Using Airlines Data to Predict Flights Delay

Group 10

Ashish Sharma
Pranjal Yadav

857-869-4602 (Ashish Sharma)
857-498-7241 (Pranjal Yadav)

sharma.ashis@northeastern.edu
yadav.pran@northeastern.edu

Percentage of effort Contributed by Ashish Sharma : 50%

Percentage of effort Contributed by Pranjal Yadav : 50%

Signature of Student 1 : Ashish Sharma

Signature of Student 2 : Pranjal Yadav

Submission Date : 04/18/2022

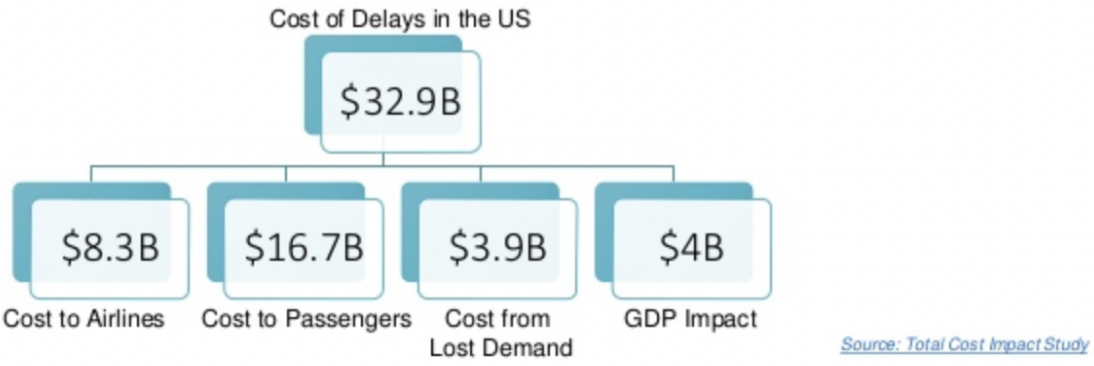
Section Overview

- **Problem Setting**
- **Problem Definition**
- **Data Sources**
- **Data Description**
- **Data Exploration and Data mining tasks**
- **Data Mining Tasks**
- **Data Mining Models/Methods**
- **Performance Evaluation**
- **Project Results**
- **Impact of the Project Outcomes**

Problem Setting :

The Aviation industry currently has 709,554 workers employed in its ranks. Safety and being on time are the 2 rules to live by for any airline company. The two fastest emerging technologies Artificial intelligence and machine learning have started being adopted in a widespread manner in Aviation and these are now taking over the previously existing method of making decisions. We Identified biggest pain point in the aviation Industry by far that is flight delays. We aim to tackle the issue of flight delays and the financial losses caused by it head on through our efforts of accurately predicting time delay. According to a study done by UC Berkeley academics, domestic flight delays cost the US economy $32.9 billion, with airline passengers bearing nearly half of the cost.

One of the most startling data is that nearly 2 million flights operated by major airlines did not arrive on time in 2015 due to delays, cancellations, or diversions. This equates to over a quarter of the operations reported by these airlines. Imagine any other company informing its consumers that the service they paid for is only available 27% of the time. This is a major issue that is unique to this industry.



Problem Definition :

Currently airline negligence is at an all-time high. The aviation domain is vast and has n-number of challenges like logistics, adverse weather conditions, flight scheduling mismanagement, human error etc. All of us at some point have been left agonized and stranded at an airport in our life. Countless resources and man hours have been lost due to flight delays and cancellations, In some emergency cases it has also led to loss of precious human lives.

We want to explore the hidden factors in our dataset that directly correlate to delays in flight departure. In our project we want to undertake a novel approach to build models to achieve correct predictions about all underlying aspects of flight delays.

Data Source:

- The data has been taken from **Kaggle** and the U.S. **Department of Transportation's** (DOT).
- Dataset contains **31 columns** and up to **1,936,758 different internal flights** in the US for 2015 and their causes for delay, diversion and cancellation; if any.
- https://www.kaggle.com/suryanarayanan02/flight-delay-prediction-dataset

Data Description:

We are using a flight details dataset to aid us in the process of flight delay prediction for different airlines. We are using the following number of rows and attributes with attributes names such as: Month, Date, Flight Number, Origin_Airport, Destination_Airport, Scheduled Arrival, Scheduled Departure, Taxi_in , Taxi_out etc.:

No. of attributes : 30

No. of rows : 1936758

	Unnamed: 0	Year	Month	DayofMonth	DayOfWeek	DepTime	CRSDepTime	ArrTime	CRSArrTime	UniqueCarrier	...	TaxiIn	TaxiOut
0	0	2008	1	3	4	2003.0	1955	2211.0	2225	WN	...	4.0	8.0
1	1	2008	1	3	4	754.0	735	1002.0	1000	WN	...	5.0	10.0
2	2	2008	1	3	4	628.0	620	804.0	750	WN	...	3.0	17.0
3	4	2008	1	3	4	1829.0	1755	1959.0	1925	WN	...	3.0	10.0
4	5	2008	1	3	4	1940.0	1915	2121.0	2110	WN	...	4.0	10.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
1936753	7009710	2008	12	13	6	1250.0	1220	1617.0	1552	DL	...	9.0	18.0
1936754	7009717	2008	12	13	6	657.0	600	904.0	749	DL	...	15.0	34.0
1936755	7009718	2008	12	13	6	1007.0	847	1149.0	1010	DL	...	8.0	32.0
1936756	7009726	2008	12	13	6	1251.0	1240	1446.0	1437	DL	...	13.0	13.0
1936757	7009727	2008	12	13	6	1110.0	1103	1413.0	1418	DL	...	8.0	11.0

