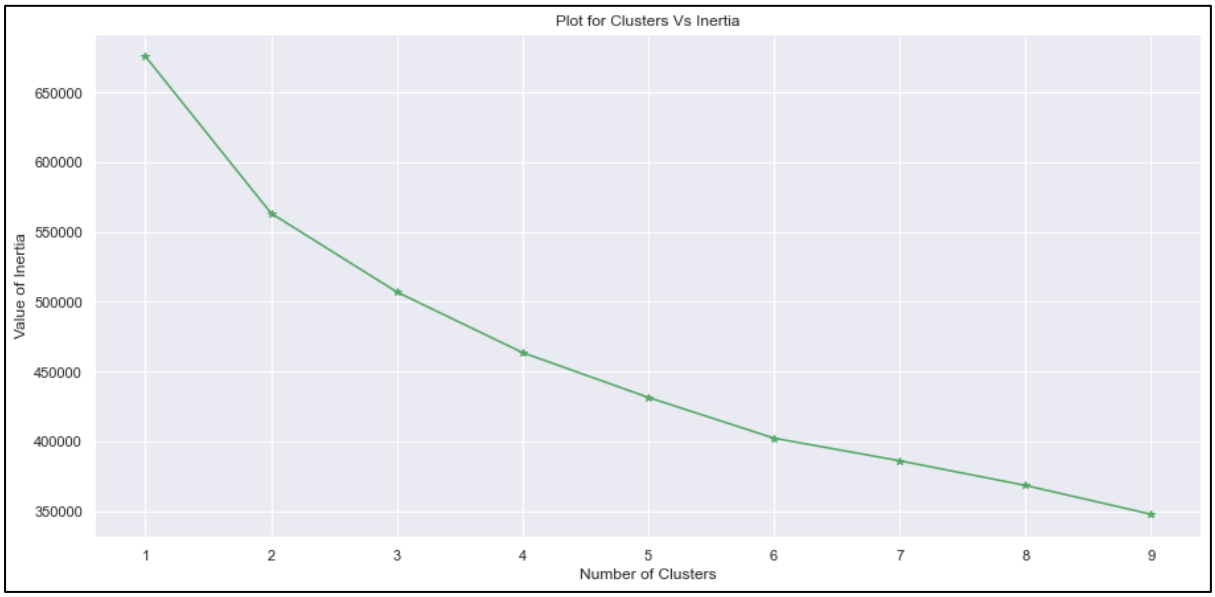
7.1.1 KMeans clustering:

In general, k-means is the first choice for clustering because of its simplicity. Here, the user has to define the number of clusters (Post on how to decide the number of clusters would be dealt later). The clusters are formed based on the closeness to the center value of the clusters.

```
In [62]:
for k in range(2,10):
    kmean1 = KMeans(n_clusters=k,random_state=10)
    kmean1.fit(df_scale)
    print(f'Cluster NUmber K = {k}: Silhoutte Score = {silhouette_score(df_scale,labels=kmean1.labels_)}')
```

```
Cluster NUmber K = 2: Silhoutte Score = 0.2121272955544751
Cluster NUmber K = 3: Silhoutte Score = 0.17794238968905113
Cluster NUmber K = 4: Silhoutte Score = 0.1835885367730882
Cluster NUmber K = 5: Silhoutte Score = 0.1985417149729528
Cluster NUmber K = 6: Silhoutte Score = 0.1910823015867603
Cluster NUmber K = 7: Silhoutte Score = 0.1917733163529539
Cluster NUmber K = 8: Silhoutte Score = 0.19343814660860112
Cluster NUmber K = 9: Silhoutte Score = 0.19549373668990178
```

The initial center value is chosen randomly. K-means clustering is top-down approach, in the sense, we decide the number of clusters (k) and then group the data points into k clusters.



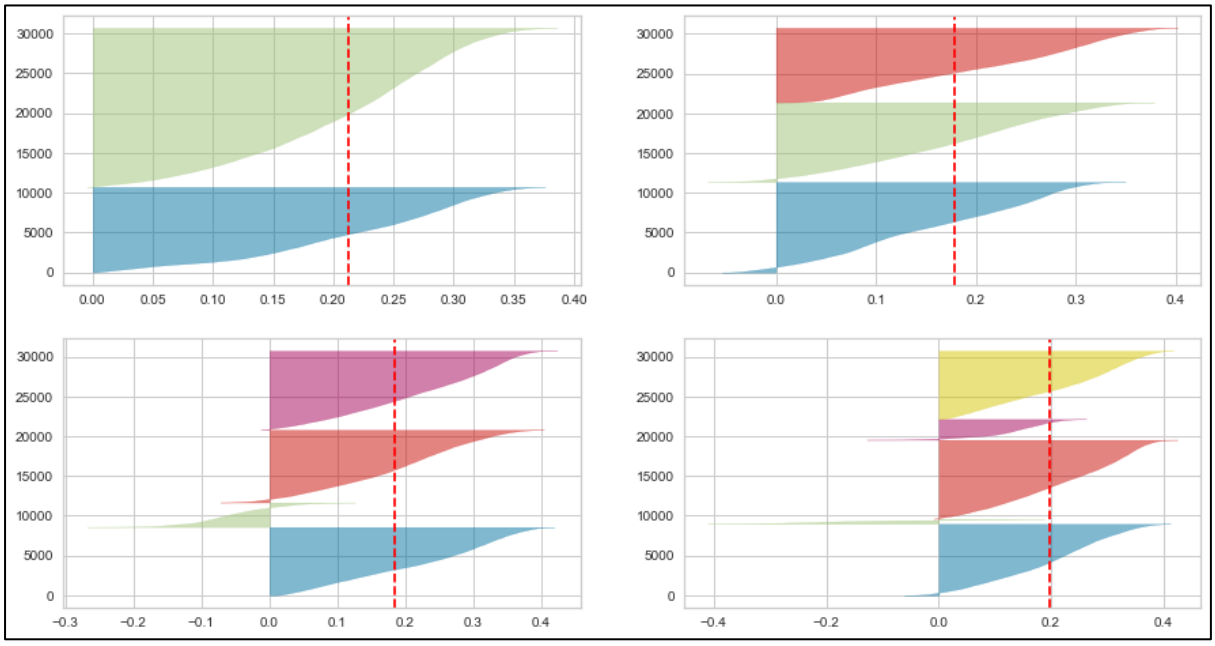
Silhouette score for a set of sample data points is used to measure how dense and well-separated the clusters are.

Silhouette score takes into consideration the intra-cluster distance between the sample and other data points within same cluster (a) and inter-cluster distance between sample and next nearest cluster (b).

The silhouette score falls within the range [-1, 1]. The silhouette score of 1 means that the clusters are very dense and nicely separated. The score of 0 means that clusters are overlapping. The score less than 0 means that data belonging to clusters may be wrong / incorrect.

The silhouette plots can be used to select the most optimal value of the K (no. of cluster) in K-means clustering.

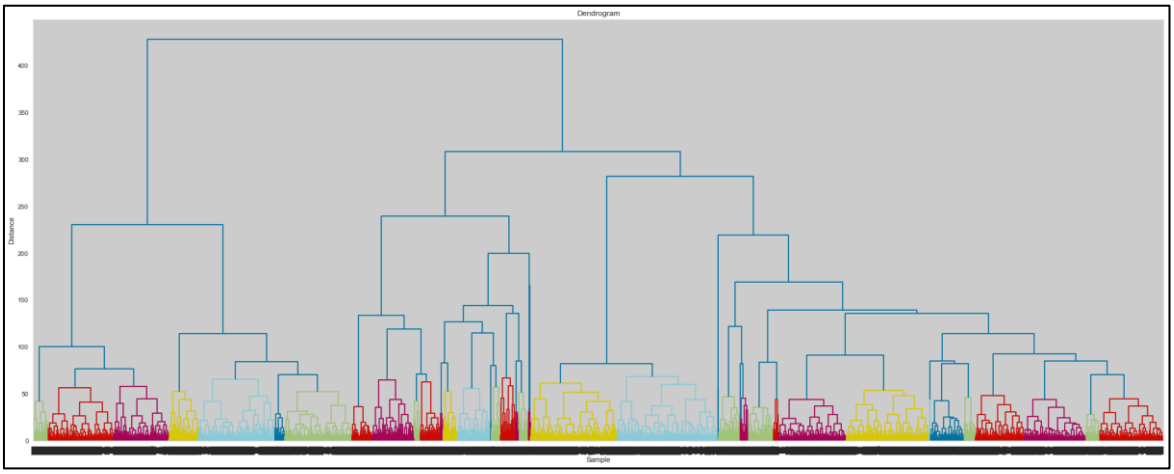
The aspects to look out for in Silhouette plots are cluster scores below the average silhouette score, wide fluctuations in the size of the clusters and also the thickness of the silhouette plot.



```
In [67]:  1 km_sc = KMeans(n_clusters=2,random_state=10,n_init=20,init='k-means++')
          2 km_sc.fit(df_scale)

Out[67]: KMeans(n_clusters=2, n_init=20, random_state=10)

In [68]:  1 km_sc.inertia_

Out[68]: 563355.2764817034

In [70]:  1 print('Silhoutte Score for KMeans clustering using StandardScaler:',silhouette_score(df_scale,labels=km_sc.labels_))

         Silhoutte Score for KMeans clustering using StandardScaler: 0.21213398843885112
```

7.1.2 Agglomerative Clustering:

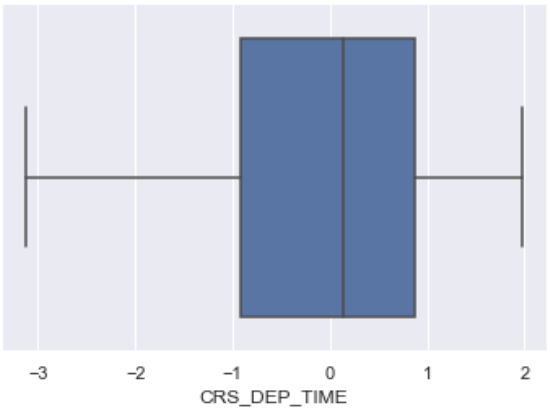
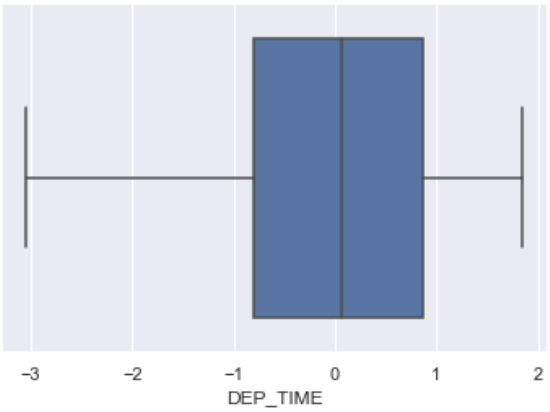
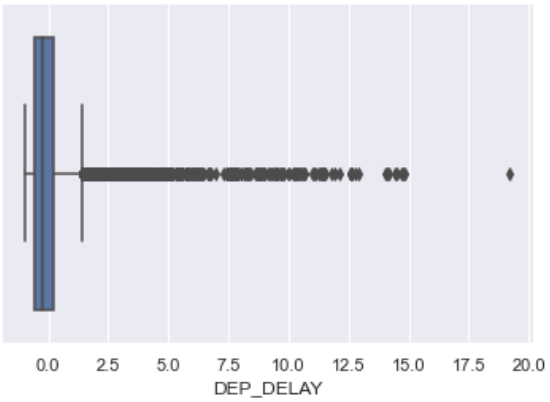
Also known as Hierarchical clustering, does not require the user to specify the number of clusters. Initially, each point is considered as a separate cluster, then it recursively clusters

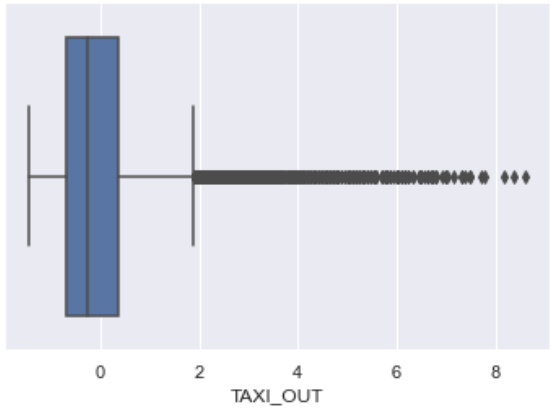
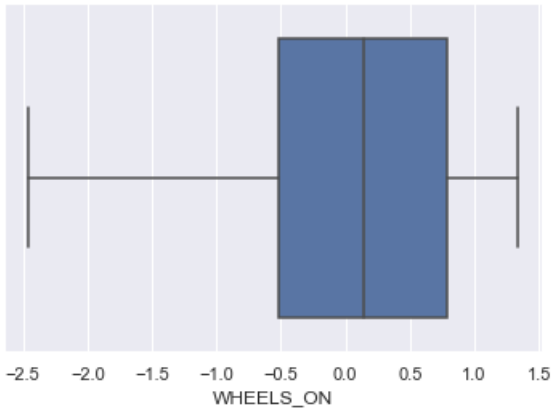
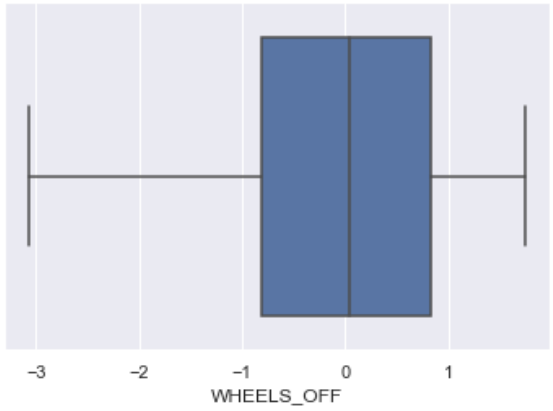
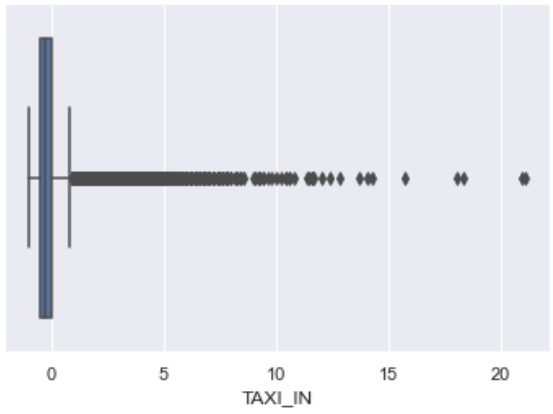


