DEVELOPING A FLIGHT DELAY PREDICTION MODEL USING MACHINE LEARNING

A Project report submitted in partial fulfilment of 7th semester in degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

Submitted By

TEAM ID-PNT2022TMID06101

PRADEEP A (19CS026)

DURAI SETHUPATHY M(19CS012)

HEMA T(19CS017)

ROHITH M(19CS029)

CONTENTS 1. **INTRODUCTION**

CHAPTER 1

INTRODUCTION

Travelers have begun to favor air travel more and more over the past 20 years, primarily due to its quickness and occasional comfort. Both on the ground and in the air, as a result, have experienced amazing growth. Massive amounts of ground and airborne aircraft delays have also been brought on by an increase in air traffic. Large economic and environmental losses are the result of these delays. The model's primary goal is to correctly forecast flight delays in order to improve aircraft operations and reduce delays.

1.1. PROJECT OVERVIEW

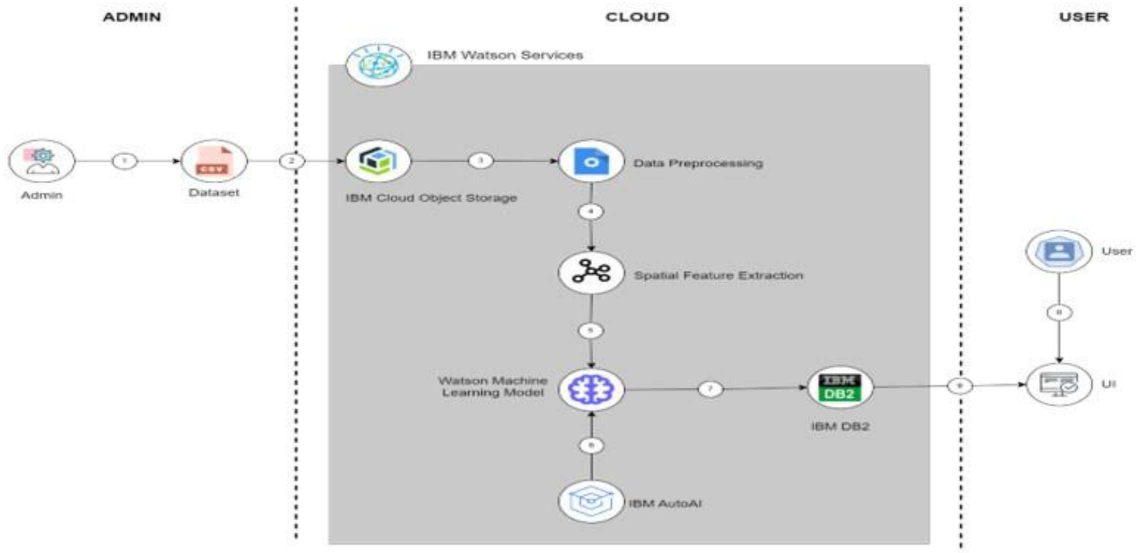


Figure 1.1. Technical Architecture

Flight arrival delays can be predicted using a machine learning algorithm. Rows of feature vectors, such as departure date, delay, travel time between the two airports, and scheduled arrival time, provide the input to our algorithm. The decision tree classifier is then used to determine whether or not the flight arrival will be delayed. When there is more than a 15-minute gap between the scheduled and actual arrival timings, a flight is deemed to be delayed. For various figures of merit, we contrast the decision tree classifier with logistic regression and a straightforward neural network.

1.2. PURPOSE

The main goal of this project is to predict the flight delay using machine learning algorithms. Flight planning is one of the difficulties in the industrial environment because there are many unpredictabilities. One such condition is the incidence of delays, which can result from a variety of causes and impose significant expenses on airlines, operators, and passengers. Delays in departure can be brought on by inclement weather, seasonal and holiday demands, airline policies, technical issues with airport infrastructure, baggage handling, and mechanical equipment, and a buildup of delays from earlier flights. Hence Predicting flight delays can improve airline operations and passenger satisfaction, which will result in a positive impact on the economy.

CHAPTER 2

LITERATURE SURVEY

1. Flight Delay Prediction System - Yogita Borse , Dhruvin Jain , Shreyash Sharma , Viral Vora, Aakash Zaveri (2020)

Statistical analysis

Statistical model requires the use of correlation analysis, parametric and non parametric tests, multivariate analysis and econometric models. Government agencies have invested in these econometric models to understand the relationship between delay and Passenger demand, fare, size of aircraft etc

Probabilistic models

Probabilistic model requires analysis tools that estimates the probability of an event based on the historic data. The estimated outcome is given in form of a distribution function of the probability. The factor of randomness always makes an impact on the decision or the outcome produced by the probabilistic model.

2.A deep learning approach to flight delay prediction - Young Jin Kim; Sun Choi; Simon Briceno; Dimitri Mavris(2016)

Deep learning has achieved significant improvement in various machine learning tasks including image recognition, speech recognition, machine translation and etc. Inspired by the huge success of the paradigm, there have been lots of tries to apply deep learning algorithms to data analytics problems with big data including traffic flow prediction. However, there has been no attempt to apply the deep learning algorithms to the analysis of air traffic data. This paper investigates the effectiveness of the deep learning models in the air traffic delay prediction tasks. By combining multiple models based on the deep learning paradigm, an accurate and robust prediction model has been built which enables an elaborate analysis of the patterns in air traffic delays. In particular, Recurrent Neural Networks (RNN) has shown its great accuracy in modeling sequential data. Day-to- day sequences of the departure and arrival flight delays of an individual airport have been modeled by the Long ShortTerm Memory RNN architecture. It has been shown that the accuracy of RNN improves with deeper architectures. In this study, four different ways of building deep RNN architecture are also discussed. Finally, the accuracy of the proposed prediction model was measured, analyzed and compared with previous prediction methods. It shows best accuracy compared with all other methods.