1936758 rows × 30 columns

Data Exploration

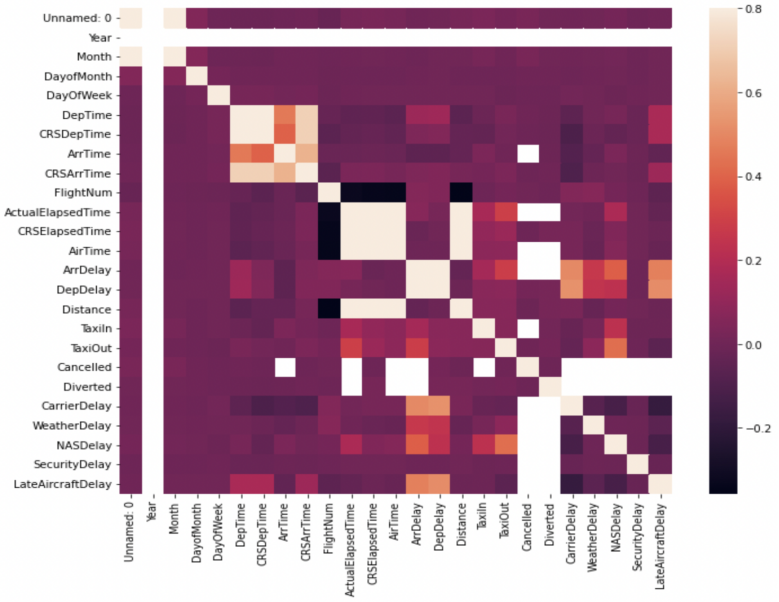
- We started with searching for any **missing values** in the attributes of the dataset.
- Then we moved to finding the **Correlation** between the attributes, as it will not contribute much in the prediction.

```
Dropping the columns which as high amount of null values and high correlation values.
```

```
flights = flights.drop("Unnamed: 0",1) #Empty
flights = flights.drop("Year",1) #Converted to date
flights = flights.drop("DayofMonth",1) #Converted to date
flights = flights.drop("DepTime",1) #Of the departure data we only keep the expected
flights = flights.drop("DepDelay",1)
flights = flights.drop("ArrTime",1)
flights = flights.drop("CRSArrTime",1)
flights = flights.drop("ActualElapsedTime",1)
flights = flights.drop("CRSElapsedTime",1)
flights = flights.drop("Diverted",1)
flights = flights.drop("Cancelled",1)
flights = flights.drop("Distance",1)
flights = flights.drop("FlightNum",1)
flights = flights.drop("TailNum",1)
```

```
flights.dropna()
```

```
Unnamed: 0                0
Year                      0
Month                     0
DayofMonth                0
DayOfWeek                 0
DepTime                   0
CRSDepTime                0
ArrTime                7110
CRSArrTime                0
UniqueCarrier             0
FlightNum                 0
TailNum                   5
ActualElapsedTime      8387
CRSElapsedTime          198
AirTime                8387
ArrDelay               8387
DepDelay                  0
Origin                    0
Dest                      0
Distance                  0
TaxiIn                 7110
TaxiOut                 455
Cancelled                 0
CancellationCode          0
Diverted                  0
CarrierDelay         689270
WeatherDelay         689270
NASDelay             689270
SecurityDelay        689270
LateAircraftDelay    689270
```