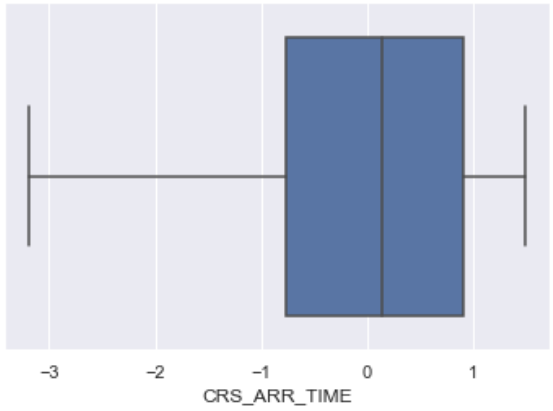
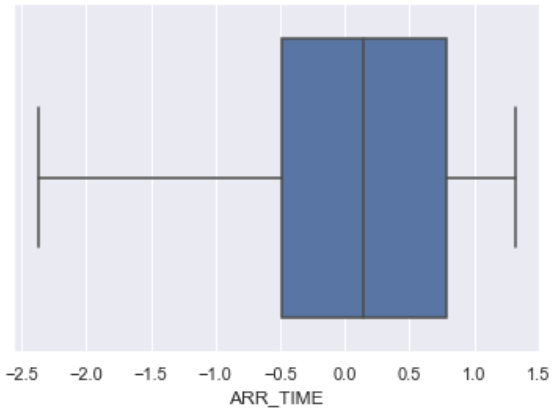
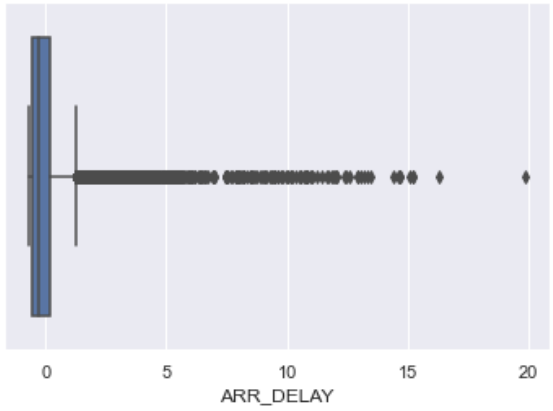
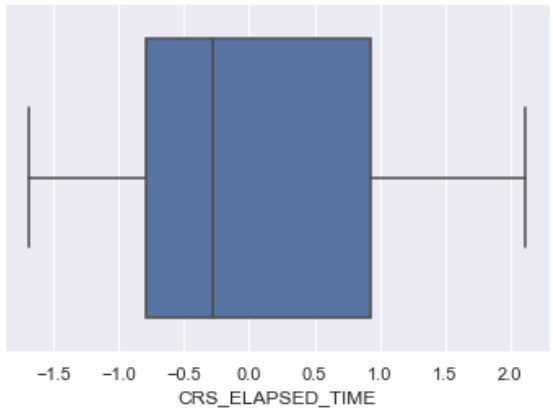
the points together depending upon the distance between them. The points are clustered in such a way that the distance between points within a cluster is minimum and distance between the cluster is maximum. Commonly used distance measures are Euclidean distance, Manhattan distance or Mahalanobis distance. Unlike k-means clustering, it is "bottom-up" approach.

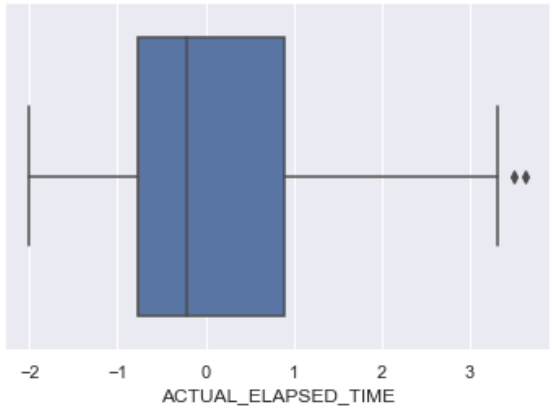
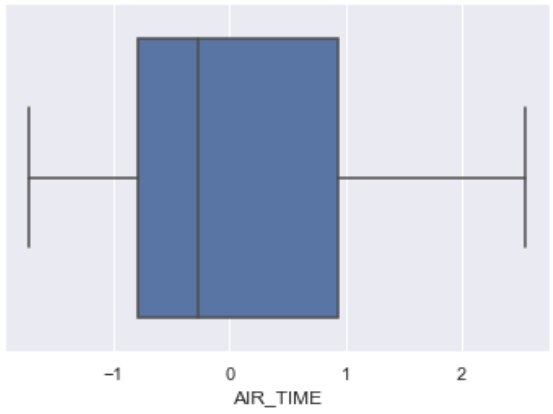
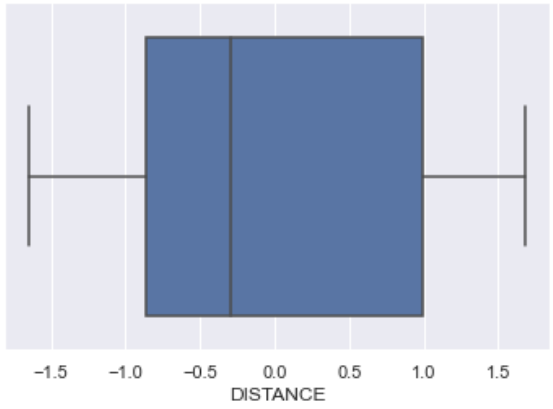
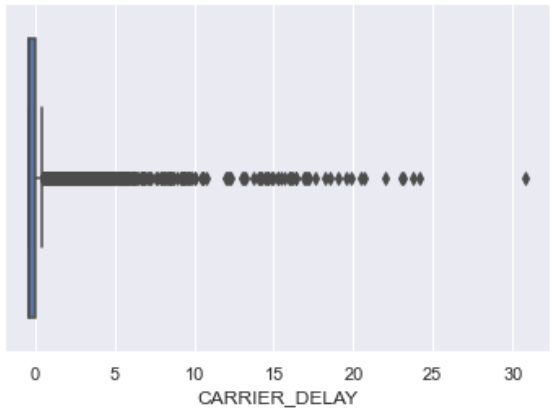
```
In [72]: 1 agg_sc = AgglomerativeClustering(n_clusters=2,affinity='euclidean',linkage='ward')

In [73]: 1 agg1 = agg_sc.fit(df_scale)

In [74]: 1 predict_sc = agg1.fit_predict(df_scale)
         2 predict_sc

Out[74]: array([0, 0, 1, ..., 1, 0, 0], dtype=int64)

In [75]: 1 print('Silhoutte Score for Agglomerative clustering using StandardScaler:',silhouette_score(df_scale,labels=predict_sc))

         Silhoutte Score for Agglomerative clustering using StandardScaler: 0.16704671634514046
```

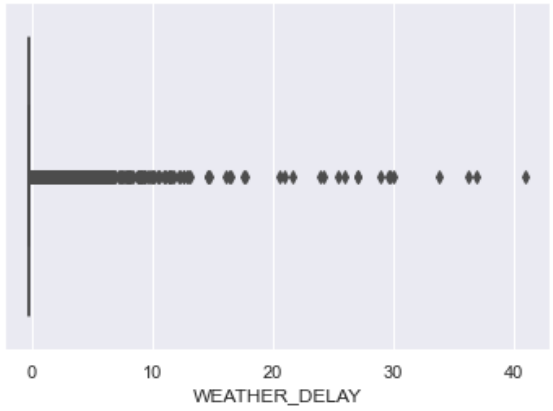
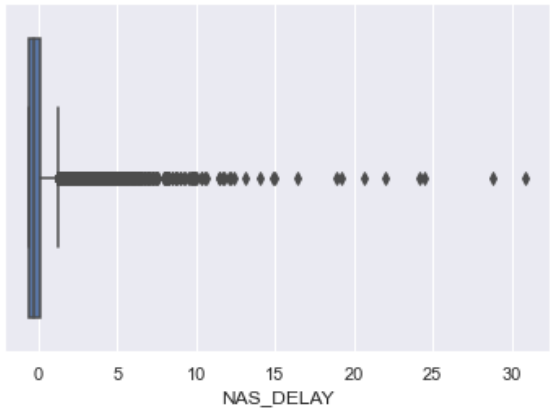
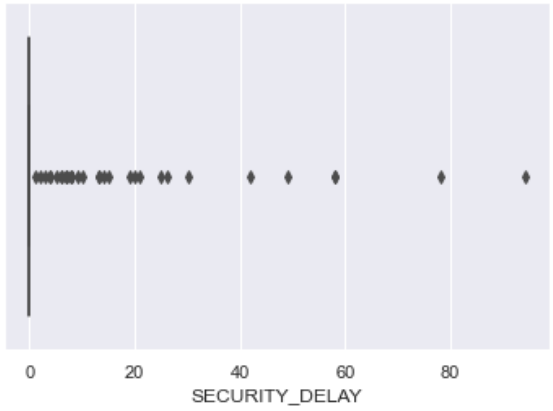
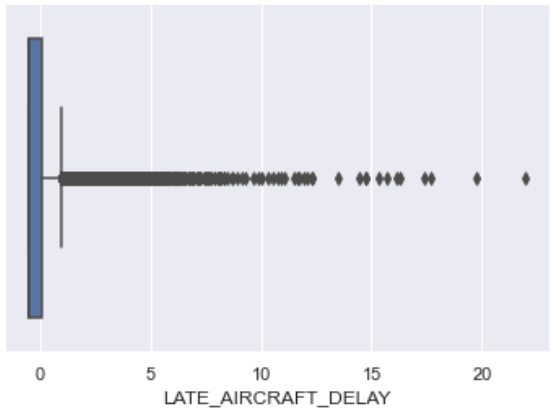
7.1.3 Checking Outliers:

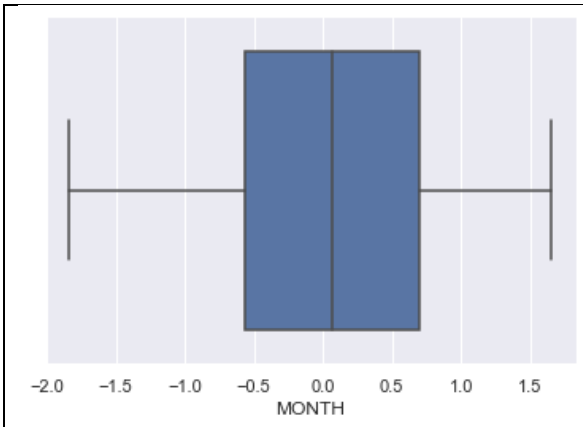
	• There are no outliers.
	• There are no outliers.
	• There are significant outliers present.

	• There are significant outliers present.
	• There are no outliers.
	• There are no outliers.
	• There are significant outliers present.

	• There are no outliers.
	• There are no outliers.
	• There are significant outliers present.
	• There are no outliers.

	• There are few outliers present.
	• There are no outliers.
	• There are no outliers.
	• There are significant outliers present.

	• There are significant outliers present.
	• There are significant outliers present.
	• There are significant outliers present.
	• There are significant outliers present.



- There are no outliers.

7.2 Preprocessing using PowerTransformer

Since there are still some existences of outliers, we will pre-process and work on the ML methods. PowerTransformer are a family of parametric, monotonic transformations that aim to map data from any distribution to as close to a Gaussian distribution as possible in order to stabilize variance and minimize skewness. StandardScaler () doesn't guarantee balanced feature scales in the presence of outliers.

PowerTransfromer applies a power transformation to each feature to make the data more Gaussian-like. scikit-learn's PowerTransformer () implements the Yeo-Johnson and Box-Cox transforms. The power transform finds the optimal scaling factor to stabilize variance and minimize skewness. By default, PowerTransformer also applies zero-mean, unit variance normalization to the transformed output. Box-Cox can only be applied to positive data. If

PowerTransformer

```
In [76]: 1 pt=PowerTransformer()
         2 df_pt=pd.DataFrame(pt.fit_transform(df_scale),columns=df_scale.columns)

In [77]: 1 df_pt.head()
```

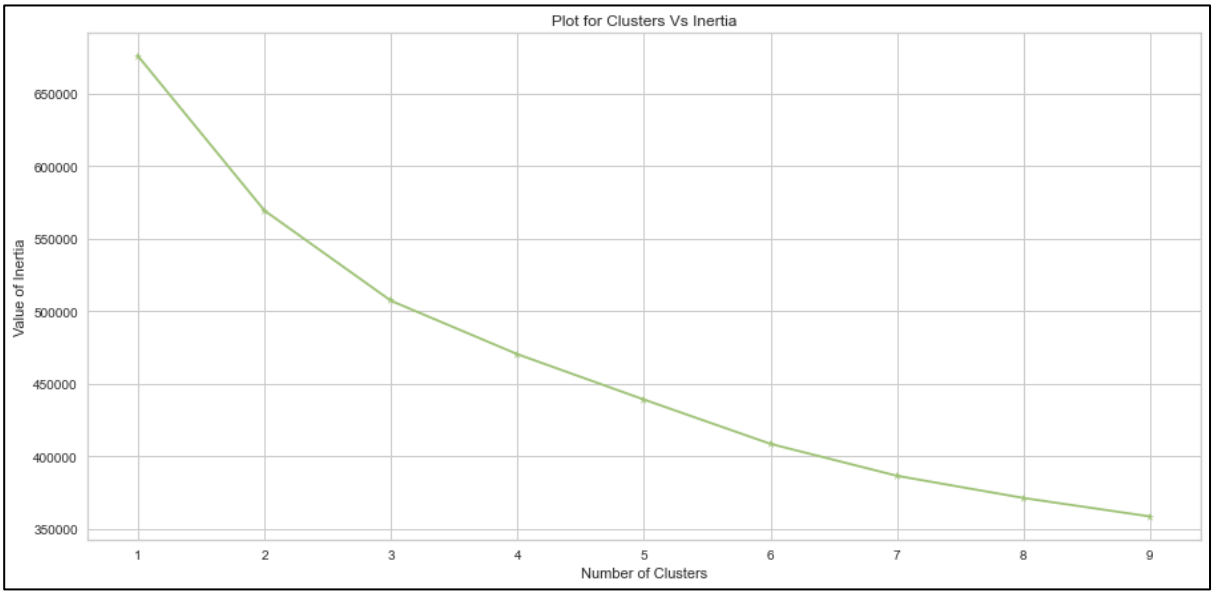
Out[77]:

	OP_CARRIER	ORIGIN	DEST	CRS_DEP_TIME	DEP_TIME	DEP_DELAY	TAXI_OUT	WHEELS_OFF	WHEELS_ON	TAXI_IN	...	CRS_ELAPSED_TIME	A
0	1.043103	0.0	-0.254744	-0.238775	-0.048694	0.873204	-1.527204	-0.114785	0.456966	0.177557	...	0.386218	
1	1.043103	0.0	-0.254744	1.121241	1.104012	0.217658	-0.805479	1.050169	-1.887606	-0.425748	...	0.400931	
2	1.043103	0.0	0.451701	-1.161986	-0.936157	1.122218	-0.939461	-0.895748	-0.682948	-0.101818	...	1.546605	
3	1.043103	0.0	1.506238	-1.541028	-1.431297	0.310101	-1.222926	-1.372909	-0.967190	1.803422	...	1.527116	
4	1.043103	0.0	-0.550663	-0.249537	-0.249160	-0.245718	-0.010028	-0.286849	-0.118103	-0.101818	...	-1.839638	

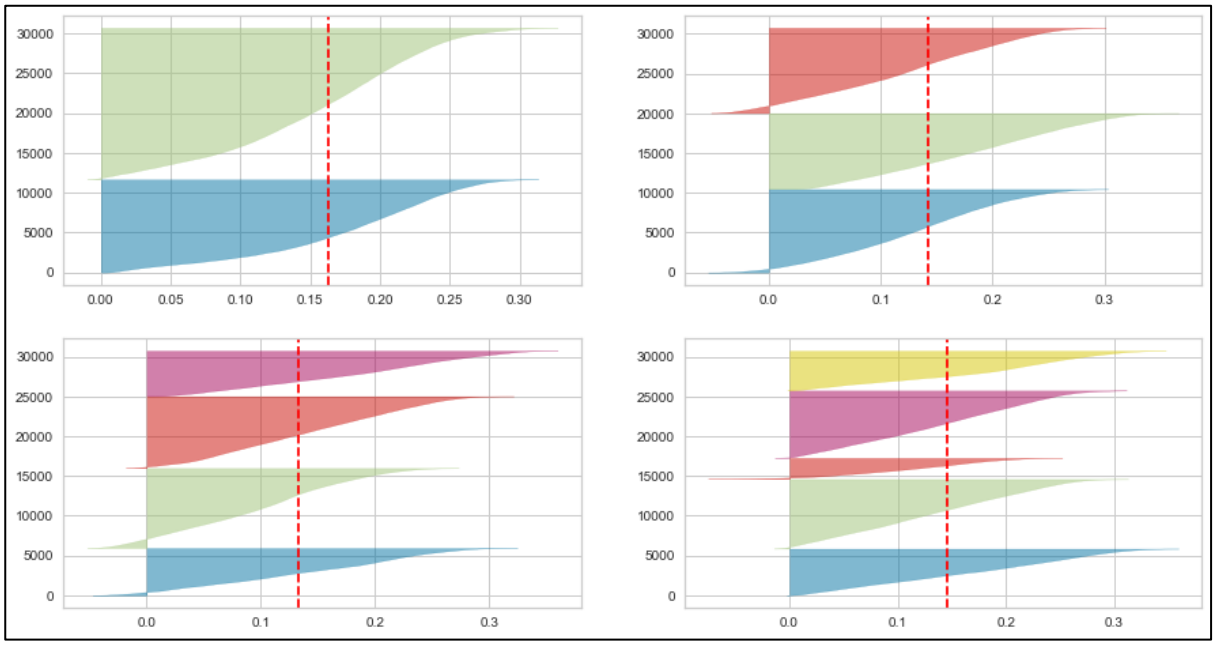
5 rows × 23 columns

negative values are present the Yeo-Johnson transformed is to be preferred. After using the power transformation, the transformed dataset is treated again for the KMean clustering technique.