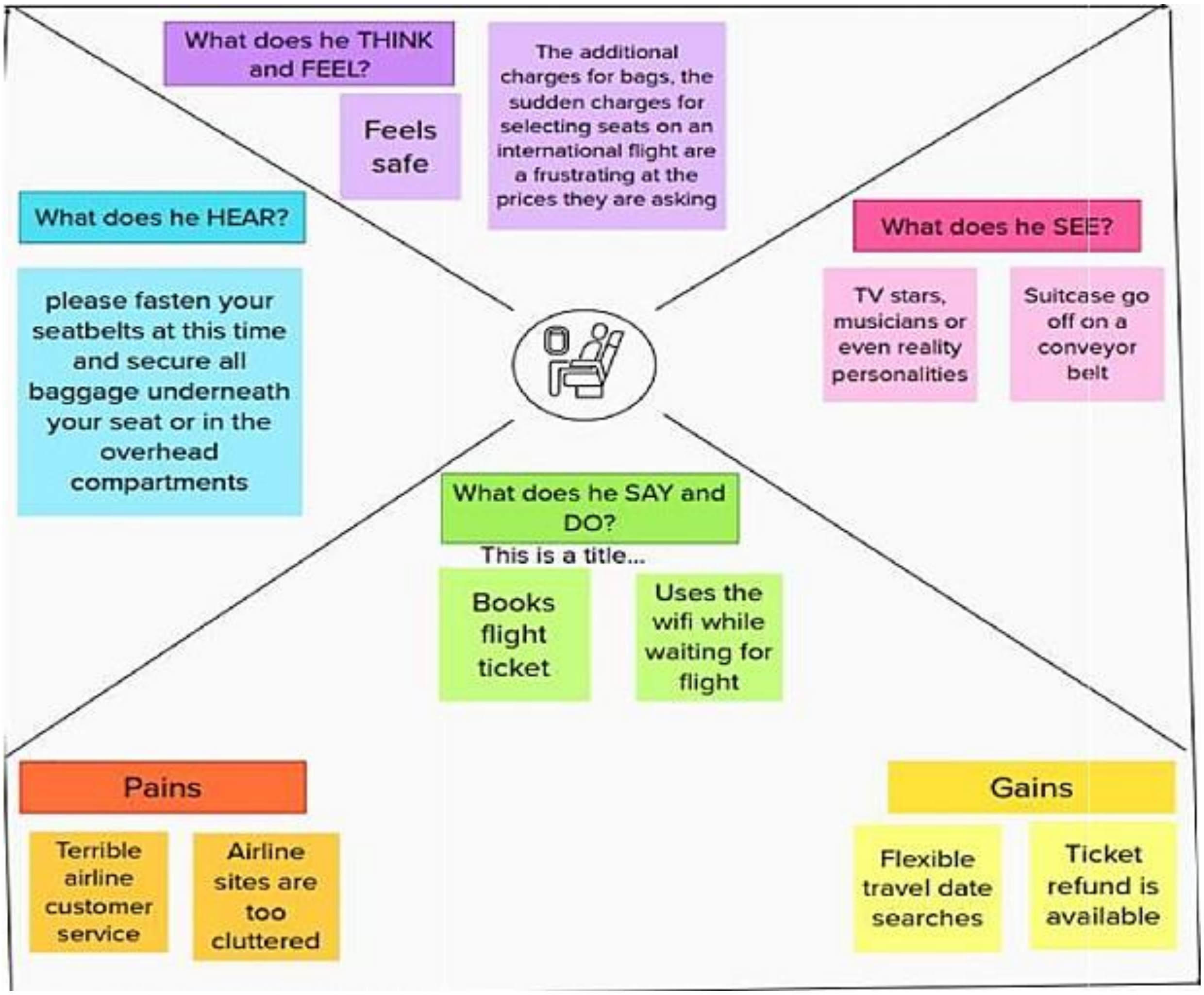
3.Research on Flight Delay Prediction Based on Random Forest - Peng Hu; Jianping Zhang; Ning Li(2021)

Based on the random forest model, this paper proposes a flight delay prediction model. By analyzing the departure flight data of Guangzhou Baiyun

International Airport in June 2020, and selecting the data of ten landing airports, it analyzes the distribution of delayed, punctual, and early arrived. It studies the selection of features that impact on flight delays, and establishes random forest predictions model. Through case study, it researches the mean square error of different leaf sizes when the forest scale is 50 trees. The results show that the optimal leaf size is 5, and the minimum mean square error is 0.1096. And it analyzes the importance of features such as departure flight delay time, scheduled flight time, number of scheduled departure flights on the day, date, and landing airport. The research results also found that, when the forest size is 100 trees and the leaf size is 5, the out-of-bag mean square error is 0.1090, and the accuracy of the prediction model is high, which is close to 90%.

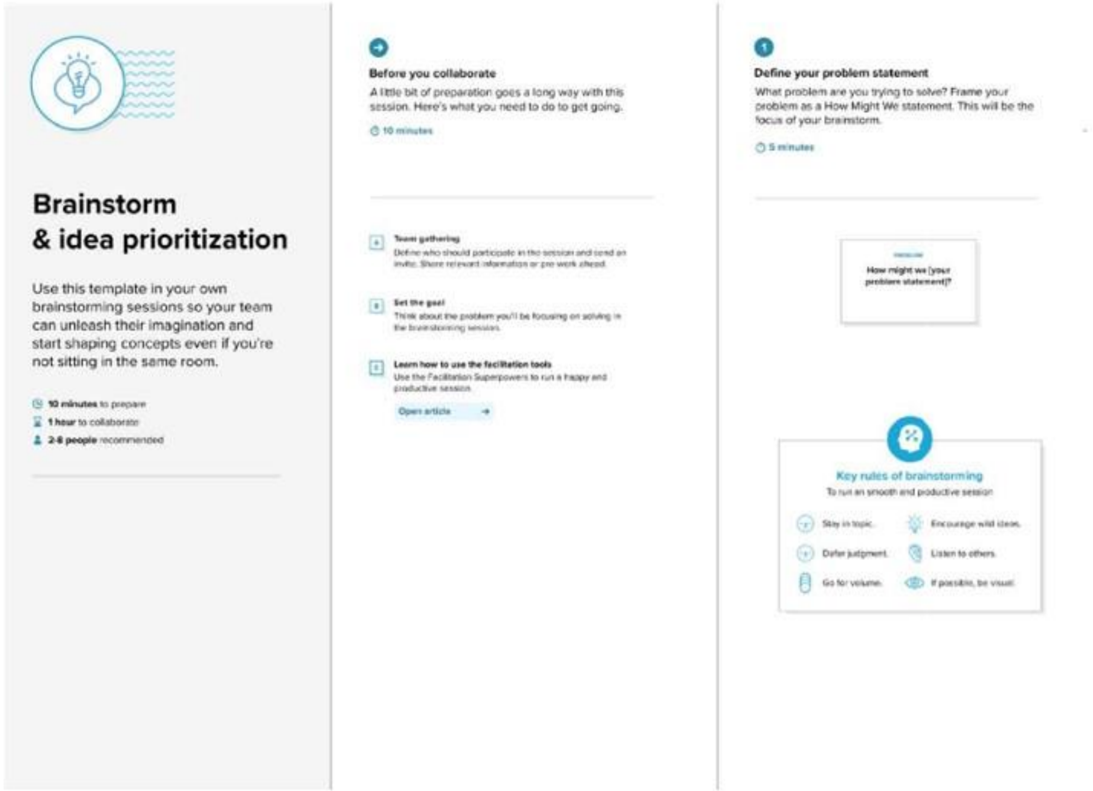
IDEATION & PROPOSED SOLUTION

3.1.EMPATHY MAP CANVAS


What does he THINK and FEEL?
Feels safe
The additional charges for bags, the sudden charges for selecting seats on an international flight are a frustrating at the prices they are asking
What does he HEAR?
please fasten your seatbelts at this time and secure all baggage underneath your seat or in the overhead compartments
What does he SEE?
TV stars, musicians or even reality personalities
Suitcase go off on a conveyor belt
What does he SAY and DO?
This is a title...
Books flight ticket
Uses the wifi while waiting for flight
Pains
Terrible airline customer service
Airline sites are too cluttered
Gains
Flexible travel date searches
Ticket refund is available

3.2. IDEATION & BRAINSTORMING

Step 1 - Team Gathering, Collaboration and Selecting the Problem Statement



Brainstorm & idea prioritization

Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.

10 minutes to prepare
1 hour to collaborate
2-8 people recommended

Before you collaborate

A little bit of preparation goes a long way with this session. Here's what you need to do to get going.

10 minutes

Team gathering
Define who should participate in the session and send an invite. Share relevant information or pre work ahead.

Set the goal
Think about the problem you'll be focusing on solving in the brainstorming session.

Learn how to use the facilitation tools
Use the Facilitation Superpowers to run a happy and productive session.

1

Define your problem statement

What problem are you trying to solve? Frame your problem as a How Might We statement. This will be the focus of your brainstorm.