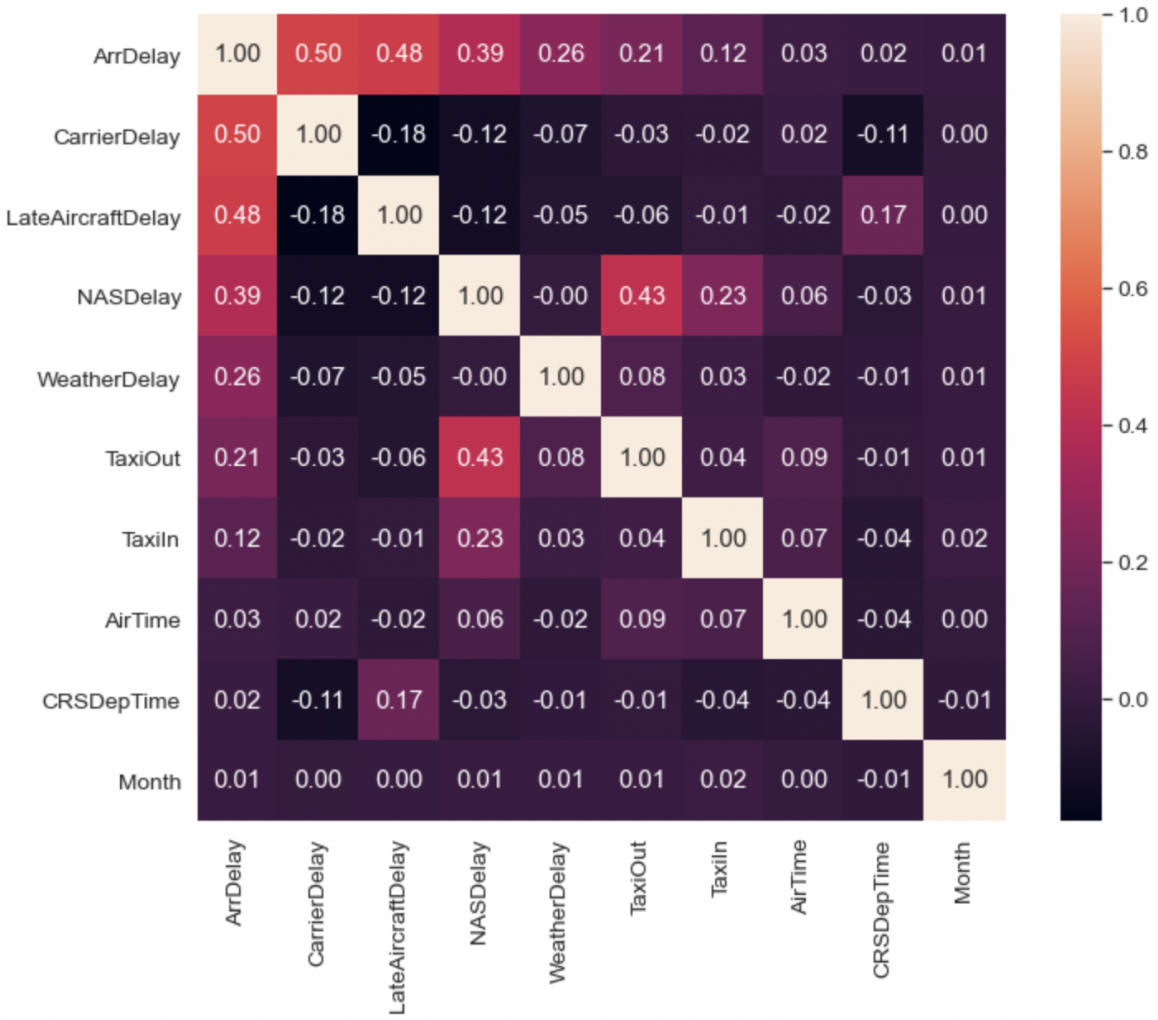


- The data had few attributes with a high **number of null values**, so we dropped as it will hamper the predictions of the dataset.
- Also, the attributes with correlation with **0.9 and above** have been removed.
- With further analysis, columns with less relevance towards our prediction goal are dropped such as the ‘Cancellation Code’, ‘Canceled’ columns

```
flights1 = pd.read_csv("flights (1).csv")
```

```
This the new updated dataset after cleaning the data.
```

```
delcorrmat = flights1.corr()

k = 10 #number of variables for heatmap
f, ax = plt.subplots(figsize=(12, 9))
cols = delcorrmat.nlargest(k, 'ArrDelay')['ArrDelay'].index
cm = np.corrcoef(flights1[cols].values.T)
sns.set(font_scale=1.25)
hm = sns.heatmap(cm, cbar=True, annot=True, square=True, fmt='.2f', annot_kws={'size': 15}, yticklabels=cols.values, xt
plt.show()
```



	Month	DayOfWeek	CRSDepTime	UniqueCarrier	AirTime	ArrDelay	Origin	Dest	TaxiIn	TaxiOut
0	1	4	1755	WN	77.0	34.0	IND	BWI	3.0	10.0
1	1	4	1830	WN	230.0	57.0	IND	LAS	3.0	7.0
2	1	4	1510	WN	107.0	80.0	IND	MCO	6.0	8.0
3	1	4	1425	WN	213.0	15.0	IND	PHX	7.0	8.0
4	1	4	1255	WN	110.0	16.0	IND	TPA	4.0	9.0
...	...	...	...	...	...	...	...	...	...	...
1247483	12	6	830	DL	82.0	64.0	ATL	PBI	8.0	21.0
1247484	12	6	1520	DL	27.0	17.0	HSV	ATL	9.0	7.0
1247485	12	6	1220	DL	120.0	25.0	MSP	ATL	9.0	18.0
1247486	12	6	600	DL	78.0	75.0	RIC	ATL	15.0	34.0
1247487	12	6	847	DL	122.0	99.0	ATL	IAH	8.0	32.0

1247488 rows × 16 columns

- The updated dataset consists of **16 Columns** and around **1.2 million** flight information as a fresh and clean dataset.

- For prediction, we considered the Target Variable as **"ArrDelay"** as it has the values containing the minutes the airline delayed.

- For data exploration, we visualized the reasons for flight arrival delays according to various reasons for the delay such as LateAircraft Delay, Weather Delay , NAS Delay, Carrier Delay, Security Delay.

- After the data exploration analysis, we concluded that LateAircraftDelay was the largest contributor in Arrival Delay of flights.

How are these categories defined?

Air Carrier: This is the reason for cancellation or delay because of issues like maintenance or crew problems, aircraft cleaning, baggage loading, fueling, etc that are out of the control of airlines.

Extreme Weather: extreme meteorological circumstances (actual or forecasted) occurring, due to which the operation of flights gets delayed or sometimes even canceled. These circumstances are out of the control of airlines and cause huge losses.