7.2.1 KMeans Clustering:




```
In [83]:
for k in range(2,10):
    kmean2 = KMeans(n_clusters=k,random_state=10)
    kmean2.fit(df_pt)
    print(f'Cluster NUmber K = {k}: Silhoutte Score = {silhouette_score(df_pt,labels=kmean2.labels_)}')

Cluster NUmber K = 2: Silhoutte Score = 0.16315079331096718
Cluster NUmber K = 3: Silhoutte Score = 0.14286325535055375
Cluster NUmber K = 4: Silhoutte Score = 0.1332691488602817
Cluster NUmber K = 5: Silhoutte Score = 0.1454426155705803
Cluster NUmber K = 6: Silhoutte Score = 0.13665277704486803
Cluster NUmber K = 7: Silhoutte Score = 0.13659441062511074
Cluster NUmber K = 8: Silhoutte Score = 0.12841672635396123
Cluster NUmber K = 9: Silhoutte Score = 0.12542416669695755
```



Here is the Silhouette analysis done on the above plots with an aim to select an optimal value for n_clusters.

The value of n_clusters as 4 and 5 looks to be suboptimal for the given data due to the following reasons:

Presence of clusters with below average silhouette scores

Wide fluctuations in the size of the silhouette plots.

The value of 2 and 3 for n_clusters look to be optimal one. The silhouette score for each cluster is above average silhouette scores. Also, the fluctuation in size is similar. The thickness of the silhouette plot representing each cluster also is a deciding point. For plot with n_clusters 3 (top right), the thickness is more uniform than the plot with n_clusters as 2 (top left) with one cluster thickness much more than the other. Thus, one can select the optimal number of clusters as 2.

```
km_pt = KMeans(n_clusters=2,random_state=10,n_init=20,init='k-means++')
km_pt.fit(df_pt)
```

```
KMeans(n_clusters=2, n_init=20, random_state=10)
```

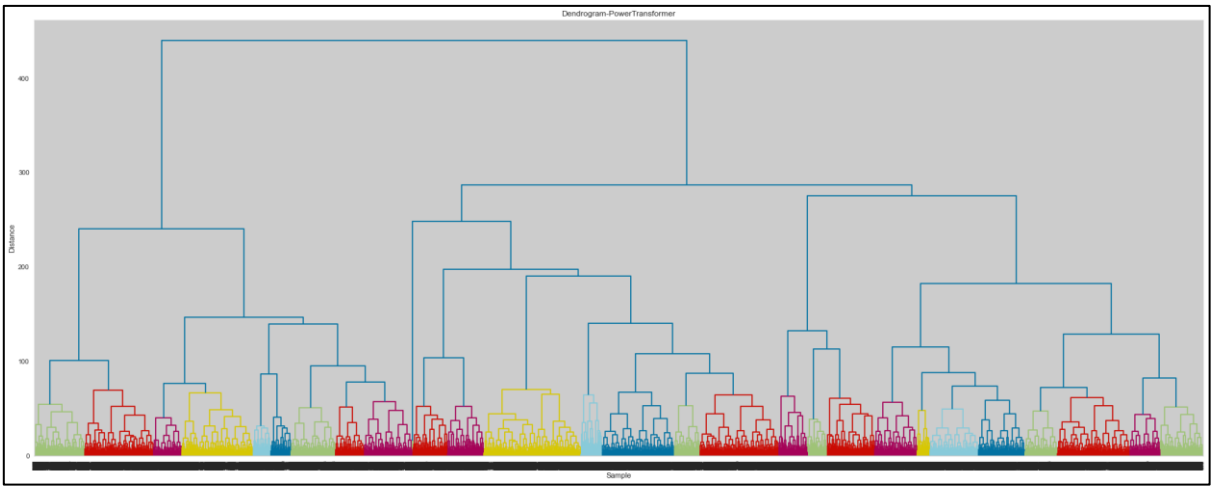
```
km_pt.inertia_
```

```
569314.9403731767
```

```
print('Silhoutte Score for KMeans clustering using PowerTransformer:',silhouette_score(df_pt,labels=km_pt.labels_))
```

```
Silhoutte Score for KMeans clustering using PowerTransformer: 0.16315079331096718
```

7.2.3 Agglomerative Clustering:




```
In [92]: agg_pt = AgglomerativeClustering(n_clusters=2,affinity='euclidean',linkage='ward')
```

```
In [93]: agg2 = agg_pt.fit(df_pt)
```

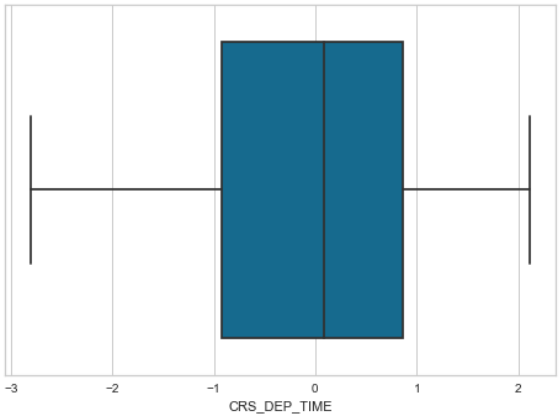
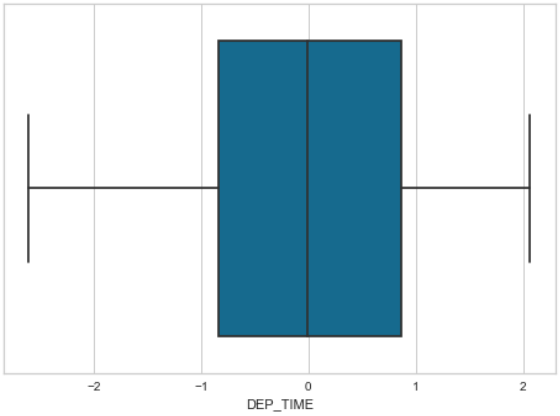
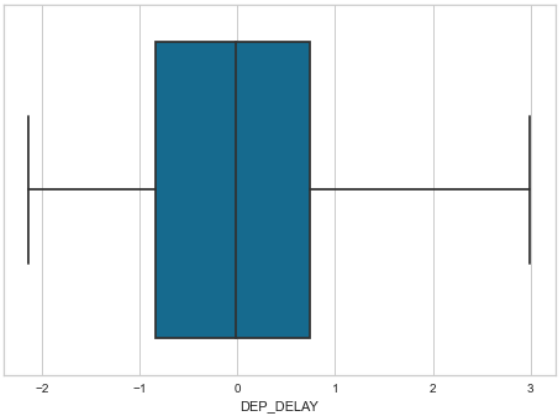
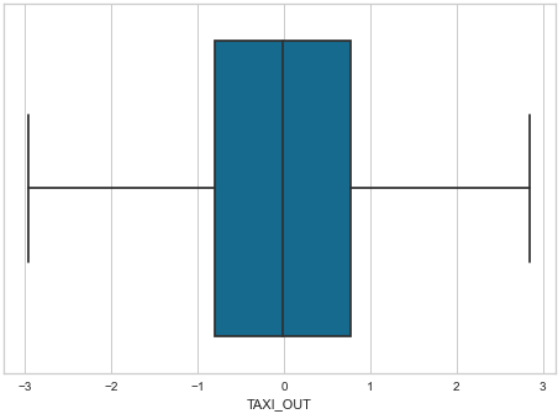
```
In [94]: predict_pt = agg2.fit_predict(df_pt)
         predict_pt
```

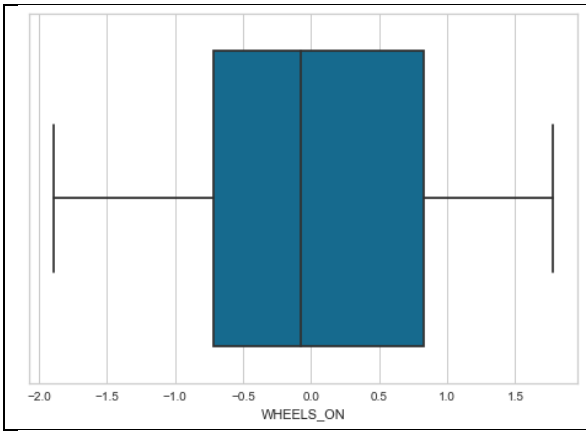
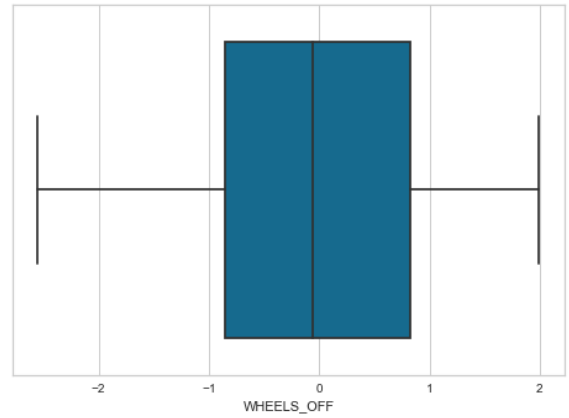
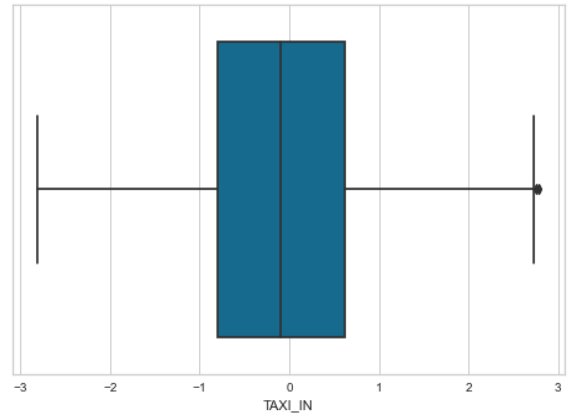
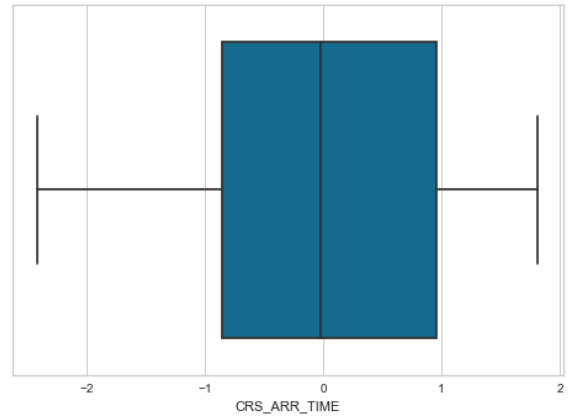
```
Out[94]: array([0, 0, 1, ..., 1, 0, 0], dtype=int64)
```

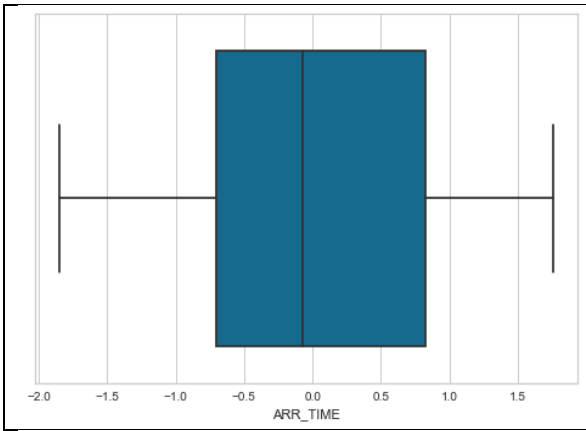
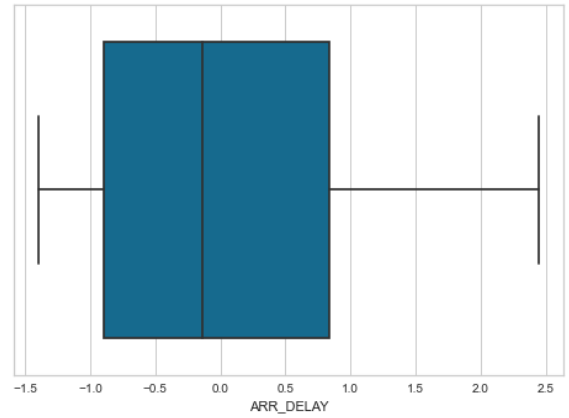
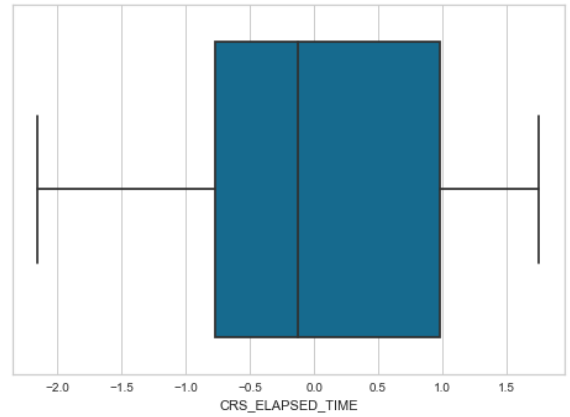
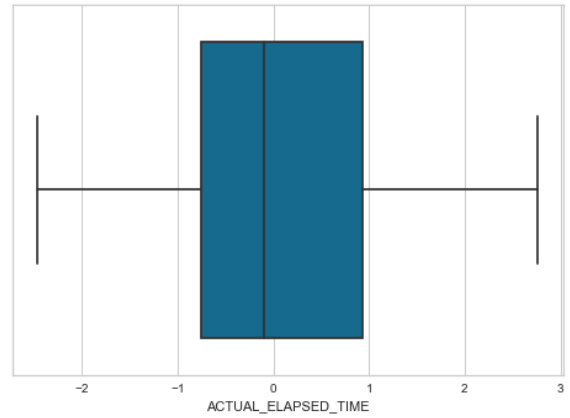
```
In [95]: print('Silhoutte Score for Agglomerative clustering using PowerTransformer:',silhouette_score(df_pt,labels=predict_pt))
```

