FLIGHTS DELAY PREDICTION

AI & ML on Python

Panda Projects

WeWork, Sector 15
Gurugram Haryana

Kunal Gehlot
Gehlotkunal4@outlook.com

Data Cleaning

Starting with the provided **flights.csv** file and import into a Pandas DataFrame, then convert the **SCHEDULED_DEPARTURE, DEPARTURE_TIME, WHEELS_OFF, SCHEDULED_ARRIVAL, WHEELS_ON, ARRIVAL_TIME** into HH:MM:SS format.
Then dropping the rows which are **DIVERTED** or **CANCELLED**.

Now by looking at the columns, the columns **CANCELLATION_REASON, AIR_SYSTEM_DELAY, SECURITY_DELAY, AIRLINE_DELAY, LATE_AIRCRAFT_DELAY, WEATHER_DELAY, YEAR, TAIL_NUMBER**

We observe that the column **YEAR** does not provide any significant relevance as the whole dataset of the year 2015.
TAIL_NUMBER does not signify anything as it is next to random.
The columns **CANCELLATION_REASON, AIR_SYSTEM_DELAY, SECURITY_DELAY, AIRLINE_DELAY, LATE_AIRCRAFT_DELAY** and **WEATHER_DELAY** has a high number of null blocks, so we drop these columns altogether.

Now we convert all the time values of 24:00:00 to 00:00:00 to turn make it sensible, then create new columns named **SCH_DEP_HR, SCH_DEP_MN, DEP_TM_HR, DEP_TM_MN, WHL_OFF_HR, WHL_OFF_MN, WHL_ON_HR, WHL_ON_MN, SCH_ARVL_HR, SCH_ARVL_MN, ARVL_TM_HR, ARVL_TM_MN** by splitting the columns **SCHEDULED_DEPARTURE, DEPARTURE_TIME, WHEELS_OFF, WHEELS_ON, SCHEDULED_ARRIVAL, ARRIVAL_TIME** into their respective hours and minutes and themselves.

Now we create two Database files namely Airline Database and Airport Database where we convert **AIRLINE, ORIGIN_AIRPORT, DESTINATION_AIRPORT** into hash values and preserving the hash values' meaning into **AirlineDatabase.csv** and **AirportDatabase.csv**, then we replace the above columns with their hash values and save this new Dataframe as **flightsNew.csv**

Data modelling

Starting with importing the previously created **flightsNew.csv** file, we ease our prediction process by changing the ARRIVAL_DELAY values to 1 if the delay is equal to or more than 15 minutes and 0 if the delay is less than 15 minutes.

Now we drop the columns **ARVL_TM_HR, ARVL_TM_MN, ARRIVAL_DELAY, ELAPSED_TIME, AIR_TIME, WHL_ON_HR, WHL_ON_MN** and **TAXI_IN** as they cannot be provided beforehand by the user to forecast a delay.

We keep the output data as a single columns of ARRIVAL_DELAY in 1s and 0s format as previously described and train the data using four different models and split the whole data in the 30-70 ratio for ideal testing and training of the models.

Linear Regression

Accuracy: 2%

Logistic Regression

Accuracy: 61%

KNeighborsClassifier

Accuracy:

- 3n - 82.4%
- 4n - 82.5%
- >60n - 81.4%

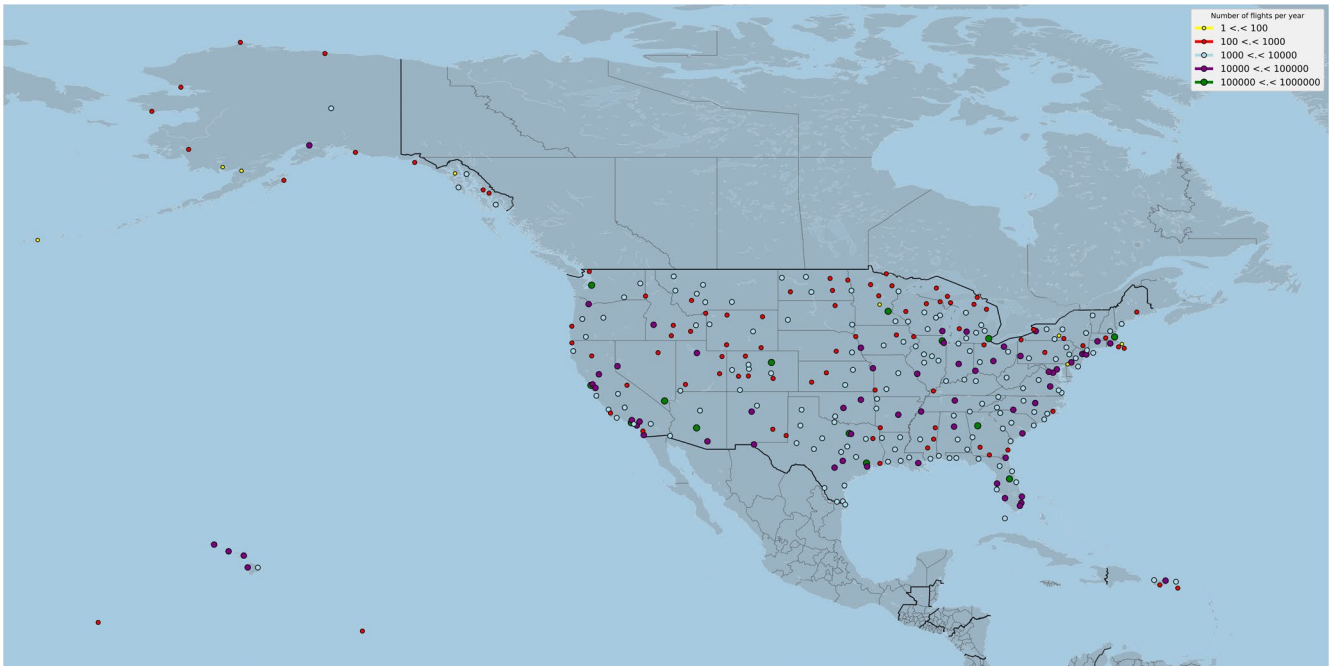
RandomForestClassifier

Accuracy: **92.87%**

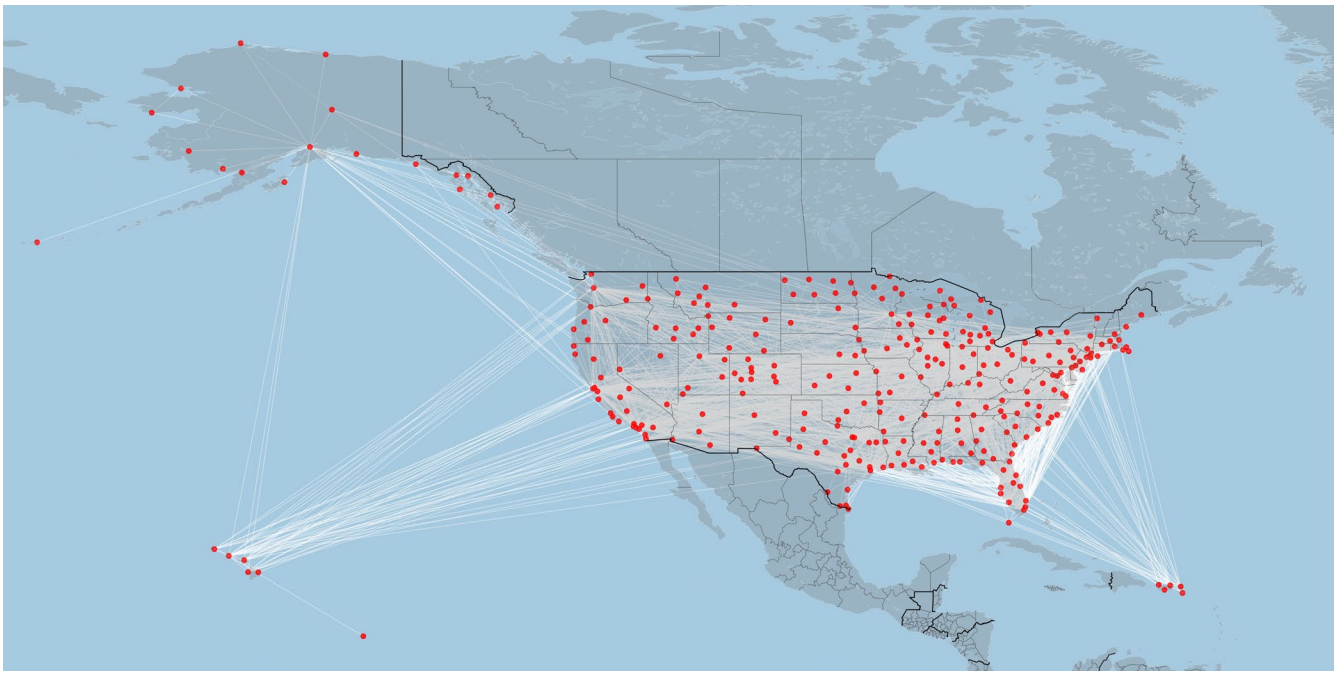
Thus making the RandomForestClassifier most useful Model to predict flight delays as there are 12 input variables and 1 output variable.