National Aviation System (NAS): Delays and cancellations caused by the national aviation system, which encompass a wide range of factors such as non-extreme weather, airport operations, high traffic volume, and air traffic management.

Late-arriving aircraft: This occurs when the flight with the same aircraft that was supposed to arrive gets delayed leading to the delay of the next series of flights.

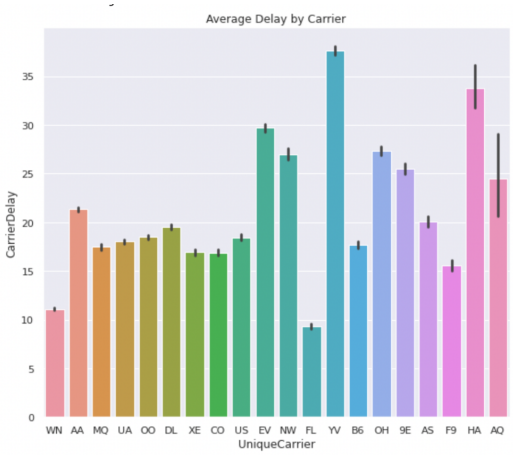
Security: Delays or cancellations caused by a terminal or concourse evacuation, re-boarding of aircraft due to a security breach, malfunctioning screening equipment, and/or large lineups of more than 29 minutes at screening locations.

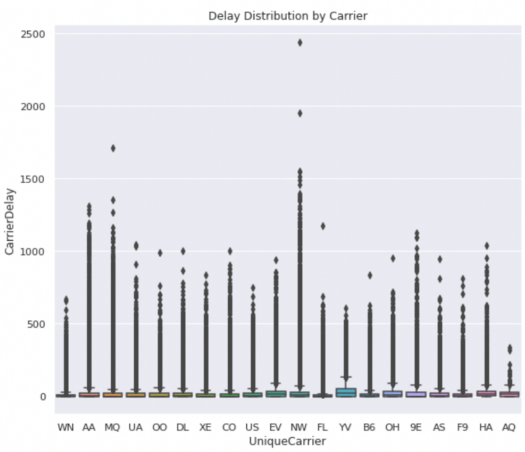
- Next, we started evaluating the delay of flights by individual Carriers and the delay distribution of Individual Carrier.

```
print(flights1['UniqueCarrier'].value_counts())
```

```
WN    203559
AA    132257
MQ     97555
UA     95465
OO     88991
DL     72252
XE     72008
US     59508
CO     58958
EV     56781
NW     54744
YV     50646
FL     46991
OH     39293
B6     38232
9E     35631
AS     24012
F9     15940
HA      4325
AQ       340
Name: UniqueCarrier, dtype: int64
```

Through the visualization we can clearly see that Late-arriving aircraft is the key driving factor in flight delays. We are also able to visualize that weather delay only contributes to a very small amount of flight delays, which contradicts our initial assumption. This gives Airlines insightful information on which factors to focus on while mitigating delays and helps them prioritize their resources for maximum efficiency.



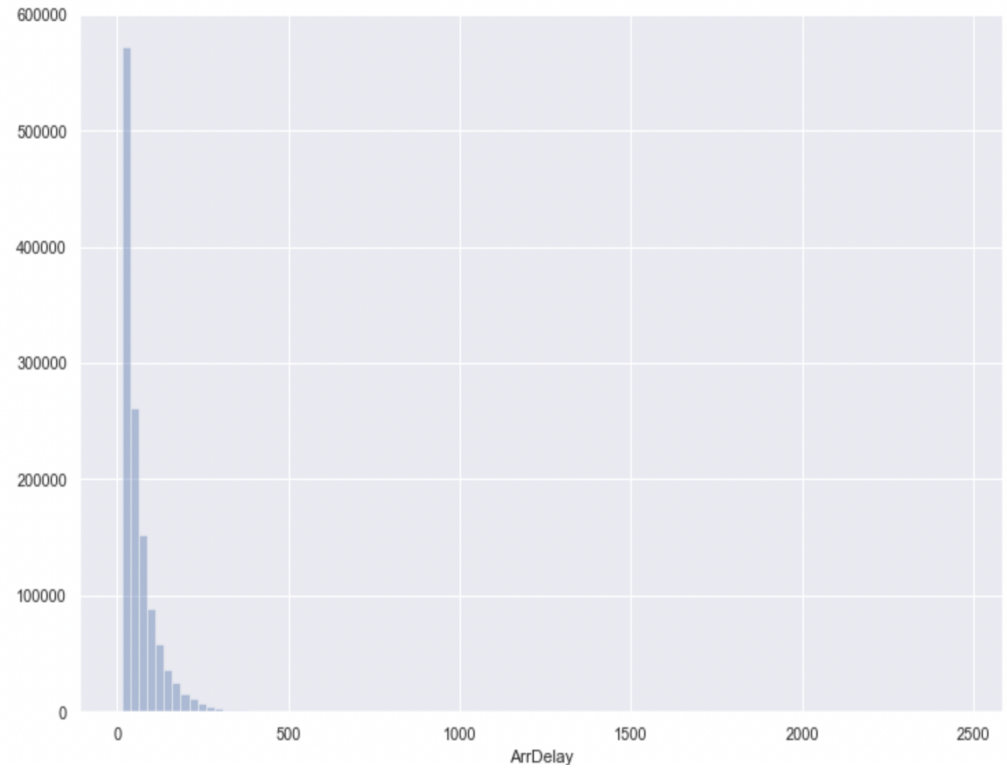



```
f,ax=plt.subplots(1,2,figsize=(20,8))
sns.barplot('UniqueCarrier','CarrierDelay', data=flights1,ax=ax[0], order=['WN', 'AA', 'MQ', 'UA','OO','DL','XE','CO',
                                                                           'US','EV', 'NW','FL','YV', 'B6','OH','9
                                                                            'AS','F9','HA','AQ'])
ax[0].set_title('Average Delay by Carrier')
sns.boxplot('UniqueCarrier','CarrierDelay', data=flights1,ax=ax[1], order=['WN', 'AA', 'MQ', 'UA','OO','DL','XE','CO',
                                                                           'US','EV', 'NW','FL','YV', 'B6','OH','9
                                                                            'AS','F9','HA','AQ'])
ax[1].set_title('Delay Distribution by Carrier')
plt.close(2)
plt.show()

print(['WN: Southwest Airlines', 'AA: American Airlines', 'MQ: American Eagle Airlines', 'UA: United Airlines',
       'OO: Skywest Airlines','DL: Delta Airlines','XE: ExpressJet','CO: Continental Airlines','US: US Airways',
       'EV: Atlantic Southeast Airlines', 'NW: Northwest Airlines','FL: AirTran Airways','YV: Mesa Airlines',
       'B6: JetBlue Airways','OH: Comair','9E: Pinnacle Airlines','AS: Alaska Airlines','F9: Frontier Airlines',
       'HA: Hawaiian Airlines','AQ: Aloha Airlines'])
```