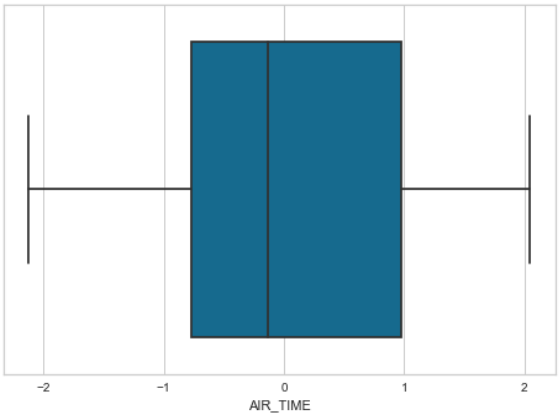
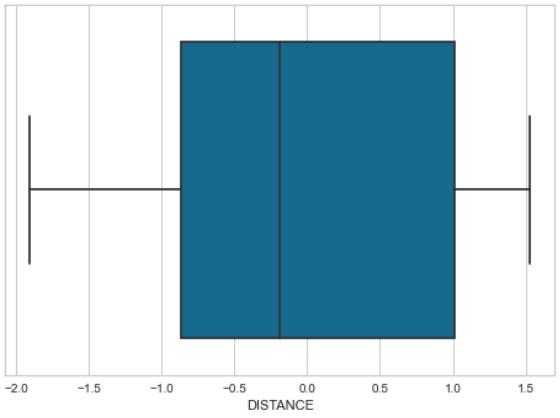
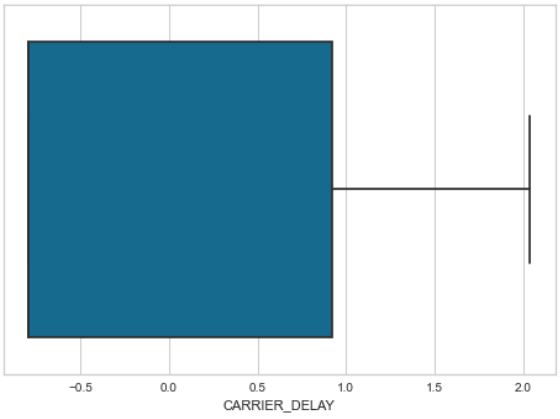
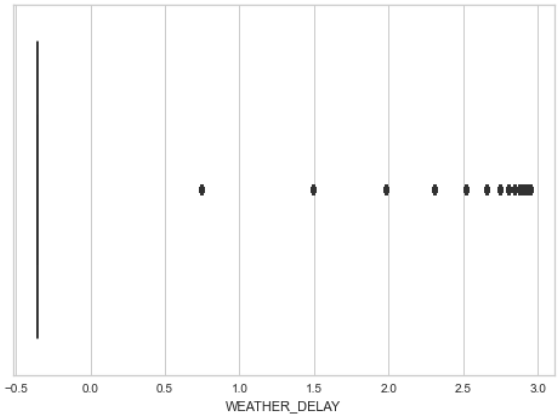
```
Silhoutte Score for Agglomerative clustering using PowerTransformer: 0.14683702694698972
```

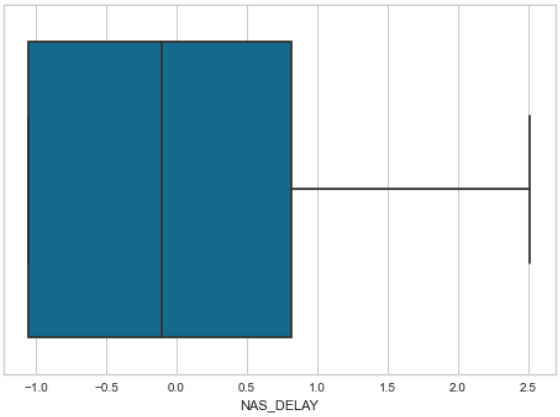
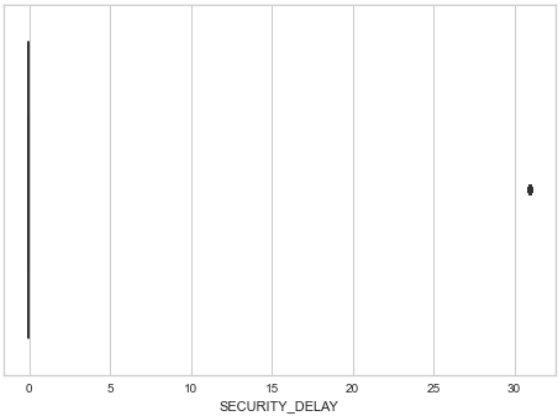
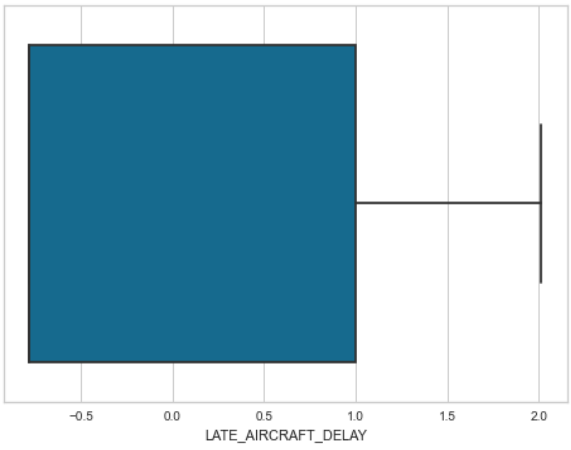
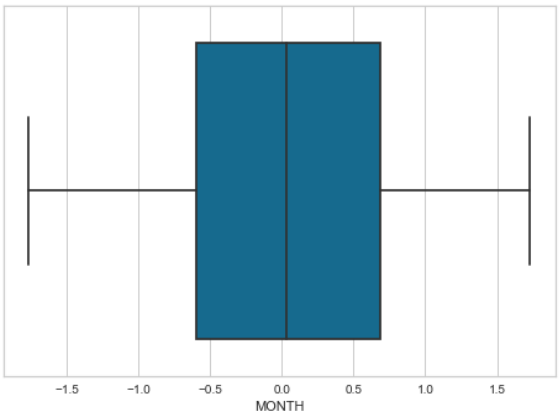
7.2.3 Checking Outliers:

	• There are no outliers.
	• There are no outliers.
	• There are no outliers.
	• There are no outliers.

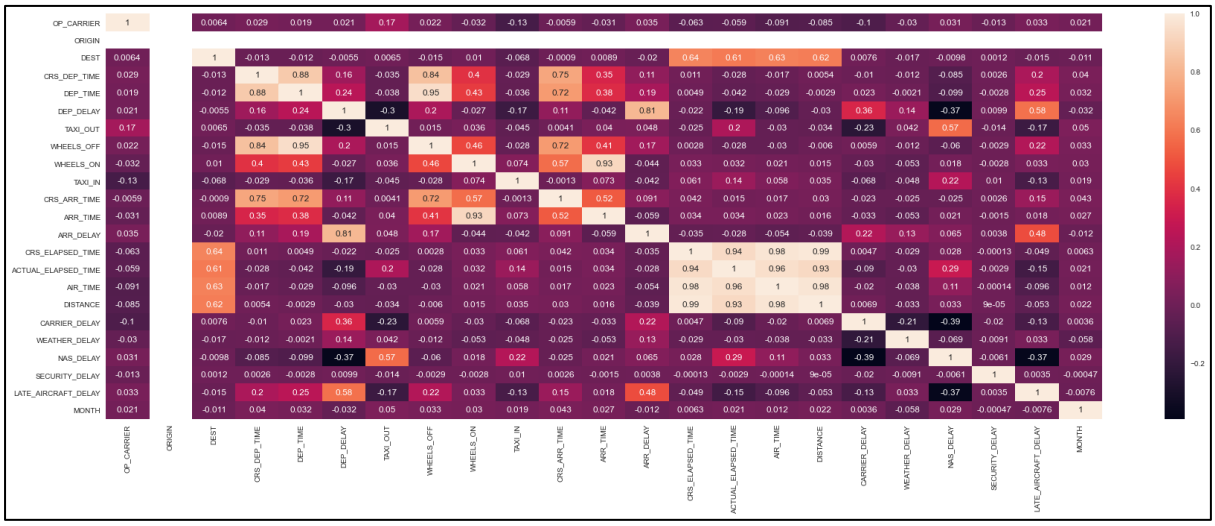
	• There are no outliers.
	• There are no outliers.
	• There is a very minimal outlier present.
	• There are no outliers

	• There are no outliers.
	• There are no outliers.
	• There are no outliers.
	• There are no outliers.

	• There are no outliers.
	• There are no outliers
	• There are no outliers and the lower whisker value is the minimum value.
	• There are outliers present.

	• There are no outliers and the lower whisker value is the minimum value.
	• The are no data except the few outliers at the extreme end on maximum side.
	• There are no outliers and the lower whisker value is the minimum value.
	• There are no outliers.

7.2.4 Correlation of features:



There are still the features that are correlated and this could affect the results.

7.3 PRINCIPAL COMPONENT ANALYSIS-DIMENSIONALITY REDUCTION TECHNIQUE

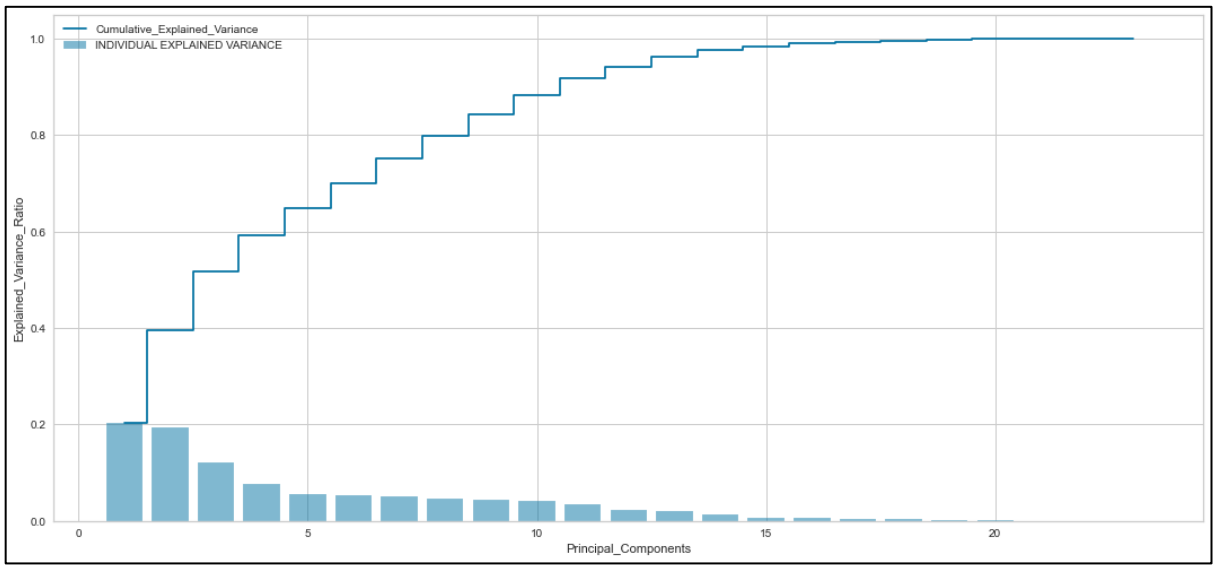
Principal Component Analysis is a technique used for dimensionality reduction, meaning that the dimensionality or complexity of the data is represented in a simpler fashion. The Principal Component Analysis algorithm finds new dimensions for the data that are orthogonal. While the dimensionality of the data is reduced, the variance between the data should be preserved as much as possible. What this means in practical terms is that it takes the features in the dataset and distils them down into fewer features that represent most of the data.

```
In [102]:
pca=PCA()
df_pca=pca.fit_transform(df_pt)

In [103]:
pca.explained_variance_

Out[103]: array([4.45978002e+00, 4.25961012e+00, 2.64297273e+00, 1.67475297e+00,
       1.22709026e+00, 1.16443739e+00, 1.11392229e+00, 1.00559612e+00,
       9.81026339e-01, 9.05798080e-01, 7.84858590e-01, 5.02286570e-01,
       4.51924802e-01, 3.14460884e-01, 1.52162270e-01, 1.40646890e-01,
       7.27448669e-02, 6.76873384e-02, 4.49435361e-02, 1.74353177e-02,
       8.99471635e-03, 7.58411056e-03, 2.54270470e-33])

In [104]:
var_exp_ratio =pca.explained_variance_ratio_
var_exp_ratio

Out[104]: array([2.02710674e-01, 1.93612339e-01, 1.20131213e-01, 7.61226567e-02,
       5.57750144e-02, 5.29272491e-02, 5.06311829e-02, 4.57074267e-02,
       4.45906547e-02, 4.11712996e-02, 3.56742291e-02, 2.28304645e-02,
       2.05413677e-02, 1.42932112e-02, 6.91624167e-03, 6.39283231e-03,
       3.30647722e-03, 3.07659704e-03, 2.04282150e-03, 7.92488642e-04,
       4.08837433e-04, 3.44721076e-04, 1.15573724e-34])

In [105]:
cum_var = np.cumsum(var_exp_ratio)
cum_var

Out[105]: array([0.20271067, 0.39632301, 0.51645423, 0.59257688, 0.6483519 ,
       0.70127915, 0.75191033, 0.79761776, 0.84220841, 0.88337971,
       0.91905394, 0.9418844 , 0.96242577, 0.97671898, 0.98363522,
       0.99002806, 0.99333453, 0.99641113, 0.99845395, 0.99924644,
       0.99965528, 1.        , 1.        ])
```

As per the correlation map plotted using PowerTransformer technique, we could visualise that there are certain variables which were still highly correlated. It means there is a significant multicollinearity. PCA help in reducing the redundancy caused due to these highly correlated variables by converting in to the smaller size variables called Principal Components (PC). There was a linear relation between two variables in the pairplot. This

again signifies that all the variables are not independent. Some time it might also cause curse of dimensionality unnecessarily. PCA helps in eliminating those issues.