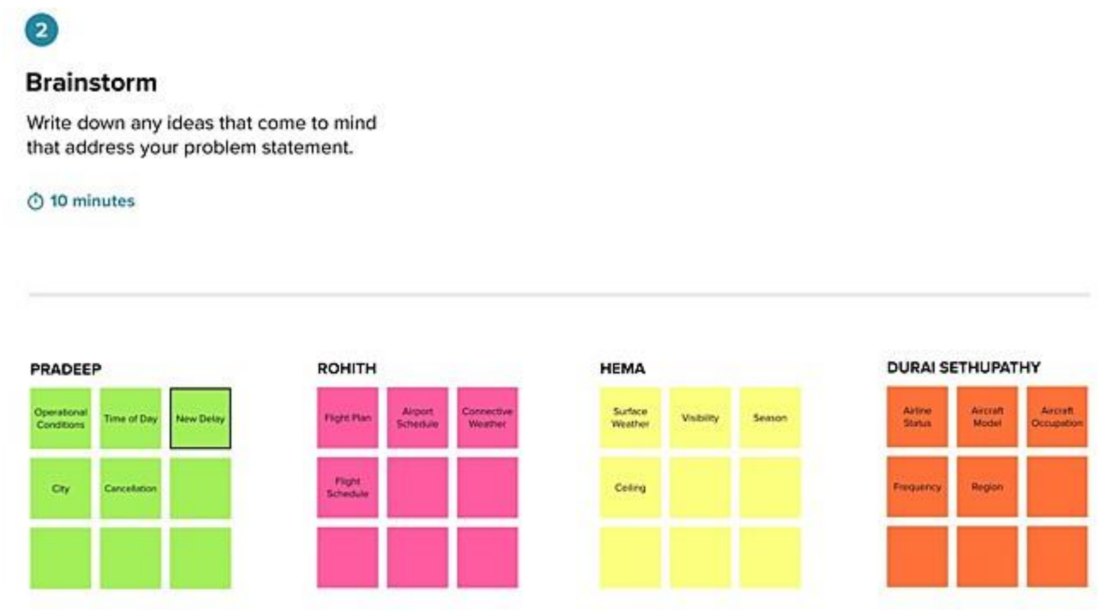
5 minutes

How might we [your problem statement]?

Key rules of brainstorming
To run an smooth and productive session

- Stay in topic.
- Encourage wild ideas.
- Defer judgment.
- Listen to others.
- Go for volume.
- If possible, be visual.

Step-2: Brainstorm, Idea Listing and Grouping



2

Brainstorm

Write down any ideas that come to mind that address your problem statement.

10 minutes

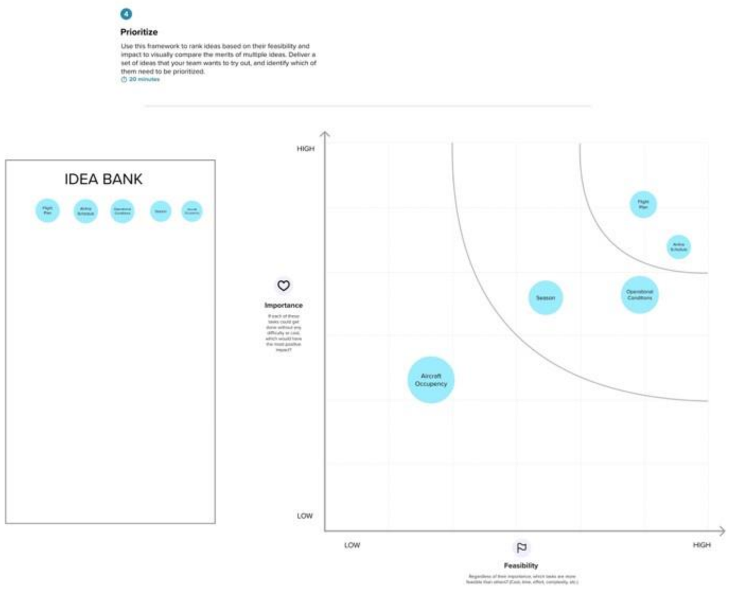
PRADEEP
- Operational Conditions
- Time of Day
- New Delay
- City
- Cancellation

ROHITH
- Flight Plan
- Airport Schedule
- Connective Weather
- Flight Schedule

HEMA
- Surface Weather
- Visibility
- Season
- Ceiling

DURAI SETHUPATHY
- Airline Status
- Aircraft Model
- Aircraft Occupation
- Frequency
- Region

Step 3 - Idea Prioritization



4

Prioritize

Use this framework to rank ideas based on their feasibility and impact to visually compare the merits of multiple ideas. Deliver a set of ideas that your team wants to try out, and identify which of them need to be prioritized.

20 minutes

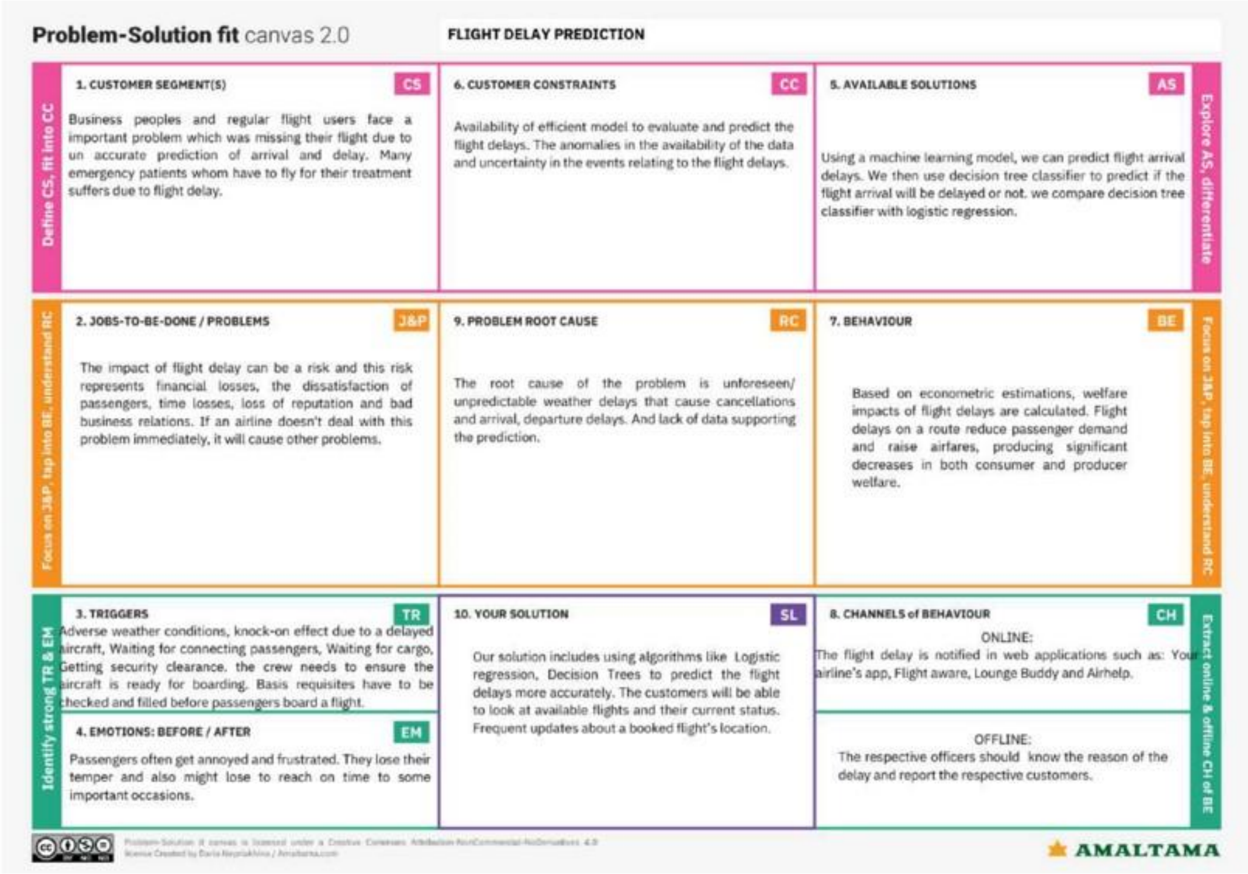
IDEA BANK
HIGH
LOW
Importance
If each of these tasks could get done without any difficulty or cost, which would have the most positive impact?
LOW
HIGH
Feasibility
Regardless of their importance, which tasks are more feasible than others? (Cost, time, effort, complexity, etc.)
Flight Plan
Airline Schedule
Season
Operational Conditions
Aircraft Occupency