We can clearly understand the average arrival delay of any specific flight carrier by this visualization, such as Northwest Airlines, Southwest Airlines, United Airlines, and Mesa Airlines, which have the biggest average flight delay. Southwest Airlines (WN), American Eagle Airlines (MQ), United Airlines (UA), and Skywest Airlines (OO) are four of the top five domestic airlines with average delays below the mean (19 minutes). Southwest Airlines has the second-lowest flying time of all the airlines, at 11.7 minutes every flight.

We carry out the visualization for delay distribution by individual carriers. We find outliers and through those outliers we take a close look at the meteorological data and find out that the large delay in a single instance was due to a very heavy snowstorm in Chicago.

```
<AxesSubplot:xlabel='ArrDelay'>
```



Data Mining Tasks

- **Label Encoding.**

The Approach of label encoding is fundamental. We convert the textual values of the columns into numerical values to bridge the gap of machine access. Our dataset of flights have column names such as unique carriers. We performed a Data Label Encoder to convert the Text into Numeric. Preprocessing of the data we first did label encoding to convert the text data into numerical data to make it as the machine readable form. It is an important part of the preprocessing of the data.

```
#labeling text to numeric
from sklearn import preprocessing
from collections import defaultdict

# select text columns
cat_cols = flights1.select_dtypes(include='object').columns

# this is a way to apply label_encoder to all category cols at once, returning a label encoder per categorical column,
d = defaultdict(preprocessing.LabelEncoder)

 # transform all text columns to numbers
flights1[cat_cols] = flights1[cat_cols].apply(lambda x: d[x.name].fit_transform(x.astype(str)))
```

```
flights1
```

UniqueCarrier	AirTime	ArrDelay	Origin	Dest
WN	77.0	34.0	IND	BWI
WN	230.0	57.0	IND	LAS
WN	107.0	80.0	IND	MCO
WN	213.0	15.0	IND	PHX
WN	110.0	16.0	IND	TPA

UniqueCarrier	AirTime	ArrDelay	Origin	Dest
17	77.0	34.0	140	48
17	230.0	57.0	140	155
17	107.0	80.0	140	177
17	213.0	15.0	140	220
17	110.0	16.0	140	284

- **Standardization of the data**.

Standardization or Z-Score Normalization is the transformation of features by subtracting from mean and dividing by standard deviation. This is often called a Z-score.

$$X\_new = (X - mean)/Std$$

```
X = flights1.drop('ArrDelay',axis = 1)
X.shape
```

```
(1247488, 14)
```

```
y = flights1['ArrDelay']
y
```

```
0           34.0
1           57.0
2           80.0
3           15.0
4           16.0
            ...
1247483     64.0
1247484     17.0
1247485     25.0
1247486     75.0
1247487     99.0
Name: ArrDelay, Length: 1247488, dtype: float64
```

```
Ultimate goal to perform standardization is to bring down all the features to a common scale without distorting the
differences in the range of the values.
```

```
from sklearn.preprocessing import StandardScaler
scaler= StandardScaler()
scaler.fit(X)
scaled_data = scaler.transform(X)
X_stand = scaled_data
```

```
scaled_data.mean(axis = 0)
```

```
array([ 7.04272812e-16, -1.36060987e-16, -4.67054628e-17, -5.83248706e-17,
        5.51807955e-17, -1.00245871e-16,  4.62953660e-17,  1.85910525e-17,
       -1.19839383e-17, -2.55399140e-17,  1.76797264e-17, -5.22018983e-17,
       -4.46549790e-18,  5.17633226e-17])
```

Standardization can be helpful in cases where the data follows a Gaussian distribution. However, this does not have to be necessarily true. Geometrically speaking, it translates the data to the mean vector of original data to the origin and squishes or expands the points if standard deviation is 1 respectively.