In [108]:
```
plt.figure(figsize=(15,7))
plt.bar(range(1,24),var_exp_ratio,alpha=0.5,align="center",label="INDIVIDUAL EXPLAINED VARIANCE")
plt.step(range(1,24),cum_var,where="mid",label="Cumulative_Explained_Variance")
plt.ylabel("Explained_Variance_Ratio")
plt.xlabel("Principal_Components")
plt.legend(loc="best")
plt.tight_layout()
plt.show()
```



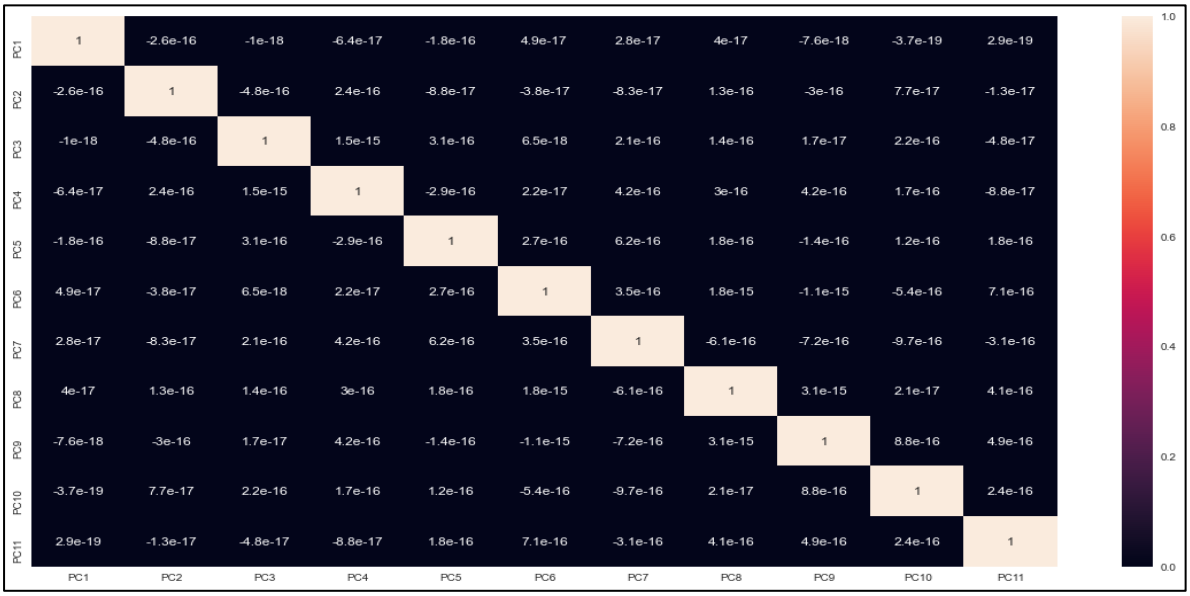
In [109]:
```
# For 90%  variance the best number of PCA components =11
```

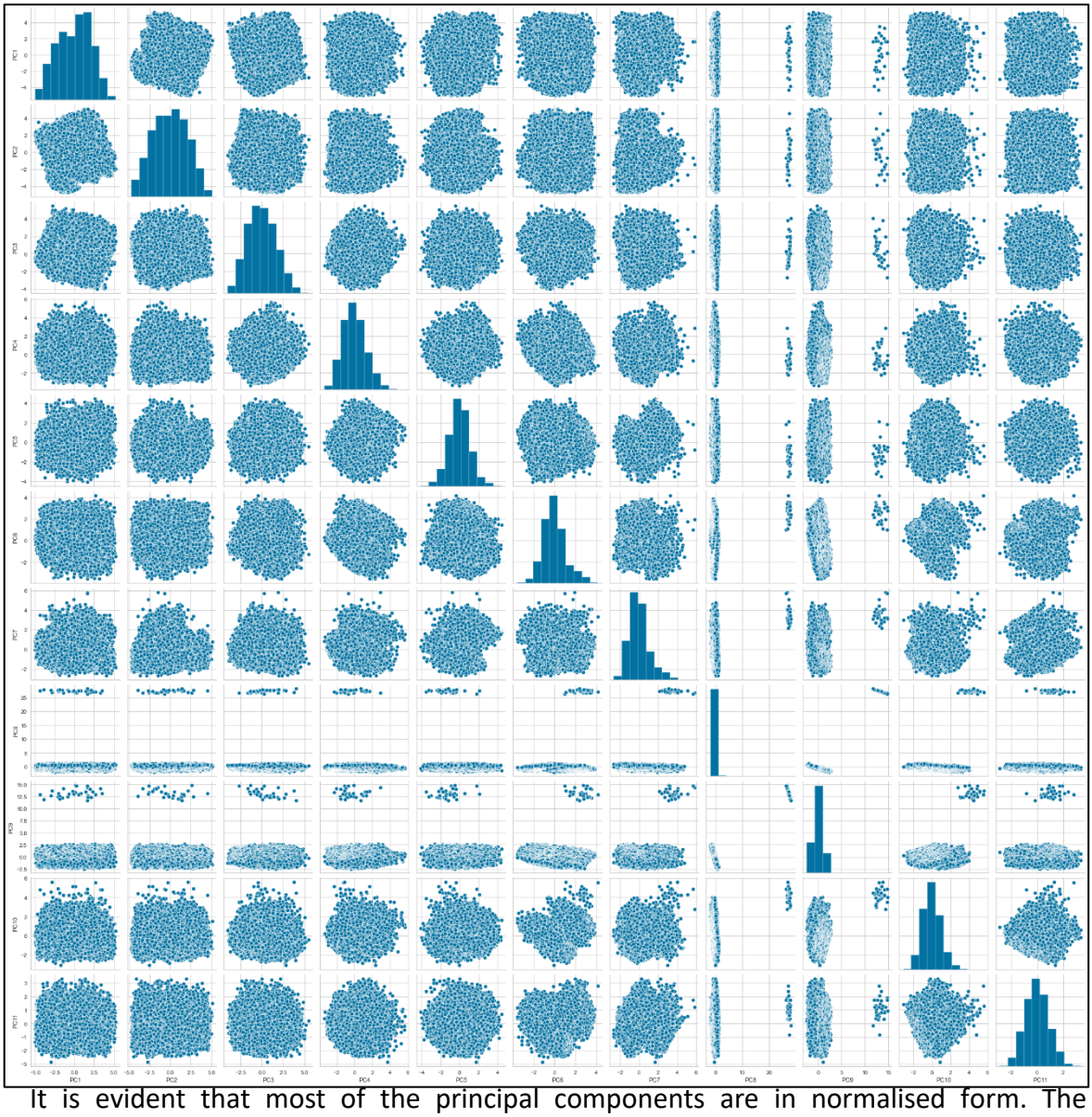
In [110]:
```
pca_pt = PCA(n_components=11).fit_transform(df_pt)
```

In [111]:
```
df_pca = pd.DataFrame(pca_pt,columns=['PC1','PC2','PC3','PC4','PC5','PC6','PC7','PC8','PC9','PC10','PC11'])
df_pca.head()
```

Out[111]:

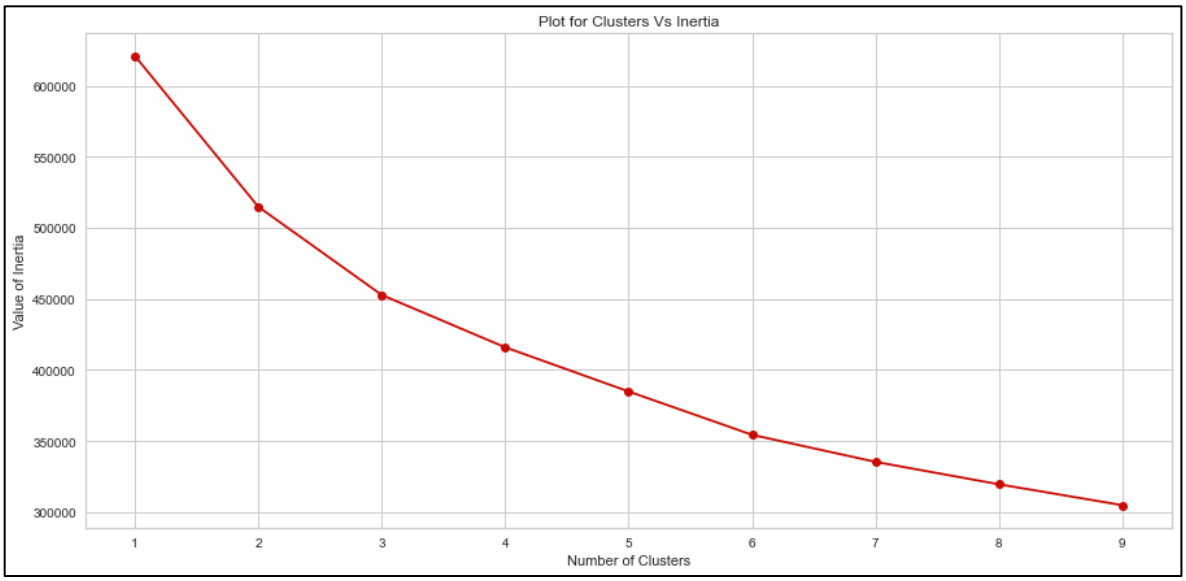
	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9	PC10	PC11
0	0.251165	-0.718901	2.119981	-0.736829	-0.359810	0.674736	-0.996928	-0.204132	0.954359	-1.837489	0.661206
1	0.207724	-0.790670	1.013057	0.785328	3.084457	1.978462	2.150800	-1.166691	1.224834	0.787391	2.011637
2	-2.746664	0.897463	2.304345	0.776900	0.391882	2.681364	-0.567098	-0.806705	0.907926	0.544650	2.192089
3	-4.038358	1.613703	1.591935	0.672689	-0.402407	2.832028	0.273367	-0.706979	0.640744	-0.307002	2.945836
4	3.168999	2.169435	-0.823574	0.098881	1.299704	2.669820	-0.423502	-0.896003	1.056931	0.785107	1.832966





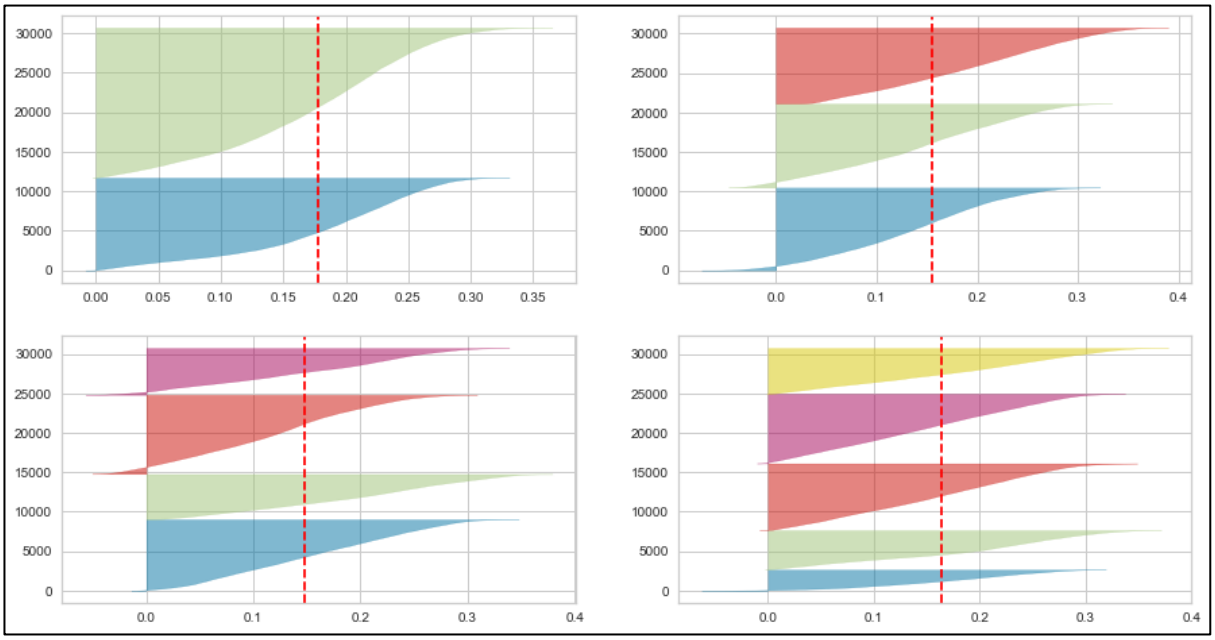
It is evident that most of the principal components are in normalised form. The scatterplot is evident to show there are no linear relation among any variables. The data points in spherical form, hyper spherical form is suitable for the clustering technique.

7.3.1 KMeans Clustering:



```
In [119]:
for k in range(2,10):
    kmean3 = KMeans(n_clusters=k,random_state=10)
    kmean3.fit(df_pca)
    print(f'Cluster Number K = {k}: Silhoutte Score = {silhouette_score(df_pca,labels=kmean3.labels_)}')
```

```
Cluster Number K = 2: Silhoutte Score = 0.17742861085311717
Cluster Number K = 3: Silhoutte Score = 0.1556280017669516
Cluster Number K = 4: Silhoutte Score = 0.14738816780015854
Cluster Number K = 5: Silhoutte Score = 0.163455199076137
Cluster Number K = 6: Silhoutte Score = 0.14900974711875997
Cluster Number K = 7: Silhoutte Score = 0.15692564675086523
Cluster Number K = 8: Silhoutte Score = 0.1461135023791945
Cluster Number K = 9: Silhoutte Score = 0.14320746657586028
```



```
In [121]:
km_pca = KMeans(n_clusters=2,random_state=10,n_init=20,init='k-means++')
km_pca.fit(df_pca)
```

```
Out[121]: KMeans(n_clusters=2, n_init=20, random_state=10)
```

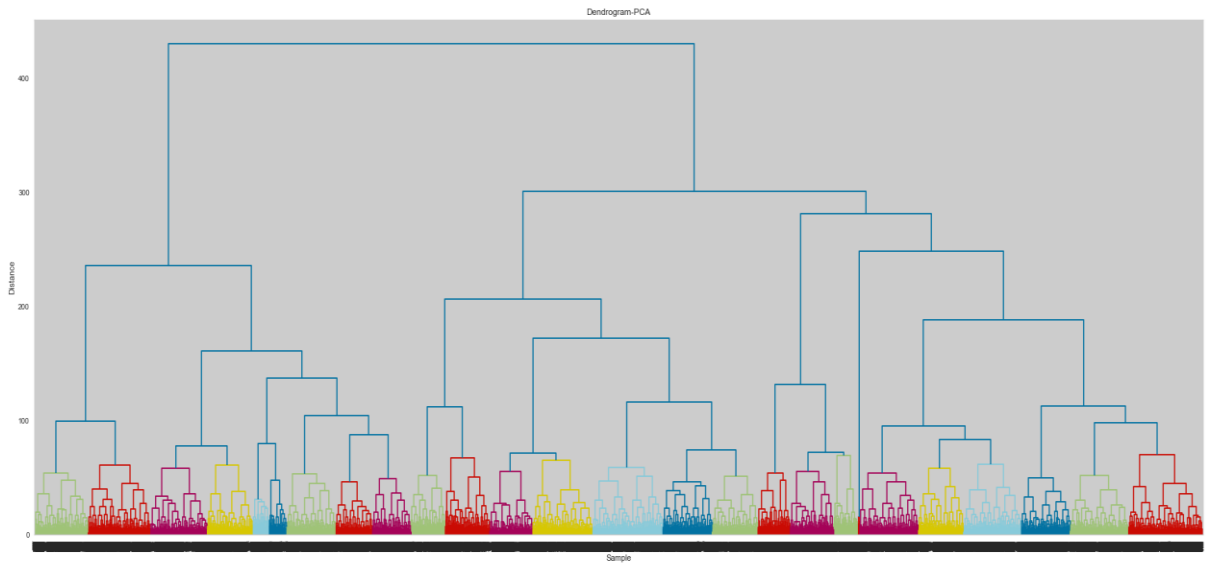
```
In [122]:
km_pca.inertia_
```

```
Out[122]: 514909.5762898326
```

```
In [123]:
print('Silhoutte Score for KMeans clustering after doing PCA:',silhouette_score(df_pca,labels=km_pca.labels_))
```