3.3. PROPOSED SOLUTION

S.No.	Parameter	Description

1.	Problem Statement (Problem to be solved)	Flight delays have been the most challenging area for airlines to improve. They have been affecting the air industry directly and indirectly causing unforeseen expenses thereby reducing the reputation of the industry and the airlines. Thus, knowing if a flight would be delayed beforehand can let passengers and airlines be prepared for the circumstances. This solution aims at making it possible by predicting arrival and departure delays using Machine learning.
2.	Idea / Solution description	Using a machine learning model, we can predict flight arrival delays. The input to our algorithm is rows of feature vector like departure date, departure delay, distance between the two airports, scheduled arrival time etc.
3.	Novelty / Uniqueness	The solution takes into account all possible reasons for delay(crew delys, weather, air traffic, aircraft type) to provide an accurate prediction. Apart from predicting arrival delays, departure delays are also predicted in order for the passengers to prepare accordingly and for the airline to make arrangements suitably.
4.	Social Impact / Customer Satisfaction	By predicting the flight delay in advance the passengers can plan accordingly.
5.	Business Model (Revenue Model)	Knowing the probability of flight delay or cancellation is a crucial tool for travellers, so we set about creating a model to predict longterm flight delays. Rather than looking at disruptions caused by punctual factors like weather, we wanted to see which flights and itineraries had the highest probability of delays or cancellations over time

3.4. PROBLEM SOLUTION FIT



Problem-Solution fit canvas 2.0

FLIGHT DELAY PREDICTION

Define CS, fit into CC

1. CUSTOMER SEGMENT(S) CS

Business peoples and regular flight users face a important problem which was missing their flight due to un accurate prediction of arrival and delay. Many emergency patients whom have to fly for their treatment suffers due to flight delay.

6. CUSTOMER CONSTRAINTS CC

Availability of efficient model to evaluate and predict the flight delays. The anomalies in the availability of the data and uncertainty in the events relating to the flight delays.

5. AVAILABLE SOLUTIONS AS

Using a machine learning model, we can predict flight arrival delays. We then use decision tree classifier to predict if the flight arrival will be delayed or not. we compare decision tree classifier with logistic regression.

Explore AS, differentiate

Focus on J&P, tap into BE, understand RC

2. JOBS-TO-BE-DONE / PROBLEMS J&P

The impact of flight delay can be a risk and this risk represents financial losses, the dissatisfaction of passengers, time losses, loss of reputation and bad business relations. If an airline doesn't deal with this problem immediately, it will cause other problems.

9. PROBLEM ROOT CAUSE RC

The root cause of the problem is unforeseen/ unpredictable weather delays that cause cancellations and arrival, departure delays. And lack of data supporting the prediction.

7. BEHAVIOUR BE

Based on econometric estimations, welfare impacts of flight delays are calculated. Flight delays on a route reduce passenger demand and raise airfares, producing significant decreases in both consumer and producer welfare.

Focus on J&P, tap into BE, understand RC

Identify strong TR & EM

3. TRIGGERS TR

Adverse weather conditions, knock-on effect due to a delayed aircraft, Waiting for connecting passengers, Waiting for cargo, Getting security clearance. the crew needs to ensure the aircraft is ready for boarding. Basis requisites have to be checked and filled before passengers board a flight.

4. EMOTIONS: BEFORE / AFTER EM

Passengers often get annoyed and frustrated. They lose their temper and also might lose to reach on time to some important occasions.

10. YOUR SOLUTION SL

Our solution includes using algorithms like Logistic regression, Decision Trees to predict the flight delays more accurately. The customers will be able to look at available flights and their current status. Frequent updates about a booked flight's location.

8. CHANNELS of BEHAVIOUR CH

ONLINE:

The flight delay is notified in web applications such as: Your airline's app, Flight aware, Lounge Buddy and Airhelp.

OFFLINE:

The respective officers should know the reason of the delay and report the respective customers.

Extract online & offline CH of BE

AMALTAMA

CHAPTER 4

REQUIREMENT ANALYSIS

4.1. FUNCTIONAL REQUIREMENT

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Gmail
FR-2	User Confirmation	Confirmation via Email
FR-3	User Login	Login using credentials
FR-4	User Verification	To check if a user is authorized or not
FR-5	Search Flights	The system should allow users to search for their flight details .
FR-6	Flights Status Notification	Passengers can view the status of their flight anytime.

4.2. NON-FUNCTIONAL REQUIREMENTS

FR No.	Non-Functional Requirement	Description
NFR-1	Usability	Ease of use Ease of access
NFR-2	Security	Information about the users and their flight details is kept private. Provides assurance to users by informing them of possible flight delay
NFR-3	Reliability	Should provide accurate predictions
NFR-4	Performance	Should provide an uninterrupted connection. High- speed performance

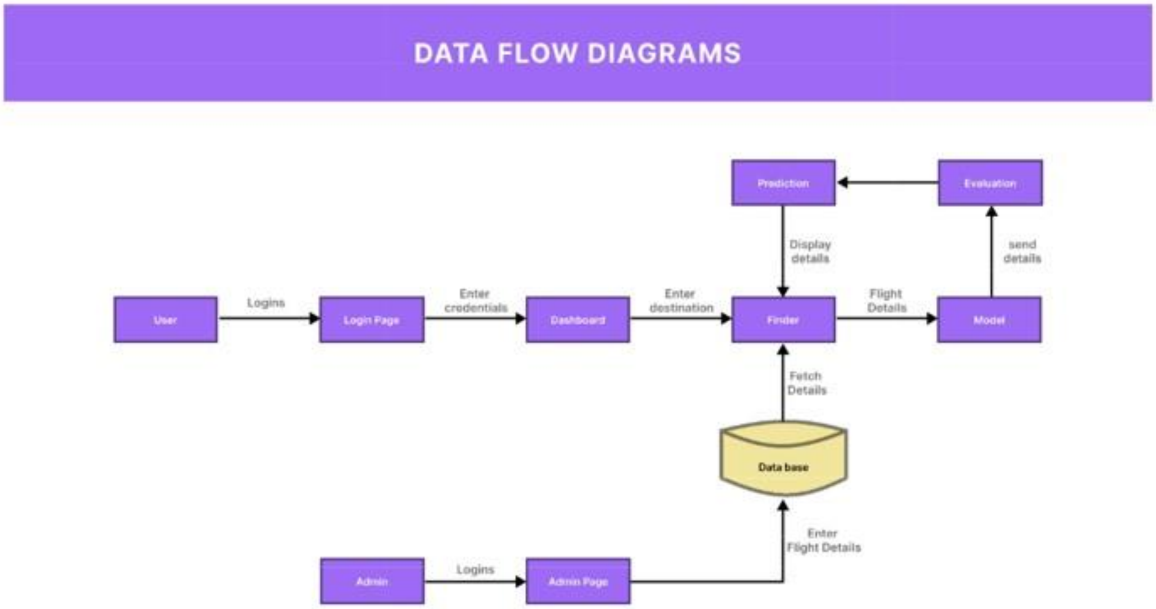
NFR-5	Availability	The system should be available at all times.
NFR-6	Scalability	Can handle multiple users at the same time Accessible even in remote areas