We can see that we are just changing mean and standard deviation to a standard normal distribution which is still normal thus the shape of the distribution is not affected.

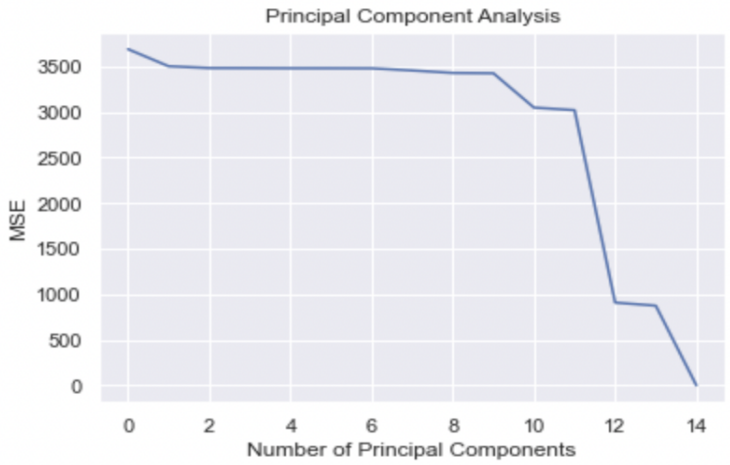
Standardization does not get affected by outliers because there is no predefined range of transformed features.

Ultimate goal to perform standardization is to bring down all the features to a **common scale without distorting the differences** in the range of the values.

- **Principal Components Analysis**

Principal components analysis (PCA) is a common method to summarize a larger set of correlated variables into smaller and more easily interpretable axes of variation.

```
pca = PCA()
X_reduced = pca.fit_transform(scale(X))

#define cross validation method
cv = RepeatedKFold(n_splits=10, n_repeats=3, random_state=1)

regr = LinearRegression()
mse = []

# Calculate MSE with only the intercept
score = -1*model_selection.cross_val_score(regr, np.ones((len(X_reduced),1)), y, cv=cv, scoring='neg_mean_squared_error
mse.append(score)

# Calculate MSE using cross-validation, adding one component at a time
for i in np.arange(1, 15):
    score = -1*model_selection.cross_val_score(regr, X_reduced[:,:i], y, cv=cv, scoring='neg_mean_squared_error').mean(
    mse.append(score)

# Plot cross-validation results
plt.plot(mse)
plt.xlabel('Number of Principal Components')
plt.ylabel('MSE')
plt.title('Principal Component Analysis')
```



In our data we saw that the principal components for our analysis were coming at 12 - 14 components to get the strong variance over 80 % in the data. This helped reduce a few more columns which were redundant and not impactful for performing models, as they will not be able to contribute to the increase of r squared value.

Data Mining Models/Methods:

- Data Modeling starts with the splitting of the data into 70% training set and 30% validation set.

- We are going to train and evaluate different machine learning models such as Linear Regression, KNN Regressor, Decision Tree Regressor, Lasso and Ridge Regression to compare models and find out the best fitting models.
- The analysis of the all the regression models are as follows

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.3,random_state = 2)
```

```
Trained and evaluated different machine learning models such as Linear Regression, KNN Regressor, Decision Tree
Regressor, Lasso and Ridge Regression to compare models
```

LinearRegression:

```
from sklearn import linear_model
from sklearn.metrics import mean_squared_error, r2_score
model = linear_model.LinearRegression()
model.fit(X_train, y_train)
```

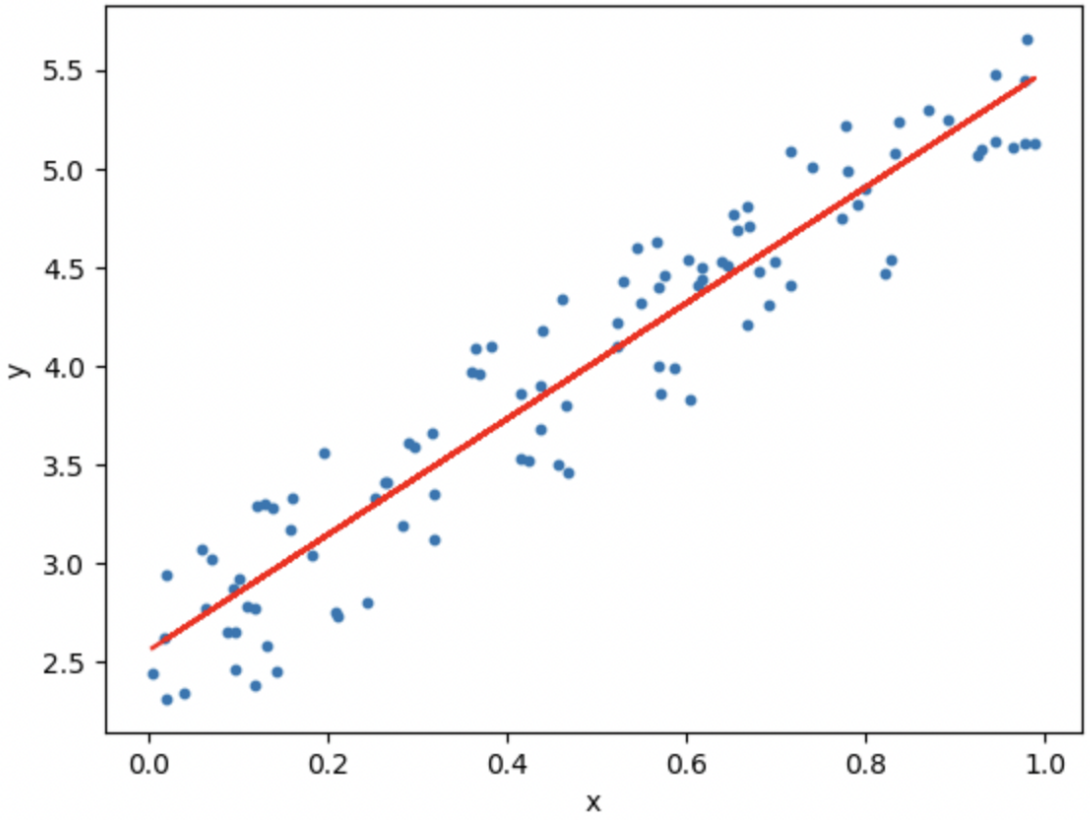
```
LinearRegression()
```