Data Visualization

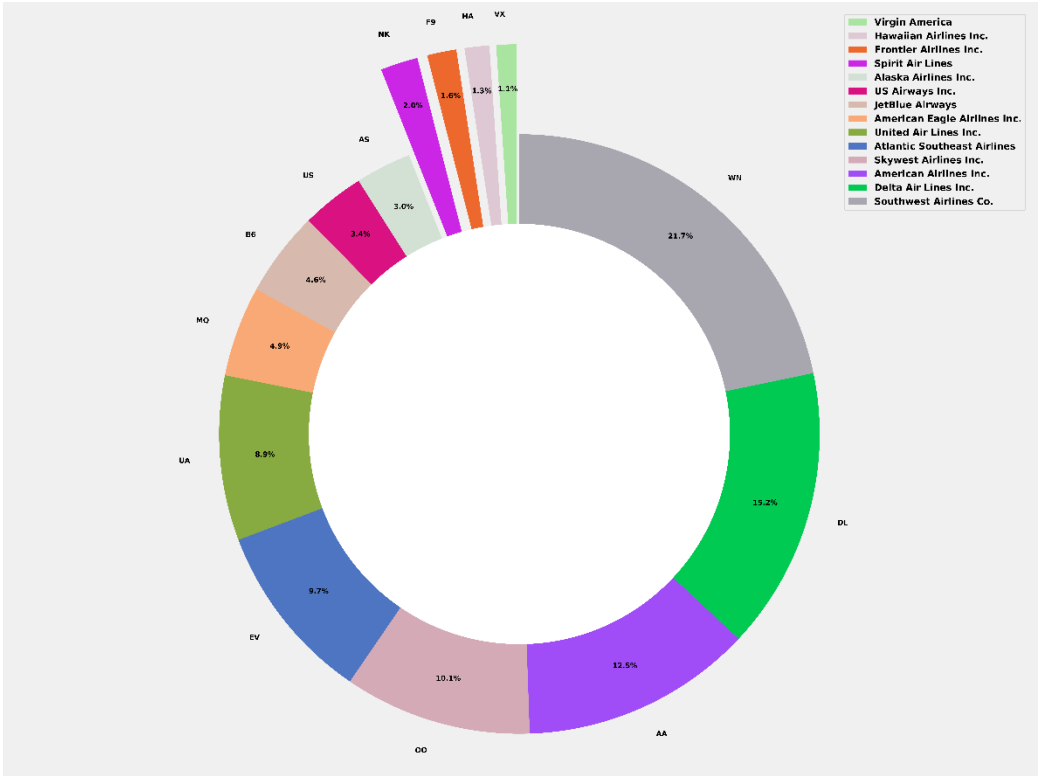
First, we try to visualize each and every airport of the data and number of flights taking from each one of them and grouping them accordingly on the map, with the help of matplotlib and base map libraries.