CHAPTER 5

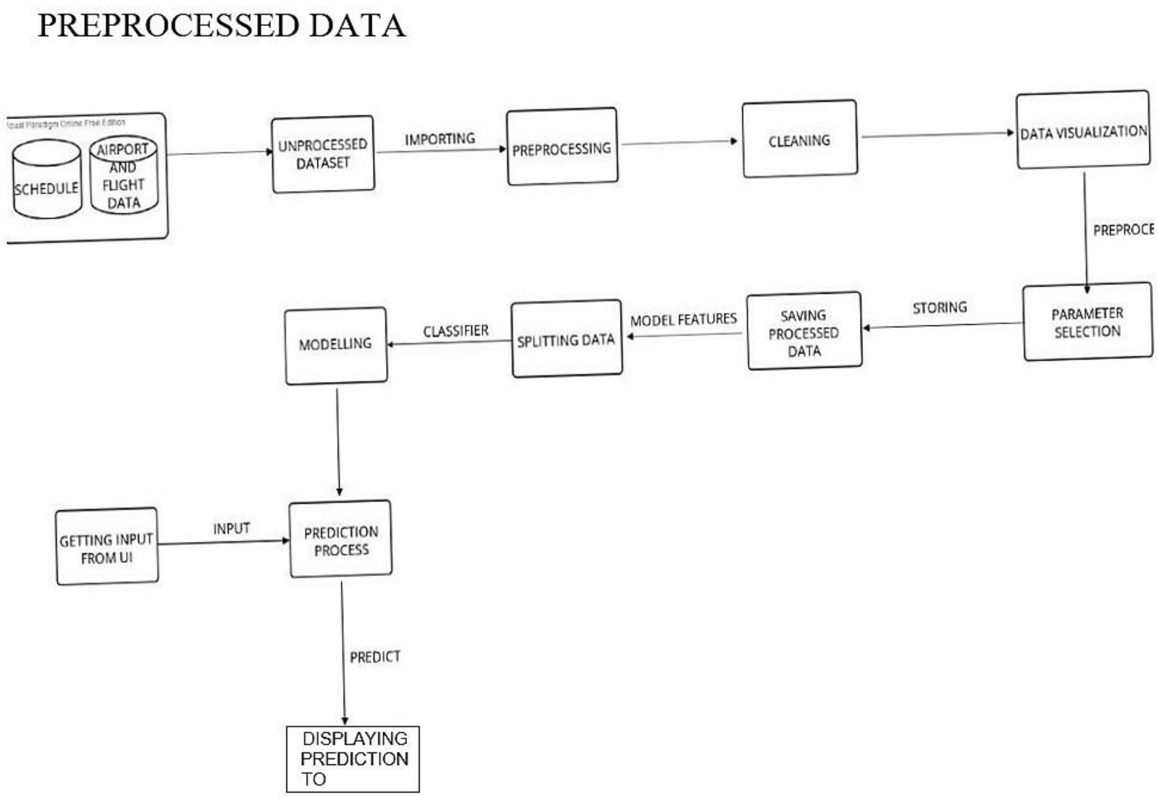
PROJECT DESIGN

5.1. DATA FLOW DIAGRAMS

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It shows how data enters and leaves the system, what changes the information, and where data is stored.



5.2. SOLUTION & TECHNICAL ARCHITECTURE



Technology Stack

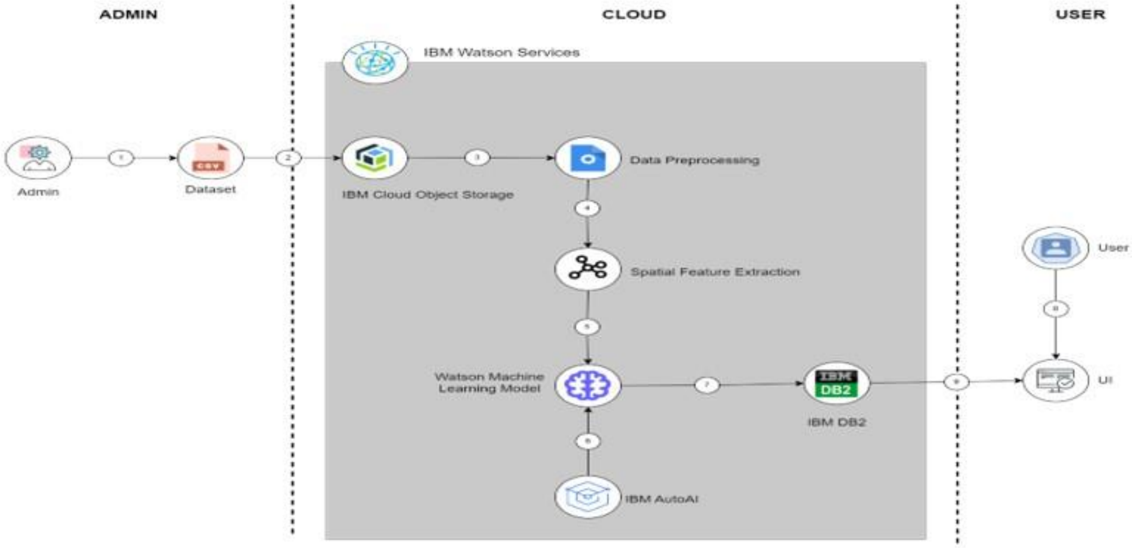


Figure 1.1. Technical Architecture

Components & Technologies

S.No	Component	Description	Technology
1.	User Interface	User can interacts with application through Web UI.	HTML , CSS , JavaScript , Bootstrap , Flask
2.	Application Logic-1	The user can enter the data in it is sent for the machine learning model for the prediction	Python
3.	Application Logic-2	The application is directly deployed in the IBM cloud	IBM Watson STT service
4.	Database	The user credentials are stored ,which is used to send notification of any updates	MySQL
5.	Cloud Database	Database Service on Cloud	IBM DB2
6.	File Storage	File storage requirements	IBM Block Storage

7.	Machine Learning Model	The model is used to predict whether the Flight Delayed or not.	Prediction Model
8.	Infrastructure (Server / Cloud)	Application Deployment on Local System / Cloud Local Server Configuration: Cloud Server Configuration :	Cloud Foundry

Application Characteristics

S.No	Characteristics	Description	Technology
1.	Open-Source Frameworks	Open-source frameworks used is IBM Watson	Technology of Opensource framework IBM Watson
2 .	Security Implementations	Authorization access scenarios and definitions, hand-over procedures for patient records	IBM Watson STT service
		between wards	
3.	Scalable Architecture	Horizontal scaling is provided by adding more machines to the pool of servers. Vertical scaling is achieved by adding more CPU and RAM to the existing machines.	IBM Watson STT service
4.	Availability	The Web interface is made available using load balancers, distributed servers etc.	IBM Watson

5.	Performance	IBM Watson –automate processes, The deep learning model is trained using IBM Watson studio for better performance, Cache, CDN’s, etc..	IBM Watson

5.3. User Stories

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority
Sprint-1	Registration and Login	USN-1	As a new user, I can register for the application by entering my email and my password.	2	High
Sprint-2	Confirmation email	USN-2	As a user, I will receive confirmation email once I have registered for the application	2	Medium
Sprint-1	User login	USN-3	As a user, I can login into the application by entering the registered email-id and password	2	High
Sprint-2	Admin Panel	USN-4	As an admin, I can authenticate the registration and login credentials of the	2	High
			passengers		
Sprint-3	Arrival and Departure time of flights	USN-5	As a user, I can find all the details of a specific flight with its number or name	2	High
Sprint-3		USN-6	As a user, I can find exactly how long the flight will be delayed	2	High