```
y_pred_train = model.predict(X_train)
print('Coefficients:', model.coef_)
print('Intercept:', model.intercept_)
print('Mean squared error (MSE): %.2f'
      % mean_squared_error(y_train, y_pred_train))
print('Coefficient of determination (R^2): %.2f'
      % r2_score(y_train, y_pred_train))
```

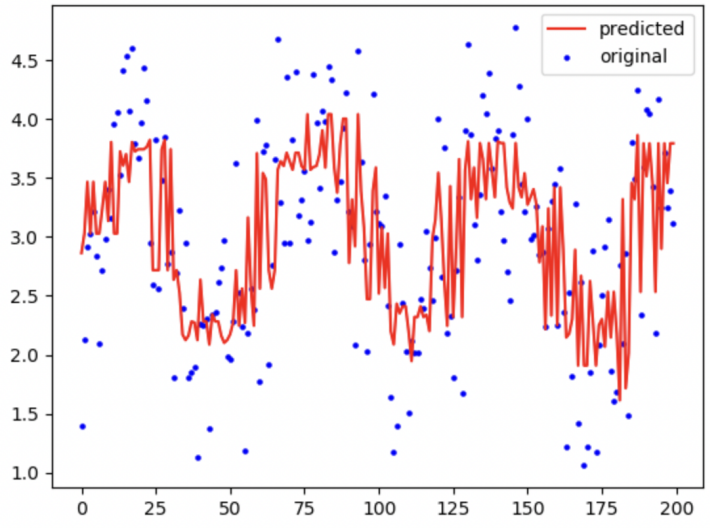
```
Coefficients: [-6.77693431e-15 -4.51028104e-15 -1.38777878e-17  1.62521906e-16
 -1.46800974e-16  1.90819582e-16 -1.21430643e-16 -6.24500451e-16
 -2.57649804e-15  1.00000000e+00  1.00000000e+00  1.00000000e+00
  1.00000000e+00  1.00000000e+00]
Intercept: 1.0658141036401503e-13
Mean squared error (MSE): 0.00
Coefficient of determination (R^2): 1.00
```

```
y_pred_test = model.predict(X_test)
print('Coefficients:', model.coef_)
print('Intercept:', model.intercept_)
print('Mean squared error (MSE): %.2f'
      % mean_squared_error(y_test, y_pred_test))
print('Coefficient of determination (R^2): %.2f'
      % r2_score(y_test, y_pred_test))
```

```
Coefficients: [-6.77693431e-15 -4.51028104e-15 -1.38777878e-17  1.62521906e-16
 -1.46800974e-16  1.90819582e-16 -1.21430643e-16 -6.24500451e-16
 -2.57649804e-15  1.00000000e+00  1.00000000e+00  1.00000000e+00
  1.00000000e+00  1.00000000e+00]
Intercept: 1.0658141036401503e-13
Mean squared error (MSE): 0.00
Coefficient of determination (R^2): 1.00
```



KNN regressor:

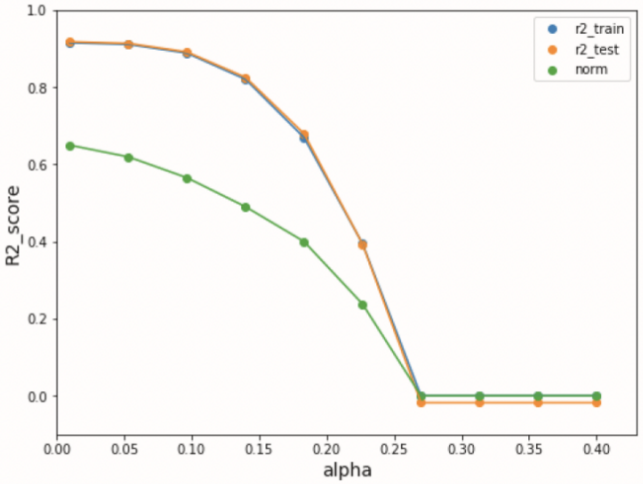


Lasso Regressor:

$$\hat{y}_i = w_0 + \sum_{j=1}^{m} X_{ij} w_j$$

$$J(w) = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 + \alpha \sum_{j=1}^{m} |w_j|$$

$$\|w\|^2 = \sum_{j=1}^{m} |w_j|^2$$



Performance Evaluation and Interpretation:

```
# multiple models with holdout validation
from sklearn import neighbors
from sklearn.metrics import *
mm = pd.DataFrame(columns=['t_model', 'rmse', 'mae','r-squared'])
index = 0
model = [linear_model.LinearRegression(),
              linear_model.Lasso(alpha=1),
              linear_model.Ridge(alpha=1),
              neighbors.KNeighborsRegressor(n_neighbors=3),
              DecisionTreeRegressor(criterion = "mse", max_depth=5)]
for t_o_m in model:
    t_o_m.fit(X_train, y_train)
    pred = t_o_m.predict(X_test)
    rmse = mean_squared_error(y_test, pred, squared=False)
    mae = mean_absolute_error(y_test, pred)
    rsquared = r2_score(y_test,pred)

    mm.loc[index] = [t_o_m, rmse, mae, rsquared]
    index+=1
mm
```

	t_model	rmse	mae	r-squared
0	LinearRegression()	1.498617e-13	1.148305e-13	1.000000
1	Lasso(alpha=1)	4.599679e-01	8.567211e-02	0.999942
2	Ridge(alpha=1)	5.275093e-07	9.978124e-08	1.000000
3	KNeighborsRegressor(n_neighbors=10)	1.253032e+01	7.665142e+00	0.957155
4	DecisionTreeRegressor(max_depth=5)	2.638108e+01	1.954397e+01	0.810086

Here, we worked on various algorithms. We can see that the various models like Linear Regression, Lasso, Ridge got overwhelming results. The models had an r squared value of 1.00 which is overfitting and misleading. The other selected models which are KNN regressor and Decision Tree, we further looked into the data to get more insights and better accuracy from the data. It is seen that the Decision Tree Regressor has good r-squared value (0.81) and low rmse value (26.3) but KNN regressor out performs the Decision Tree so we will be selecting KNN regressor as our best data modeling algorithm.

Implementation of Selected Models:

We saw that the selected model KNN regressor had r squared value of .957 .As per our previous results we obtained the accuracy of 95% with the rmse values in between k = 1 to 15.

Best rmse was for k=3, to take it further we chose n=3 in KNN regressor.

```
RMSE value for k=  1 is: 13.474427199535278
RMSE value for k=  2 is: 12.162806853892304
RMSE value for k=  3 is: 11.902870351013002
RMSE value for k=  4 is: 11.915855083757894
RMSE value for k=  5 is: 12.007654218385785
RMSE value for k=  6 is: 12.147557320025346
RMSE value for k=  7 is: 12.285178239289783
RMSE value for k=  8 is: 12.429696394781377
RMSE value for k=  9 is: 12.58478800225529
RMSE value for k=  10 is: 12.735440438042373
RMSE value for k=  11 is: 12.885841447125319
RMSE value for k=  12 is: 13.022713958601674
RMSE value for k=  13 is: 13.167248670892919
RMSE value for k=  14 is: 13.298954927105614
RMSE value for k=  15 is: 13.42811772542424
```

For n=3, our accuracy increased to 96% with the rmse score of 11.69 which was the least in all the regression models.

t_model	rmse	mae	r-squared
LinearRegression()	1.498617e-13	1.148305e-13	1.000000
Lasso(alpha=1)	4.599679e-01	8.567211e-02	0.999942
Ridge(alpha=1)	5.275093e-07	9.978124e-08	1.000000
KNeighborsRegressor(n_neighbors=3)	1.169378e+01	7.384141e+00	0.962685
DecisionTreeRegressor(max_depth=5)	2.639902e+01	1.954467e+01	0.809827

Project Results:

- The final performance of best models are evaluated with an unseen test data and also compared models created from good understanding of data

- We were able to get insights of Flight Delays occur and what are the reasons impacting the Delays

- We were able to find the best performed Regression Model which is **KNN Regressor** with **K value = 3** and **efficiency of 96.26%**

Impacts of the Project Outcomes:

- The basic idea of implementing the prediction model on the prediction of Flight Delays was reached, and the insights of various reasons for delays helped us to predict the model more efficiently .

- We got to learn more about the implementation of regression models and got to interpret the results of various models.

- The following models and results can be further implemented as the system in the airline industry to overcome the delays, and it will help airlines to move up in the game of becoming the best airlines in the world and losing money on the delays. They can also contribute better to the society by reducing the carbon footprints and can make a major environmental conservation.

References:

1] Breiman, L. (2001). Random forests. Machine learning, 45(1), 532.

[2] Breiman, L. (1996). Outofbag estimation (pp. 113). Technical report, Statistics Department, University of California Berkeley, Berkeley CA 94708, 1996b. 33, 34.

[3] Genuer, R., Poggi, J. M., & Tuleau Malot, C. (2010). Variable selection using random forests.
Pattern Recognition Letters, 31(14), 22252236.

[4] http://www.datawrangling.com/howflightcastersqueezespredictionsfromflightdata

[5] https://www.kaggle.com/datasets/suryanarayanan02/flight-delay-prediction-dataset