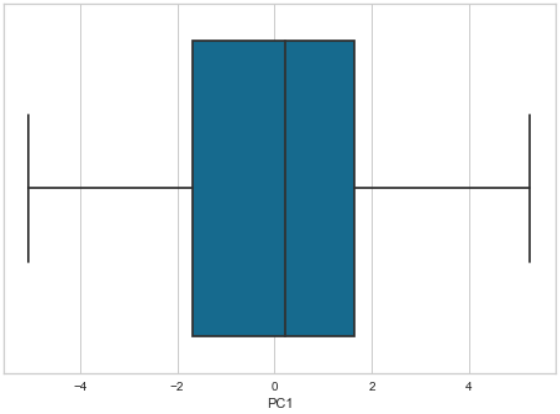
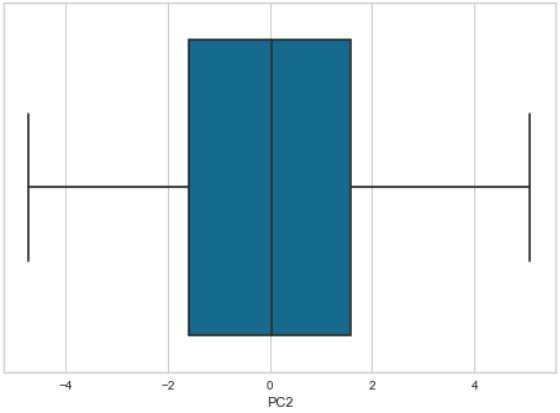
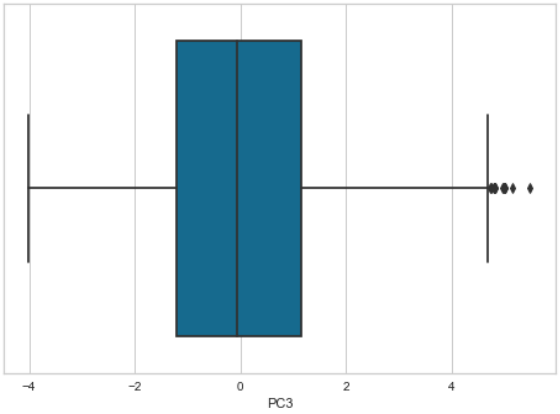
```
Silhoutte Score for KMeans clustering after doing PCA: 0.17742861085311717
```

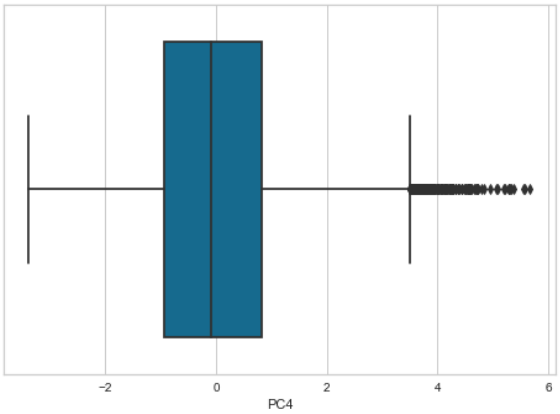
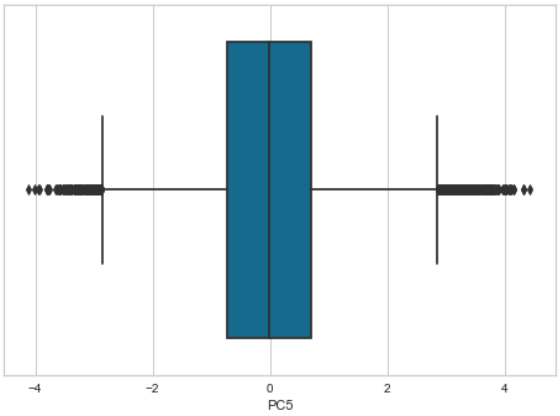
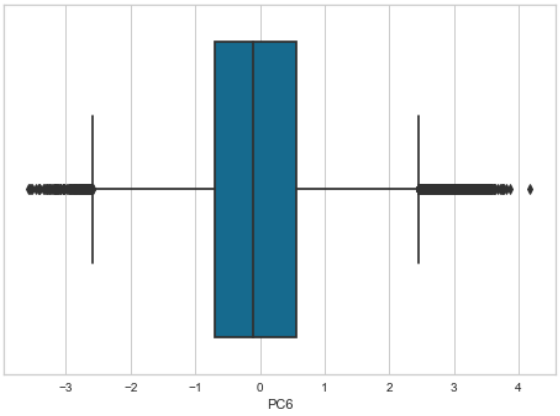
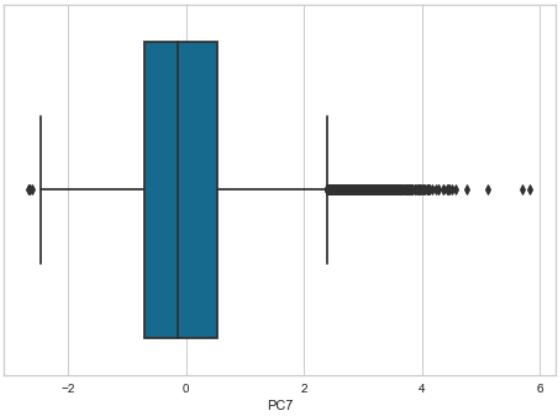
7.3.2 Agglomerative Clustering:

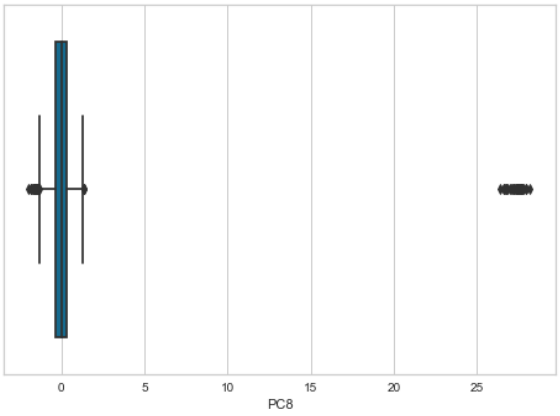
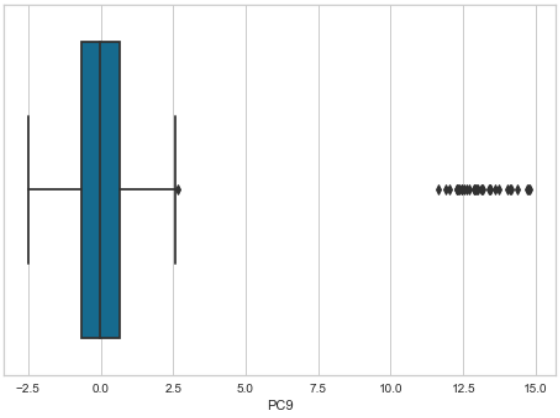
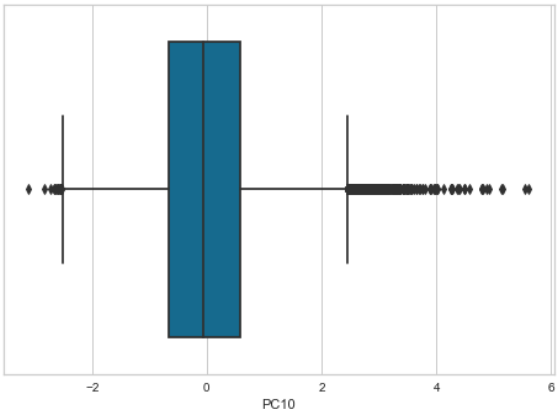
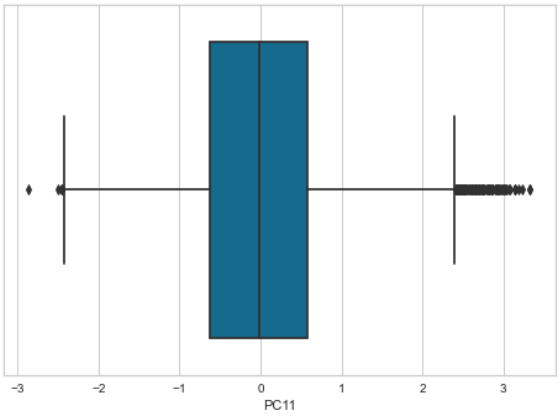


```
In [125]:
1 agg_pca = AgglomerativeClustering(n_clusters=2,affinity='euclidean',linkage='ward')

In [126]:
1 agg3 = agg_pca.fit(df_pca)

In [127]:
1 predict_pca = agg3.fit_predict(df_pca)
2 predict_pca

Out[127]: array([0, 0, 1, ..., 1, 0, 0], dtype=int64)

In [128]:
1 print('Silhoutte Score for Agglomerative clustering after doing PCA:',silhouette_score(df_pca,labels=predict_pca))

Silhoutte Score for Agglomerative clustering after doing PCA: 0.15751946793817276
```

7.3.3 Checking outliers:

	There are no outliers present.
	There are no outliers present.
	There are few outliers present in the maximum value side.

	There are outliers present in the maximum value side.
	There are outliers present in both sides.
	There are outliers present in both sides
	There are outliers present in the maximum value side.

	There are outliers present in the maximum value side which are very extreme from the distribution.
	There are outliers present in the maximum value side which are very extreme from the distribution.
	There are outliers present in both sides, but less in low value side and more in maximum value side.
	There are outliers present in both sides, but least in low value side and more in maximum value side.

7.4 MODEL INTERPRETATION:

```
In [129]:
df_silhoutte = pd.DataFrame({'Clustering_Methods':['Kmean after PowerTransformer','Agglomerative after PowerTransformer',
                                                   'Kmean after PCA','Agglomerative after PCA'],'Silhoutte_score':[0.1631507
```

```
In [130]:
df_silhoutte
```

Out[130]:

	Clustering_Methods	Silhoutte_score
0	Kmean after PowerTransformer	0.163151
1	Agglomerative after PowerTransformer	0.146837
2	Kmean after PCA	0.177429
3	Agglomerative after PCA	0.157519

- Based on the Silhouette score, the Kmean clustering done after PCA is better compared to rest which is 0.177429.

```
In [131]:
df_pt['Cluster_KM_PT']=km_pt.labels_
df_pt['Cluster_AG_PT']=predict_pt
df_pt['Cluster_KM_PCA']=km_pca.labels_
df_pt['Cluster_AG_PCA']=predict_pca
```

```
In [132]:
df_pt.head()
```

Out[132]:

	OP_CARRIER	ORIGIN	DEST	CRS_DEP_TIME	DEP_TIME	DEP_DELAY	TAXI_OUT	WHEELS_OFF	WHEELS_ON	TAXI_IN	...	CARRIER_DELAY	WEAT
0	1.043103	0.0	-0.254744	-0.238775	-0.048694	0.873204	-1.527204	-0.114785	0.456966	0.177557	...	0.672624	
1	1.043103	0.0	-0.254744	1.121241	1.104012	0.217658	-0.805479	1.050169	-1.887606	-0.425748	...	-0.792373	
2	1.043103	0.0	0.451701	-1.161986	-0.936157	1.122218	-0.939461	-0.895748	-0.682948	-0.101818	...	-0.792373	
3	1.043103	0.0	1.506238	-1.541028	-1.431297	0.310101	-1.222926	-1.372909	-0.967190	1.803422	...	-0.792373	
4	1.043103	0.0	-0.550663	-0.249537	-0.249160	-0.245718	-0.010028	-0.286849	-0.118103	-0.101818	...	-0.792373	

5 rows × 27 columns

- All the clustered values are added as separate are added to the df_pt dataframe. The dataframe which was pre-processed using PowerTransformer.

```
In [133]:
df_pt['Cluster_KM_PT'].value_counts()
```

```
Out[133]:
1    19033
0    11685
Name: Cluster_KM_PT, dtype: int64
```

```
In [135]:
df_pt['Cluster_AG_PT'].value_counts()
```

```
Out[135]:
0    20790
1     9928
Name: Cluster_AG_PT, dtype: int64
```

```
In [136]:
df_pt['Cluster_KM_PCA'].value_counts()
```

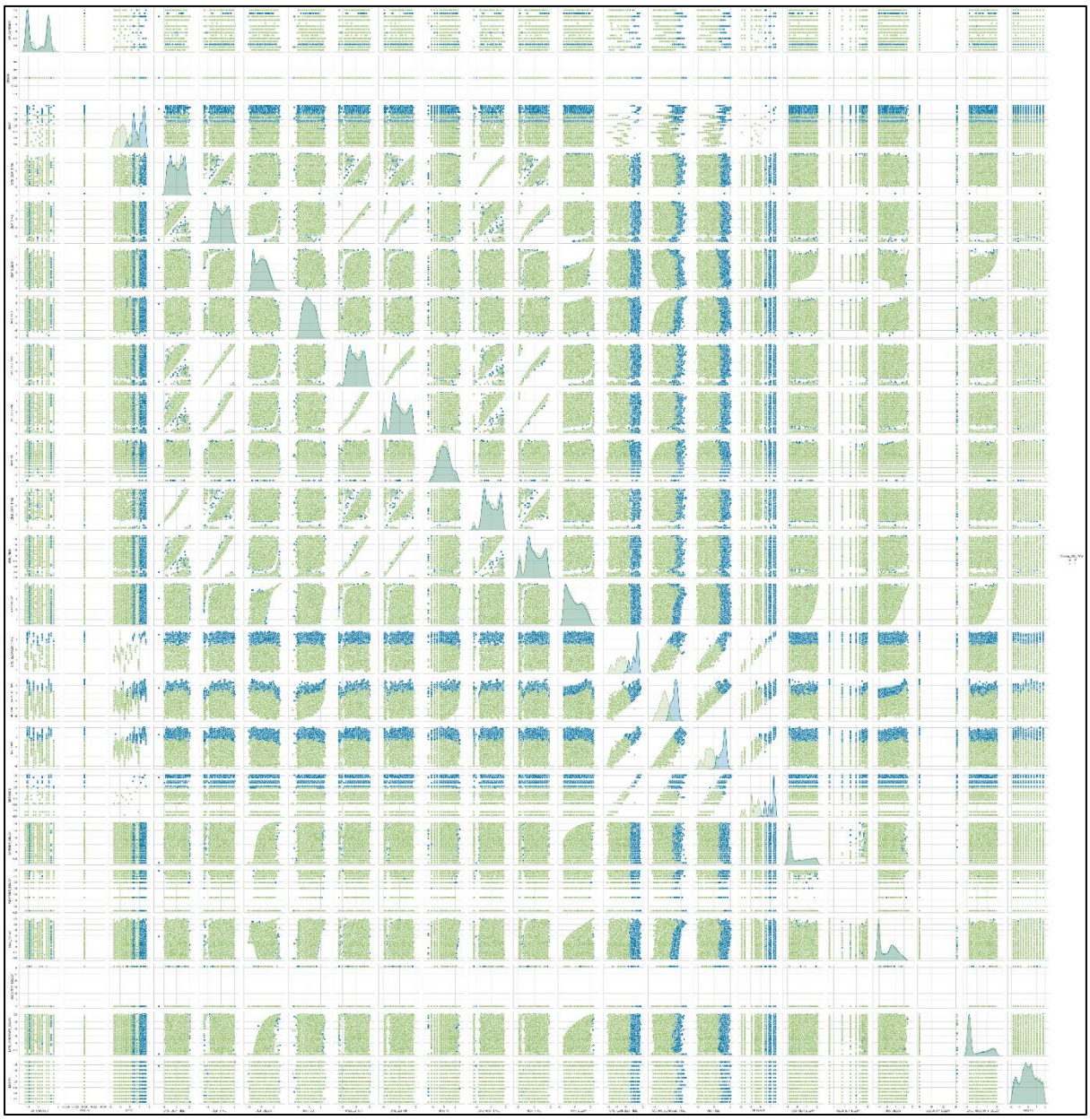
```
Out[136]:
1    18989
0    11729
Name: Cluster_KM_PCA, dtype: int64
```

```
In [137]:
df_pt['Cluster_AG_PCA'].value_counts()
```

```
Out[137]:
0    20822
1     9896
Name: Cluster_AG_PCA, dtype: int64
```

- The value counts show the clear picture of the clustering that has happened in the various techniques.
- Although the cluster number is same for all the techniques, that is 2. We can see the variation.
- The Cluster_KM_PCA which had the better Silhouette score has proper distribution of datapoints in the cluster.