6.1. SPRINT PLANNING & ESTIMATION

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration and Login	USN-1	As a new user, I can register for the application by entering my email and my password.	2	High	Pradeep A Durai Sethupathy A
Sprint-2	Confirmation email	USN-2	As a user, I will receive confirmation email once I have registered for the application	2	Medium	Hema T Durai Sethupathy A
Sprint-1	User login	USN-3	As a user, I can login into the application by entering the registered email-id and password	2	High	Durai Sethupathy A Pradeep A
Sprint-2	Admin Panel	USN-4	As an admin, I can authenticate the registration and login credentials of the passengers	2	High	PradeepA Rohith M
Sprint-3	Arrival and Departure time of flights	USN-5	As a user, I can find all the details of a specific flight with its number or name	2	High	Hema T Rohit hM
Sprint-3		USN-6	As a user, I can find exactly how long the flight will be delayed	2	High	Durai Sethupathy M Hema T Rohith M

6.2. SPRINT DELIVERY SCHEDULE

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	5 Days	28 October 2022	02 November 2022	20	03 November 2022
Sprint-2	20	5 Days	03 November 2022	08 November 2022	20	09 November 2022
Sprint-3	20	5 Days	09 November 2022	14 November 2022	20	14 November 2022

CHAPTER 7 CODING AND SOLUTIONING

7.1. FEATURE 1 - CORRELATION BETWEEN THE VARIABLES IN THE DATASET

This will help us to find out the correlation between the variables in the dataset which would help us to find out the columns that are unnecessary and hence to be dropped.

7.2. FEATURE 2 - ONE HOT ENCODING

The cities in both Origin and Destination are one-hot encoded using the above code.

7.3. FEATURE 3 - SAVING THE MODEL WEIGHTS FOR DEPLOYMENT

The above code will save the model weights for further deployment in IBM Cloud and also measure the performance metrics.

7.4. FEATURE 4 - FLASK INTERFACE - UI

7.5. FEATURE 5 - HTML PAGES FOR FRONTEND DESIGN

```
@import
url("https://fonts.googleapis.com/css2?family=Poppins:wght@200;300;400;500;600;700;800&di
splay=swap");


* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
}


body, input
{
  font-family: "Poppins", sans-serif;
}


.container {
  position: relative;
  width: 100%;
  background-color: #fff;
  min-height: 100vh;
  overflow: hidden;
}
.forms-container {
  position: absolute;
  width: 100%;
  height: 100%;
  top: 0;
left: 0;
}
.signin-signup {
  position: absolute;
```

```css
  top: 50%;
  transform: translate(-50%, -50%);
  left: 75%;
  width: 50%;
  transition: 1s 0.7s ease-in-out;
  display: grid;
  grid-template-columns: 1fr;
  z-index: 5;
}

form {
  display: flex;
  align-items: center;
  justify-content: center;
  flex-direction: column;
  padding: 0rem 5rem;
  transition: all 0.2s 0.7s;
  overflow: hidden;
  grid-column: 1 / 2;
  grid-row: 1 / 2;
}

form.sign-up-form {
  opacity: 1;
  z-index: 1;
}

form.sign-in-form {
  z-index: 2;
}
.title {
  font-size: 2.5rem;
  color: #444;
  margin-bottom: 10px;
}

.input-field {
  max-width: 380px;
  width: 100%;
  background-color: #f0f0f0;
  margin: 10px 0;
  height: 55px;
  border-radius: 55px;
  display: grid;
  grid-template-columns: 15% 85%;
  padding: 0 0.4rem;
  position: relative;
  box-shadow: 0 2px 5px  rgba(0, 0, 0, 0.6);
}

.input-field i {
  text-align: center;
  line-height: 55px;
  color: #acacac;   transition:
0.5s;
  font-size: 1.1rem;
}

.input-field input {
  background: none;
  outline: none;
  border: none;
  line-height: 1;
  font-weight: 600;
  font-size: 1.1rem;
  color: #333;
}

.input-field input::placeholder {
  color: #aaa;
  font-weight: 500;
}
.social-text {
  padding: 0.7rem 0;
  font-size: 1rem;
}
```

```
.social-media {
  display: flex;
  justify-content: center;
}
.social-icon {
  height: 46px;
  width: 46px;
  display: flex;
  justify-content: center;
  align-items: center;
  margin: 0 0.45rem;
  color: #333;
  border-radius: 50%;
  border: 3px solid #333;
  text-decoration: none;
  font-size: 1.3rem;
  transition: 0.3s;
}
.social-icon:hover {
  color: #f7543f;
  border-color: #b83120;
}
.btn {
  width: 150px;   background-color: #ee6654;
  border: none;
  outline: none;
  height: 49px;
  border-radius: 49px;
  color: #fff;
  text-transform: uppercase;
  font-weight: 600;
  margin: 10px 0;
  cursor: pointer;
  transition: 0.5s;
  box-shadow: 0 2px 5px  rgba(0, 0, 0, 0.6);
}

.btn:hover {
```

```css
  background-color: #f14b35;
}
.panels-container {
  position: absolute;
  height: 100%;
  width: 100%;
  top: 0;
left: 0;
  display: grid;
  grid-template-columns: repeat(2, 1fr);
}
.container:before {
  content: "";
  position: absolute;
  height: 2000px;
  width: 2000px;
  top: -10%;
  right: 48%;
  transform: translateY(-50%);
  background:#FF4955;
  transition: 1.8s ease-in-out;
  border-radius: 50%;
  z-index: 6;
}

.image {
  width: 70%;
  transition: transform 1.1s ease-in-out;
  transition-delay: 0.4s;
}
.panel {
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: space-around;
  text-align: center;
  z-index: 6;
}
```

```
.left-panel {
  pointer-events: all;
  padding: 3rem 17% 2rem 12%;
}

.right-panel {
  pointer-events: none;
  padding: 3rem 12% 2rem 17%;
  align-items: flex-end;
}
.panel .content {
  color: #fff;
  transition: transform 0.9s ease-in-out;
  transition-delay: 0.6s;
}
.panel h3 {
  font-weight: 600;
  line-height: 1;
  font-size: 1.5rem;
}
.panel p {
  font-size: 0.95rem;   padding:
0.7rem 0; }
.btn.transparent {
  margin: 0;
  background: none;
  border: 3px solid #fff;
  width: 130px;
  height: 41px;
  font-weight: 600;
  box-shadow: none;
  font-size: 0.8rem;
}
.right-panel .image,
.right-panel .content {
  transform: translateX(800px);
}
```

```
/* ANIMATION */

.container.sign-up-mode:before {
  transform: translate(100%, -50%);
  right: 52%;
}
.container.sign-up-mode .left-panel .image,
.container.sign-up-mode .left-panel .content {
  transform: translateX(-800px);
}
.container.sign-up-mode .signin-signup {
  left: 25%;
}
.container.sign-up-mode form.sign-up-form {
  opacity: 1;
  z-index: 2;
}
.container.sign-up-mode form.sign-in-form {
  opacity: 0;
  z-index: 1;
}
.container.sign-up-mode .right-panel .image,
.container.sign-up-mode .right-panel .content {
  transform: translateX(0%);
}
.container.sign-up-mode .left-panel {
  pointer-events: none;
}
.container.sign-up-mode .right-panel {
  pointer-events: all;
}
@media (max-width: 870px) {
  .container {
    min-height: 800px;
    height: 100vh;
  }
  .signin-signup {
    width: 100%;
    top: 95%;
    transform: translate(-50%, -100%);
    transition: 1s 0.8s ease-in-out;
  }

  .signin-signup,
  .container.sign-up-mode .signin-signup {
    left: 50%;
  }

  .panels-container {
    grid-template-columns: 1fr;
    grid-template-rows: 1fr 2fr 1fr;
  }
  .panel {
    flex-direction: row;
    justify-content: space-around;
    align-items: center;
    padding: 2.5rem 8%;
    grid-column: 1 / 2;
  }

  .right-panel {
    grid-row: 3 / 4;
  }

  .left-panel {
    grid-row: 1 / 2;
  }
  .image {
    width: 200px;
    transition: transform 0.9s ease-in-out;
    transition-delay: 0.6s;
  }

  .panel .content {
    padding-right: 15%;
    transition: transform 0.9s ease-in-out;
    transition-delay: 0.8s;
  }
  .panel h3 {
    font-size: 1.5rem;
  }
  .panel p {
    font-size: 0.7rem;
    padding: 0.5rem 0;
  }

  .btn.transparent {
    width: 110px;
    height: 35px;
    font-size: 0.7rem;
  }

  .container:before {
    width: 1500px;
    height: 1500px;
    transform: translateX(-50%);
    left: 30%;
    bottom: 68%;
    right: initial;
    top: initial;
    transition: 2s ease-in-out;
  }
  .container.sign-up-mode:before {
    transform: translate(-50%, 100%);
    bottom: 32%;
    right: initial;
  }
  .container.sign-up-mode .left-panel .image,
  .container.sign-up-mode .left-panel .content {
    transform: translateY(-300px);
  }
  .container.sign-up-mode .right-panel .image,
  .container.sign-up-mode .right-panel .content {
    transform: translateY(0px);
  }

  .right-panel .image,
  .right-panel .content {
    transform: translateY(300px);
  }
  .container.sign-up-mode .signin-signup {
    top: 5%;
    transform: translate(-50%, 0);
  }
}
@media (max-width: 570px) {
  form {
    padding: 0 1.5rem;
  }

  .image {
    display: none;
  }
  .panel .content {
    padding: 0.5rem 1rem;
  }
  .container {
    padding: 1.5rem;
  }

  .container:before {
    bottom: 72%;
    left: 50%;
  }
  .container.sign-up-mode:before {
    bottom: 28%;
    left: 50%;
  }
                                                  }
```