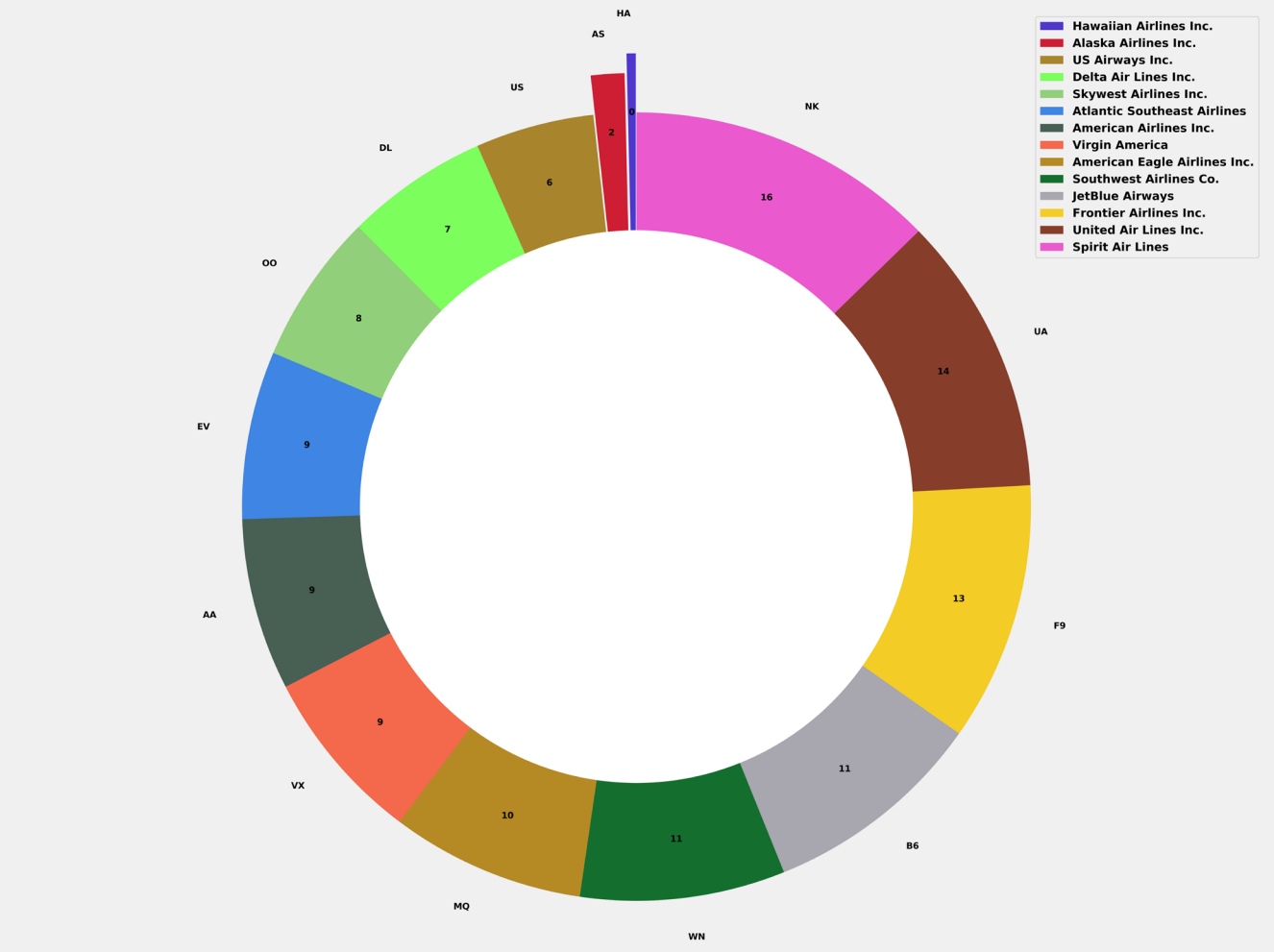
Then we connect airports to one another to visualize the flights taking off and landing from one place to another to understand the flights paths and densities at each airport.