7.4.1 Relationship of Cluster_KM_PCA with other features:



- The diagonal graph shows that the kde distribution of the two clusters in all the variable. At some variables the distribution is visible as separate like bell or gaussian distribution. And some variables its squashed.
- Even the distribution of datapoints could be seen which is dispersed between two cluster.

8.BUSINESS INFERENCE & SOLUTION

In [146]:
```
df1[df1['Cluster_KM_PCA']==0].sort_values(by='CARRIER_DELAY',ascending=False)[:10]
```

Out[146]:

	FL_DATE	OP_CARRIER	OP_CARRIER_FL_NUM	ORIGIN	DEST	CRS_DEP_TIME	DEP_TIME	DEP_DELAY	TAXI_OUT	WHEELS_OFF	...	CRS_ELAPSED
26841	2018-11-10	AA	1277	ORD	MIA	1830	2038.0	1568.0	18.0	2056.0	...	
6577	2018-04-15	DL	1887	ORD	SEA	1752	1323.0	1171.0	22.0	1345.0	...	
7280	2018-04-29	AA	2439	ORD	MIA	1347	940.0	1193.0	11.0	951.0	...	
15987	2018-07-20	OO	3099	ORD	SLC	2015	1638.0	1223.0	22.0	1700.0	...	
3508	2018-02-20	NK	1167	ORD	PHX	1515	842.0	1047.0	15.0	857.0	...	
2539	2018-02-09	AA	2193	ORD	MIA	1830	1123.0	1013.0	17.0	1140.0	...	
27859	2018-11-26	AA	1277	ORD	MIA	1830	1058.0	988.0	32.0	1130.0	...	
19896	2018-08-20	OO	5433	ORD	SLC	1952	1159.0	967.0	26.0	1225.0	...	
23019	2018-09-22	NK	245	ORD	LAS	1524	658.0	934.0	18.0	716.0	...	
12225	2018-06-18	OO	5433	ORD	SLC	1952	1048.0	896.0	26.0	1114.0	...	

10 rows × 25 columns

In [148]:
```
df1[df1['Cluster_KM_PCA']==0].sort_values(by='NAS_DELAY',ascending=False)[:10]
```

Out[148]:

	FL_DATE	OP_CARRIER	OP_CARRIER_FL_NUM	ORIGIN	DEST	CRS_DEP_TIME	DEP_TIME	DEP_DELAY	TAXI_OUT	WHEELS_OFF	...	CRS_ELAPSED
15113	2018-07-13	OO	5433	ORD	SLC	1952	644.0	652.0	22.0	706.0	...	
11381	2018-06-10	UA	262	ORD	SEA	1940	623.0	643.0	17.0	640.0	...	
27926	2018-11-26	OO	3565	ORD	SLC	735	1423.0	408.0	61.0	1524.0	...	
20758	2018-08-28	NK	731	ORD	LAX	1705	2136.0	271.0	110.0	2326.0	...	
13450	2018-06-26	NK	245	ORD	LAS	1521	1941.0	260.0	86.0	2107.0	...	
2759	2018-02-11	NK	245	ORD	LAS	1518	2044.0	326.0	15.0	2059.0	...	
20910	2018-08-28	NK	441	ORD	SEA	2030	146.0	316.0	14.0	200.0	...	
2479	2018-02-09	NK	1167	ORD	PHX	1517	2001.0	284.0	22.0	2023.0	...	
11157	2018-06-09	NK	563	ORD	SAN	1030	1507.0	277.0	23.0	1530.0	...	
5673	2018-04-03	NK	357	ORD	LAS	854	1323.0	269.0	25.0	1348.0	...	

10 rows × 25 columns

In [149]:
```
df1[df1['Cluster_KM_PCA']==0].sort_values(by='SECURITY_DELAY',ascending=False)[:10]
```

Out[149]:

	FL_DATE	OP_CARRIER	OP_CARRIER_FL_NUM	ORIGIN	DEST	CRS_DEP_TIME	DEP_TIME	DEP_DELAY	TAXI_OUT	WHEELS_OFF	...	CRS_ELAPSED
25408	2018-10-17	AA	1072	ORD	SLC	1010	1116.0	66.0	8.0	1124.0	...	
19656	2018-08-17	NK	425	ORD	FLL	1636	1806.0	90.0	47.0	1853.0	...	
24012	2018-10-05	NK	737	ORD	LAX	1030	1202.0	92.0	28.0	1230.0	...	
17203	2018-07-30	AA	2786	ORD	LAX	2205	16.0	131.0	16.0	32.0	...	
20287	2018-08-22	NK	457	ORD	LAS	2149	2316.0	87.0	15.0	2331.0	...	
22069	2018-09-09	AA	2498	ORD	MIA	630	649.0	19.0	19.0	708.0	...	
16504	2018-07-24	AA	335	ORD	MIA	1200	1317.0	77.0	13.0	1330.0	...	
22928	2018-09-21	AS	21	ORD	SEA	1940	2014.0	34.0	20.0	2034.0	...	
16470	2018-07-24	AA	562	ORD	PHX	1120	1240.0	80.0	10.0	1250.0	...	
20937	2018-08-29	AS	1201	ORD	SFO	800	810.0	10.0	37.0	847.0	...	

10 rows × 25 columns

```
In [150]: 1 df1[df1['Cluster_KM_PCA']==0].sort_values(by='LATE_AIRCRAFT_DELAY',ascending=False)[:10]
```

Out[150]:

	FL_DATE	OP_CARRIER	OP_CARRIER_FL_NUM	ORIGIN	DEST	CRS_DEP_TIME	DEP_TIME	DEP_DELAY	TAXI_OUT	WHEELS_OFF	...	CRS_ELAPSED
9803	2018-05-25	NK	563	ORD	SAN	1030	556.0	1166.0	15.0	611.0	...	
7428	2018-05-02	AA	1522	ORD	SFO	2030	1400.0	1050.0	27.0	1427.0	...	
8495	2018-05-14	NK	245	ORD	LAS	1531	726.0	955.0	14.0	740.0	...	
7497	2018-05-03	UA	2194	ORD	LAX	645	2123.0	878.0	16.0	2139.0	...	
27818	2018-11-25	AA	1244	ORD	SAN	2023	1152.0	929.0	18.0	1210.0	...	
21154	2018-08-31	NK	441	ORD	SEA	2030	1008.0	818.0	53.0	1101.0	...	
27996	2018-11-26	UA	761	ORD	SFO	800	1906.0	666.0	33.0	1939.0	...	
23261	2018-09-25	UA	483	ORD	PHX	2128	820.0	652.0	18.0	838.0	...	
29062	2018-12-07	UA	761	ORD	SFO	816	1855.0	639.0	22.0	1917.0	...	
18636	2018-08-09	UA	810	ORD	SFO	1245	11.0	686.0	21.0	32.0	...	

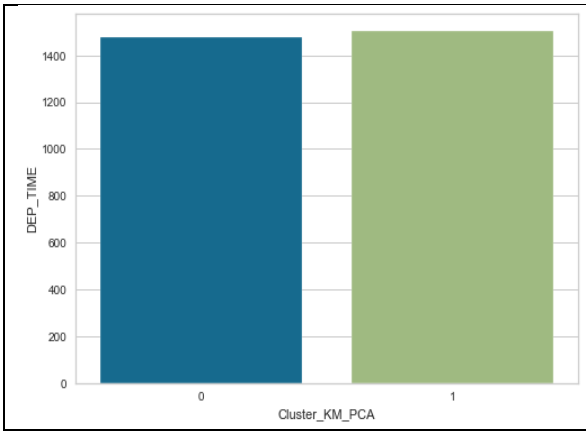
10 rows × 25 columns

```
In [147]: 1 df1[df1['Cluster_KM_PCA']==0].sort_values(by='WEATHER_DELAY',ascending=False)[:10]
```

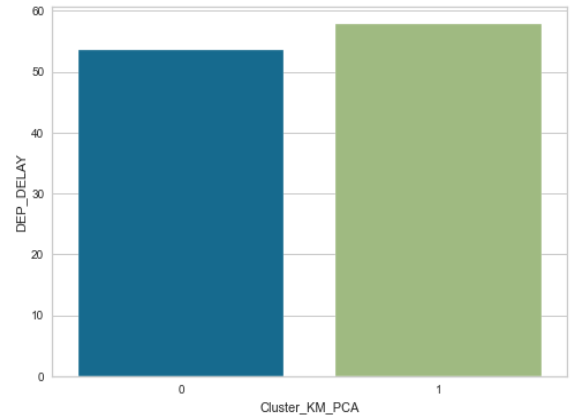
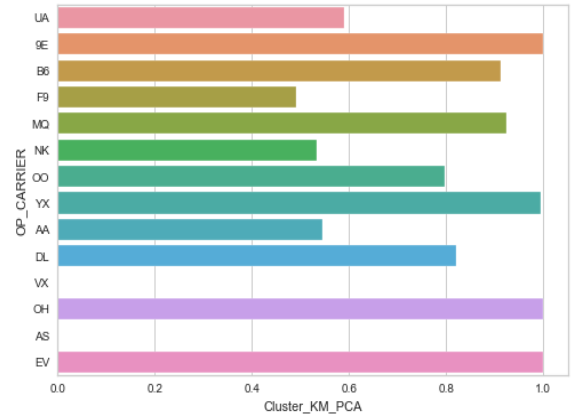
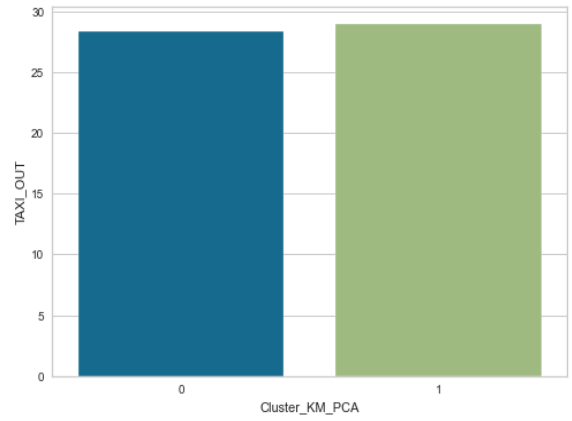
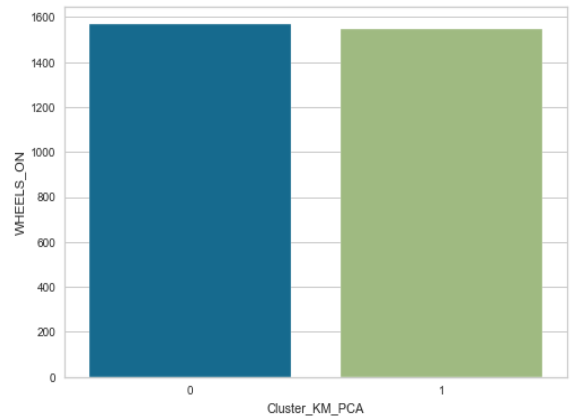
Out[147]:

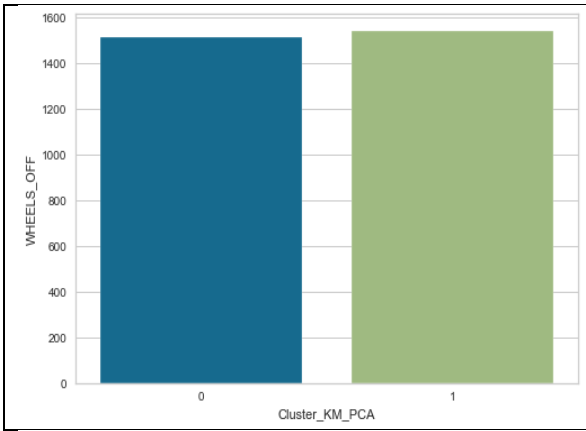
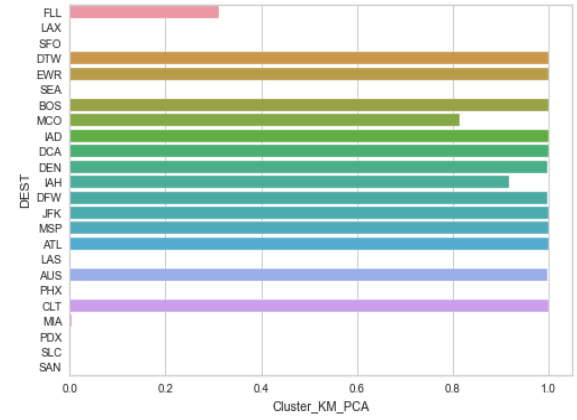
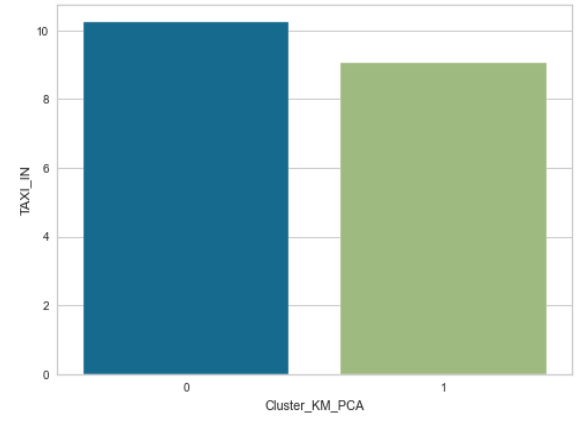
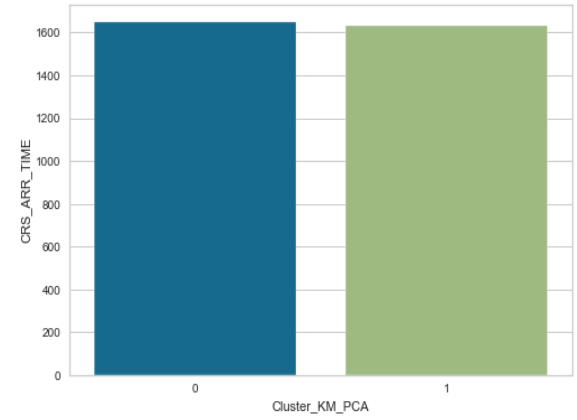
	FL_DATE	OP_CARRIER	OP_CARRIER_FL_NUM	ORIGIN	DEST	CRS_DEP_TIME	DEP_TIME	DEP_DELAY	TAXI_OUT	WHEELS_OFF	...	CRS_ELAPSED
13702	2018-06-26	DL	2863	ORD	SEA	1750	1143.0	1073.0	9.0	1152.0	...	
28041	2018-11-26	AA	2374	ORD	SFO	2035	1236.0	961.0	19.0	1255.0	...	
21323	2018-09-01	AA	2477	ORD	PDX	2026	1048.0	862.0	22.0	1110.0	...	
8727	2018-05-14	UA	439	ORD	LAS	2120	935.0	735.0	25.0	1000.0	...	
12270	2018-06-18	UA	1462	ORD	SFO	1957	713.0	676.0	33.0	746.0	...	
23778	2018-10-01	UA	2180	ORD	SLC	2127	842.0	675.0	18.0	900.0	...	
24093	2018-10-05	UA	1730	ORD	LAX	2034	753.0	679.0	42.0	835.0	...	
24107	2018-10-05	UA	1186	ORD	LAX	2112	637.0	565.0	30.0	707.0	...	
12342	2018-06-19	UA	1462	ORD	SFO	1957	842.0	765.0	12.0	854.0	...	
19323	2018-08-15	UA	483	ORD	PHX	2132	653.0	561.0	15.0	708.0	...	