TESTING

8.1. TEST

User No	Flight No	Month	Day of month	Day of week	Origin	Destination	Scheduled Departure Time	Scheduled Arrival Time	Actual Departure Time	Actual Inputs
1	1232	1	1	1	ATL	MSP	1905	2305	1945	Delayed
2	1399	1	1	1	ATL	SEA	1805	2410	1855	Delayed
3	2351	1	2	3	ATL	DTW	1305	2305	1305	Not Delayed
4	2637	2	1	3	DTW	ATL	1500	2410	1505	Not Delayed

8.2. USER ACCEPTANCE TESTING

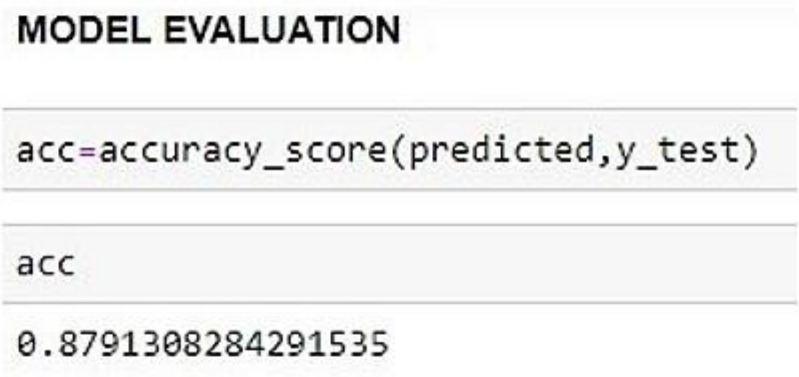
This report shows the number of test cases that have passed and failed

User No	Flight No	Month	Day Of Month	Day Of Week	Origin	Destination	Scheduled Departure Time	Scheduled Arrival Time	Actual Departure Time	Actual Output	Predict-ed Output	Correct-ness
1	1232	1	1	1	ATL	MSP	1905	2305	1945	Delayed	Delayed	Correct
2	1399	1	1	1	ATL	SEA	1805	2410	1855	Delayed	Delayed	Correct
3	2351	1	2	3	ATL	DTW	1305	2305	1305	Not Delayed	Not Delayed	Correct
4	2637	2	1	3	DTW	ATL	1500	2410	1505	Not Delayed	Not Delayed	Correct

RESULTS

9.1. PERFORMANCE METRICS

Training Accuracy



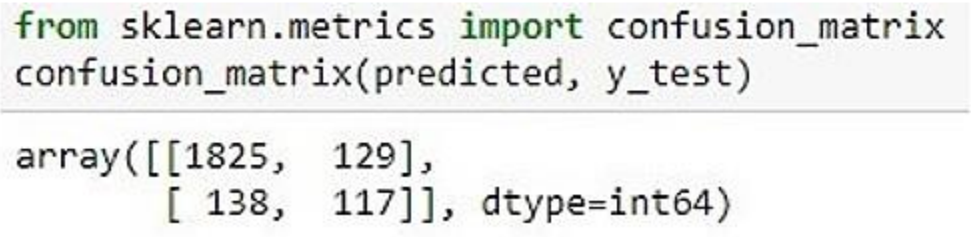
MODEL EVALUATION

```
acc=accuracy_score(predicted,y_test)
```

```
acc
```

```
0.8791308284291535
```

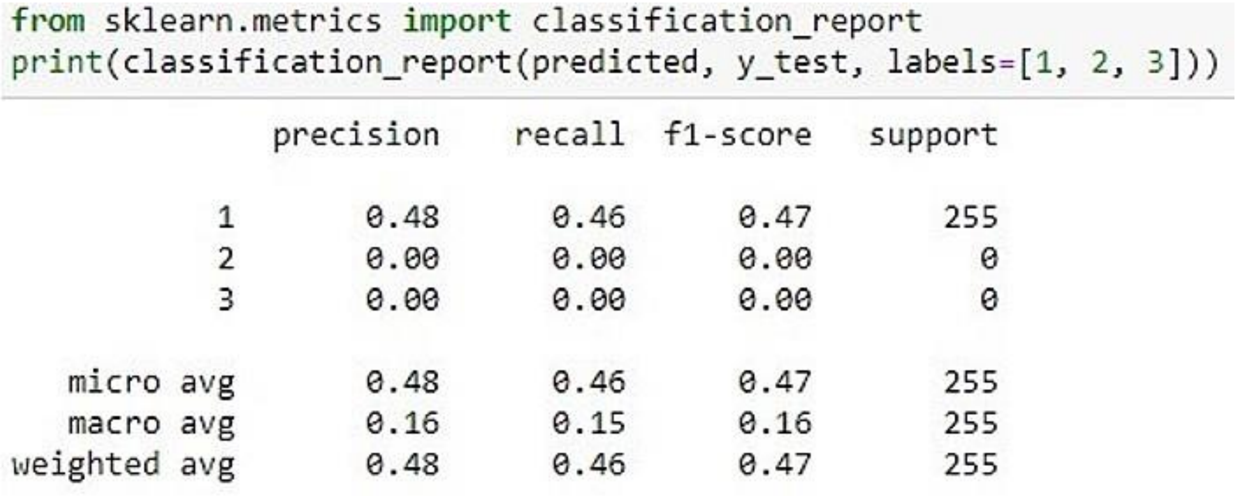
Confusion Matrix



```
from sklearn.metrics import confusion_matrix
confusion_matrix(predicted, y_test)
```

```
array([[1825,  129],
       [ 138,  117]], dtype=int64)
```

Classification Model



```
from sklearn.metrics import classification_report
print(classification_report(predicted, y_test, labels=[1, 2, 3]))
```

```
              precision    recall  f1-score   support

           1       0.48      0.46      0.47       255
           2       0.00      0.00      0.00         0
           3       0.00      0.00      0.00         0

   micro avg       0.48      0.46      0.47       255
   macro avg       0.16      0.15      0.16       255
weighted avg       0.48      0.46      0.47       255
```

ADVANTAGES AND DISADVANTAGES

Advantages

1. Customers are happy
2. The available flights are easily identified
3. Prior information will be sent if in case the flight is delayed
4. The current status of the flight can be tracked

Disadvantages

1. Wrong prediction due to noise of input data
2. If the prediction is wrong, then there will be extra expenses for the agencies, passengers and airport
3. Passengers with medical emergencies gets affected

CHAPTER 11

CONCLUSION

In this project, we use flight data, weather, and demand data to predict flight departure delay. In the end, our model correctly predicts the delayed and non-delayed flights correctly. As a result, there can be additional features related to the causes of flight delay that are not yet discovered using our existing data sources.