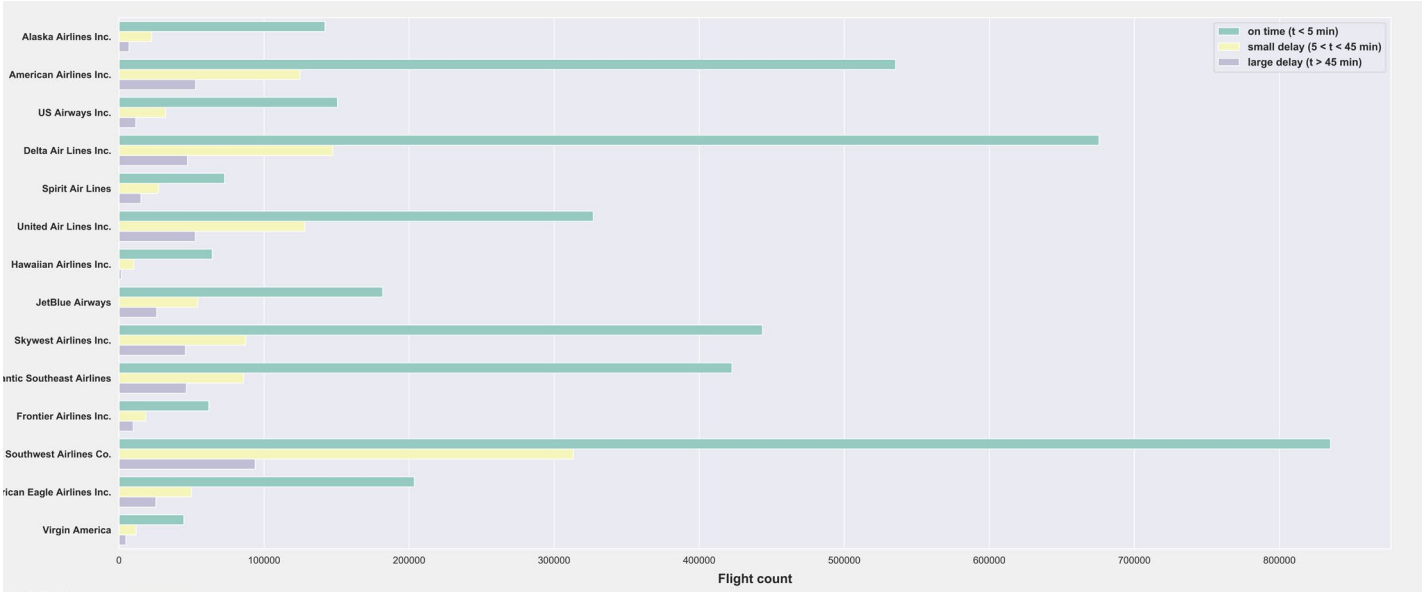
To understand the correlation between the Airline and number of flights, we sort the data into a count of flights and visualize to understand what percentage of the total flights taken does an airline has.