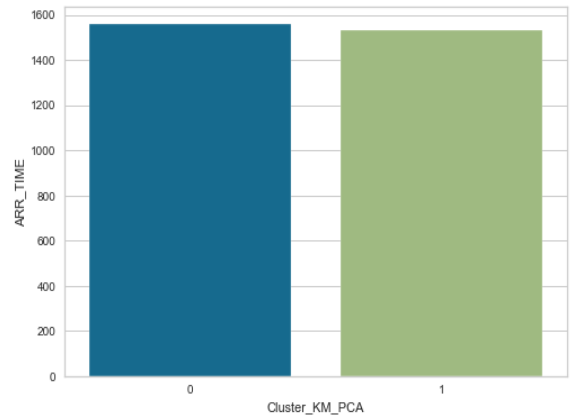
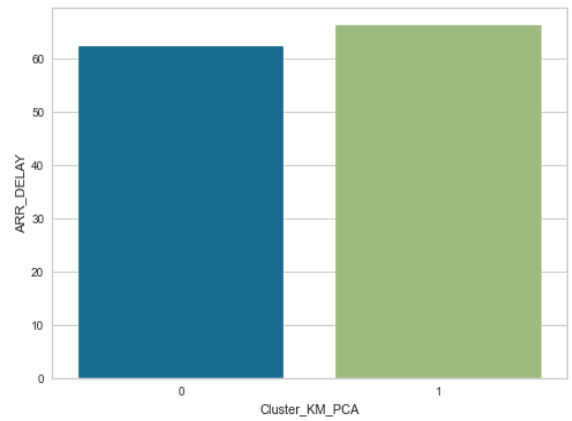
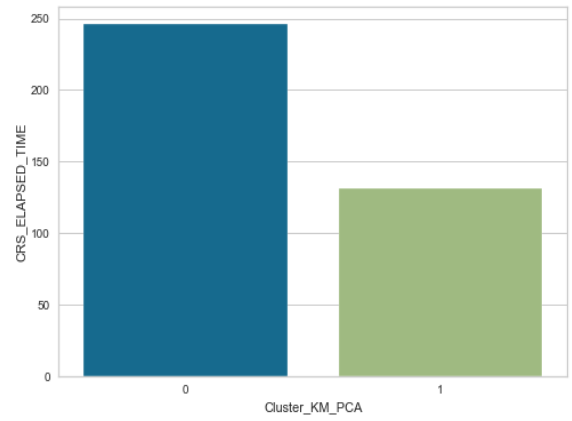
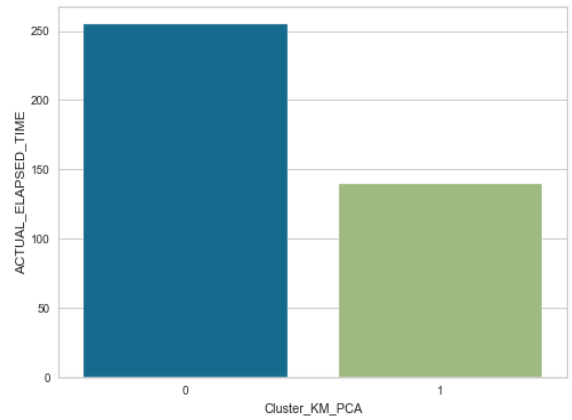
10 rows × 25 columns

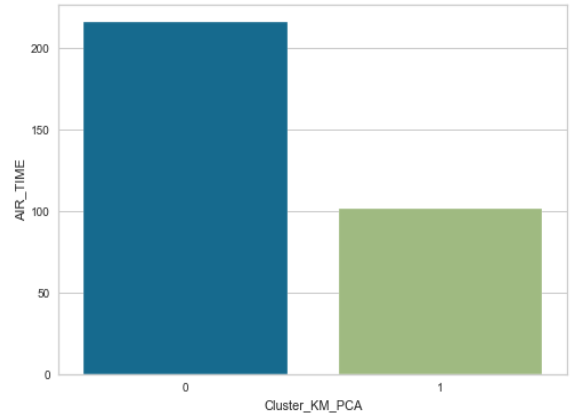
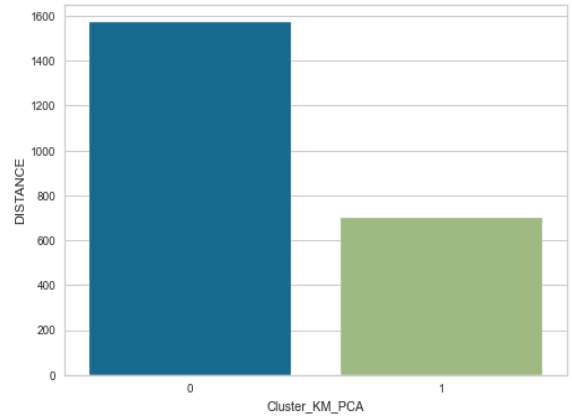
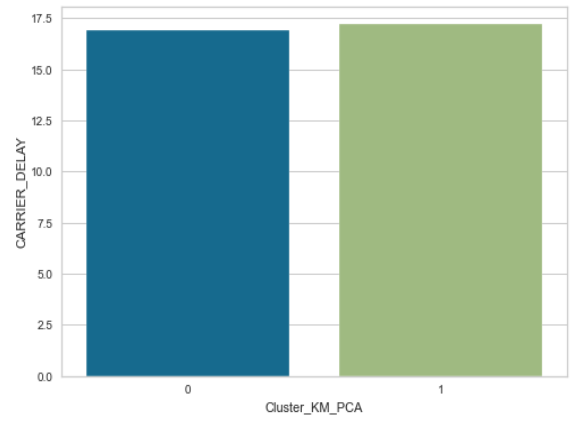
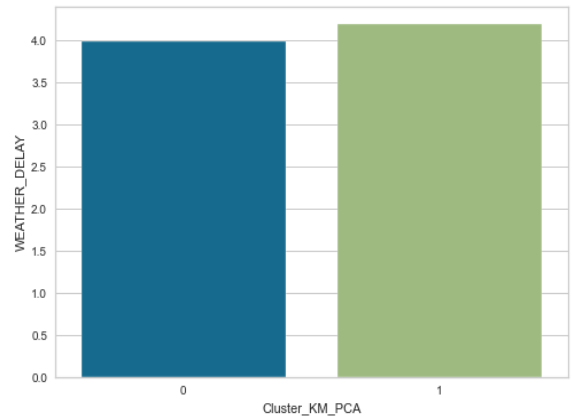


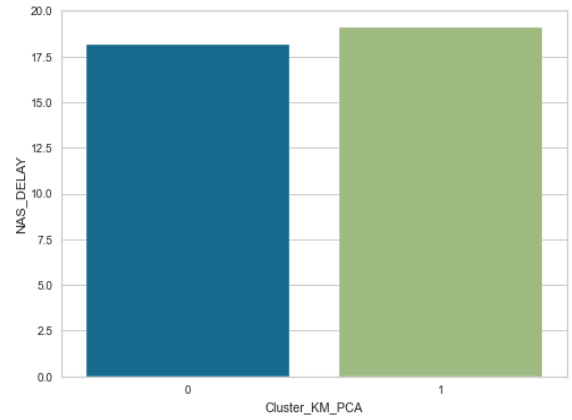
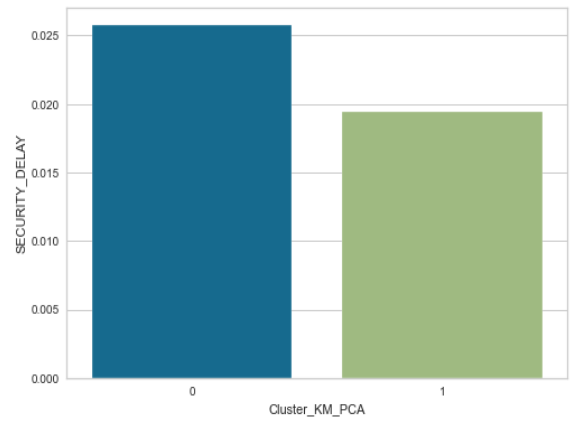
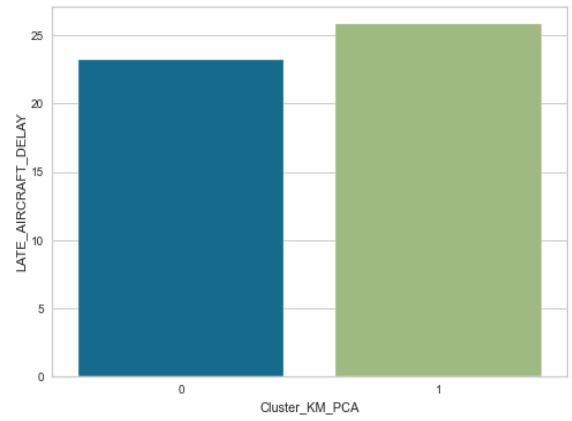
- The value of Departure time high in cluster 1 compared to cluster 0

	• The value of Departure delay is high in cluster 1 compared to cluster 0.
	• Endeavor Air, Republic Airways, Atlantic Southeast airline, Comair has the highest value, whereas Alaska airlines has lowest value.
	• The value for taxi_out is higher in Cluster 1 than cluster 0.
	• The value for Wheels_on is slightly higher in Cluster 0 than cluster 1

	• The value for Wheels_on is slightly higher in Cluster 1 than cluster 0.
	• Most of the destination has high value, except few places like San Francisco, Seattle, Los Angeles, San Diego, Miami, Portland International Airport etc.
	• Taxi_in is high in cluster 0, compared to cluster 1.
	• The CRS_Arr_time is distributed almost equal in both

	• The Arrival time is higher in cluster 1 as expected.
	• The arrival delay is higher in cluster 1.
	• The CRS_ELAPSED_TIME is quite high in cluster 0.
	• Subsequently, the ACTUAL_ELAPSED_TIME is high in cluster 0.

	• The arrival time in cluster 0 is almost double of cluster 1.
	• The distance is also double in cluster 0, compared to that of cluster 1.
	• The carrier delay is slightly higher in cluster 1 compared to cluster 0.
	• The weather delay is slightly higher in cluster 1 compared to cluster 0.

	• The NAS delay is slightly higher in cluster 1 compared to cluster 0.
	• The security delay is slightly higher in cluster 0 compared to cluster 1.
	• The late aircraft delay is higher in cluster 1 compared to cluster 0.

- Based on the above inference, the value for various features differs based on that cluster.
- Among the various delay, it shows the security delay is high in cluster 0, compared to other delays.
- The other delays are high in cluster 1.
- It is evident that the security delay is the one which is has no influence, but can be managed by the airport authority.
- The cluster 1 has such kind of delays which is dependent on some factor. We can say they are more like causation delay.
- Cluster 0 can be manged by certain operational efficiency and technological advancement. But cluster 1 requires business understanding and well-built strategic plan.

Techniques that can be implemented in business:

- Implement reduce miles-in-trail restrictions, based on more accurate weather data and on how restrictions previously performed for every route and air traffic control center. In layman's terms, that means air traffic controllers at O'Hare and other high-traffic airports needs start to reduce the spacing between planes arriving at airport when weather is an issue.
- Theoretically that should allow more planes to land in a shorter time span, but in reality, just how much air traffic controllers can constrict the spacing between planes will likely depend on just how severe weather issues may be.
- This will shorten the duration of time any weather-related slowdown in arriving planes is in effect — again with the goal of reducing air traffic delays.
- Various operational enhancements will boost system capacity in the short term but will not increase supply enough to provide lasting relief in the face of rapidly growing demand.
- Efforts to reduce congestion and delays by establishing slot quotas are also short-term approaches to the long-term problem.
- Economic techniques for managing demand do attempt to bring demand in line with existing capacity, and a combination of peak/off-peak landing fees and passenger surcharges might significantly reduce congestion and delays at the airport.
- Relieve the pressure on overutilized hub airports by shifting connecting traffic to other regional facilities. Most hub airports play a dual role handling connecting traffic and passengers originating/terminating in that city. Wayports could be built, which will be helpful enough.

Research Limitation:

- The data is limited to an individual origin airport-Chicago O'Hare International airport.
- The selective Destination airports are chosen, considering the top busiest ones.
- The affinity in Agglomerative clustering technique is done only using Euclidean. Manhattan could be considered while the datasets have outliers.
- Results of both clustering would impact based on the initial seeds.
- Kmean is very sensitive to scaling and Agglomerative is sensitive to outliers.
- Extensive Domain knowledge would be required to analyse the features in-depth.

9.REFERENCE:

1. https://analyticsindiamag.com/crisp-dm-data-science-project/
2. https://stat-or.unc.edu/wp-content/uploads/sites/182/2018/09/Paper3_MSOM_2012_AirlineFlightDelays.pdf
3. https://abcnews.go.com/Travel/story?id=3971303&page=1
4. https://www.bizjournals.com/chicago/news/2019/02/21/faa-moves-to-reduce-weather-related-flight-delays.html
5. http://onlinepubs.trb.org/Onlinepubs/trr/1989/1218/1218-001.pdf
6. https://www.justice.gov/atr/comments-congestion-and-delay-reduction-chicago-ohare-international-airport
7. https://www.bloomberg.com/news/articles/2020-08-27/faa-proposes-1-57-million-penalty-on-chicago-for-unsafe-runway
8. https://www.freightwaves.com/news/fed-up-with-cargo-congestion-freight-forwarders-flee-ohare-airport