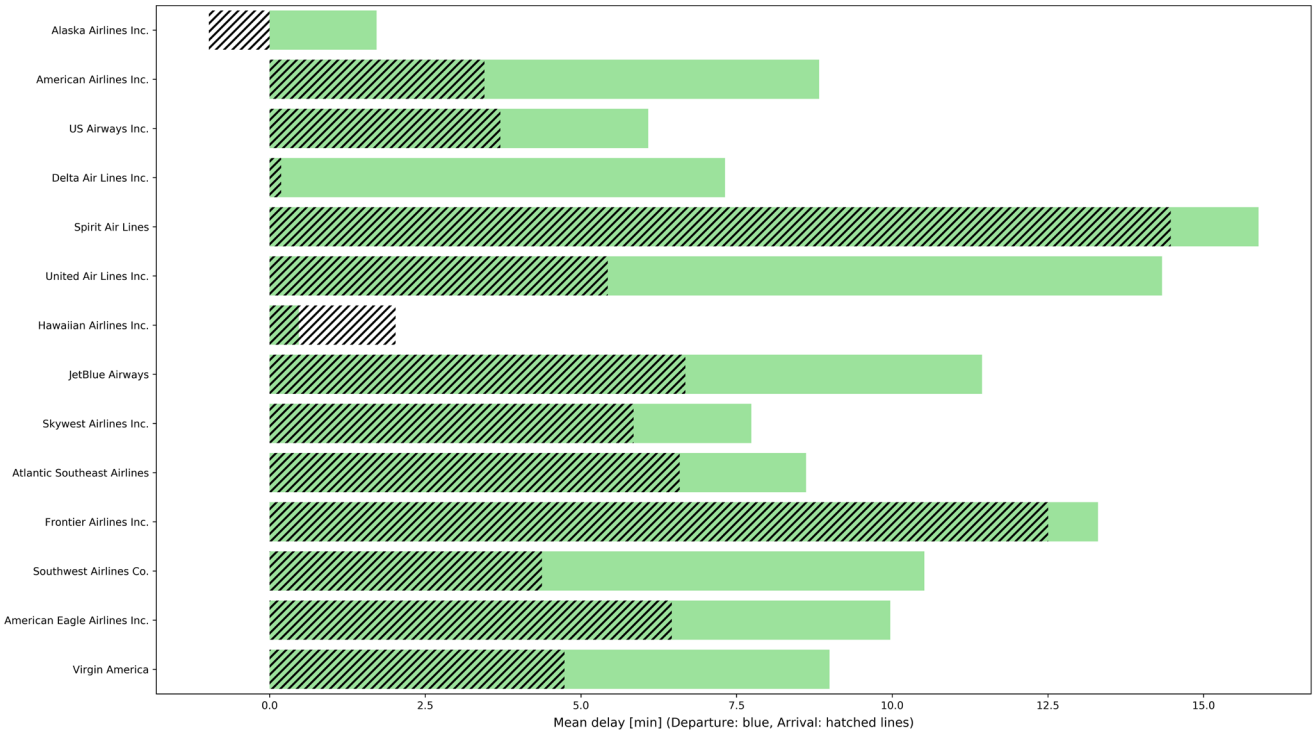
Now for the final visualization, we compare each airline with their mean delays to understand which airline has the highest probability of getting their flights delayed.



Now looking making this comparison even more understandable, we take in account all the flights by each airline and represent them as bars if they're on time (t<5 minutes), the delay is small (5 < t < 45 minutes) or if the delay is huge (t > 45 minutes).



Now for our final representation we understand the relation between departure delay and arrival delay, as it tells us if an airline increases their travel speed to avoid arrival delay or not, which helps us to understand if it gets delayed or not.



UI and Integration

We simulated a simple console-based UI to help understand how the actual API integration would take the data and predict the delay for this we take **MONTH, DAY, DAY OF WEEK, AIRLINE, FLIGHT NUMBER, ORIGIN_AIRPORT, DESTINATION_AIRPORT, SCHEDULED_DEPARTURE_HR, SCHEDULED_DEPARTURE_MN, DEPARTURE_TIME_HR, DEPARTURE_TIME_MN** and **DEPARTURE_DELAY** from the user, but the actual API integration would need only **FLIGHT_NUMBER** as the input as all the other values can be taken from the site database automatically to predict the delays.

Output:

```
----------FLIGHTS DELAY PREDICTION----------
Enter month please:  1
Enter day please:  1
Enter week day please:  4
Enter flight number please:  98
Enter airline name please:  AS
Enter origin airport please:  ATL
Enter destination airport please:  SEA
Enter scheduled departure time:  00:05
Enter departure time:  23:54
Enter delay:  -1
[[ 1.00000000e+00  1.00000000e+00  4.00000000e+00  5.22563232e+18
   9.80000000e+01 -6.75201843e+18  3.28520185e+18  0.00000000e+00
   5.00000000e+00  2.30000000e+01  5.40000000e+01 -1.10000000e+01]]
Not delayed
```