CHAPTER 12

FUTURE SCOPE

Based on data analysis from the year 2008, this project. There is a sizable dataset accessible from 1987 to 2008, but managing a larger dataset necessitates extensive preprocessing and purification of the data Therefore, adding a larger dataset is a part of this project's future effort. Preprocessing a bigger dataset can be done in a variety of methods, such as establishing a Spark cluster on a computer or using cloud services like AWS and Azure. Now that deep learning has advanced, we can employ neural networks algorithms to analyze aviation and meteorological data. Neural networks employ a form of pattern matching.

The project's focus is primarily on flight and weather data for India, but we can also include data from other nations like China, the United States, and Russia. We can broaden the project's scope by including flight information from international flights rather than just domestic flights.

CHAPTER 13

APPENDIX SOURCE CODES

PYTHON FLASK

```
import  requests  import
flask
from flask import url_for, request, render_template
from   flask_cors   import   CORS   import
requests
```

```
# NOTE: you must manually set API_KEY below using information retrieved from your IBM Cloud
account.
API_KEY = "zCU3gbntxqL8kInfTM2Q95jPfkfkVI9Mt8sLNC8NRipq" token_response =
requests.post('https://iam.cloud.ibm.com/identity/token',
data={"apikey": API_KEY, "grant_type": 'urn:ibm:params:oauth:grant-type:apikey'})
mltoken = token_response.json()["access_token"]

header = {'Content-Type': 'application/json', 'Authorization': 'Bearer ' + mltoken}


app = flask.Flask(__name__, static_url_path='')
CORS(app)

@app.route('/', methods=['GET']) def
sendhomePage():
  print("home")
  return render_template('home.html')


@app.route('/signupPage', methods=['GET']) def
signupPage():
  print("signup")
  return render_template('signup.html')


@app.route('/index', methods=['GET', 'POST']) def
sendindexPage():
  print("index")
  return render_template('index.html')


@app.route('/signupfn',methods = ['POST', 'GET']) def
signupfn():
  if request.method == 'POST':
    try:
      emailid = request.form['emailid']
      passwrd = request.form['password']
      usrname = request.form['username']
      with sql.connect("flightdelay.db") as con:
        cur1 = con.cursor()
        cur1.execute("select * from user_login where email=?",(emailid))
        check=cur1.rowcount
      if(check!=0):
        error1="User with this Email ID already Exists !!"
      else:
        cur = con.cursor()
        cur.execute("INSERT INTO user_login (email,password,name) VALUES
(?,?,?)",(emailid,passwrd,usrname) )
        con.commit()
        error1="User Sign Up Successfull ! Proceed Login"
        flash("Record successfully added!")
    except:
      con.rollback()

    finally:
      return render_template("Signup.html",error=error1)
      con.close()

@app.route('/loginfn',methods = ['POST', 'GET']) def
loginfn():
   emailid = request.form["emailid"]
   passwrd = request.form["password"]
   with sql.connect("flightdelay.db") as con:
      try:
        cur = con.cursor()
        cur.execute("select * from user_login where email=? and password=? limit
1",(emailid,passwrd))
        records=cur.fetchall
        session['email']=emailid
        if not records:
          record1="No Such Users Found"
        else:
          record1=records
      except:
        msg = "Incorrect Password / No Such Users Found"
      finally:
        return render_template("index.html",msg=record1)

@app.route('/category')    def
prefn():
  emailid = session['email']
  preferences = request.form["preferences"]
  with sql.connect("flightdelay.db") as con:
    try:
      cur = con.cursor()
      cur.execute("select * from user_data where email=?",(emailid))
      record=cur.fetchall
      if not record:
        cur1 = con.cursor()
        cur1.execute("INSERT INTO user_data (email,choices) VALUES (?,?,?)",(emailid,preferences))
        con.commit()
      else:
        cur2 = con.cursor()
        cur2.execute("UPDATE user_data SET choices=? where email=?",(preferences,emailid))
        con.commit()
    except:
      return render_template("test.html",msg="Somthing Went Wrong")
    finally:
      return render_template("test.html",email=emailid,preferences=preferences)

@app.route('/predict', methods=['GET','POST'])
def predict():   print("predict")
 name = request.form['name']
 month = request.form['month']
 dayofmonth = request.form['dayofmonth']
 dayofweek = request.form['dayofweek']
 origin = request.form['origin']
 if (origin== "MSP"):
     origin1, origin2,origin3, origin4, origin5 = 0,0,0,0,1
 if (origin == "DTW"):
     origin1, origin2,origin3, origin4, origin5 = 1,0,0,0,0
 if (origin == "JFK"):
     origin1, origin2,origin3, origin4, origin5 = 0,0,1,0,0
 if (origin == "SEA"):
     origin1, origin2,origin3, origin4, origin5 = 0,1,0,0,0
 if (origin == "ATL"):
     origin1, origin2, origin3, origin4, origin5 = 0,0,0,1,0
 destination = request.form['destination']
 if (destination == "MSP"):
     destination1, destination2, destination3, destination4, destination5 = 0,0,0,0,1
 if(destination == 'DTW'):
     destination1,  destination2,  destination3,  destination4,  destination5  =  1,0,0,0,0      if
(destination == "JFk") :
     destination1,  destination2,  destination3,  destination4,  destination5  =  0,0,1,0,0       if
(destination == "SEA") :
     destination1,  destination2,  destination3,  destination4,  destination5  =0,1,0,0,0       if
(destination == "ATL") :
     destination1, destination2, destination3, destination4, destination5 = 0,0,0,1,0
 dept = request.form['dept']
 arrtime = request.form['arrtime']
 actdept = request.form['actdept']
 #dept15=int(dept)- int(actdept)
 total  =  [[name,  month,  dayofmonth,  dayofweek,arrtime,actdept,origin1,  origin2,  origin3,  origin4,
origin5, destination1, destination2, destination3, destination4, destination5 ]]
 payload_scoring = {"input_data": [{"field": [[name, month, dayofmonth,
dayofweek,arrtime,actdept,origin1,origin2, origin3, origin4, origin5, destination1, destination2,
 destination3, destination4, destination5 ]], "values": total}]}
 response_scoring = requests.post('https://eude.ml.cloud.ibm.com/ml/v4/deployments/abf3959e-
b7bd-4fde-9f341295348fea93/predictions?version=2022-11-18', json=payload_scoring,
headers={'Authorization': 'Bearer ' + mltoken})
 print(response_scoring)
 predictions = response_scoring.json()
 y_pred= predictions['predictions'][0]['values'][0][0]
 print("Final prediction :",predict)
 if(y_pred==[0.0]):
    ans= "The Flight will be on time"
 else:
    ans= "The Flight will be delayed"
 return render_template("predict.html", showcase = ans)


   # showing the prediction results in a UI# showing the prediction results in a UI
```

```
if __name__ == '__main__' :
    app.run(debug= True)
```

HTML SIGNUP PAGE

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">     <script
    src="https://kit.fontawesome.com/64d58efce2.js"
    crossorigin="anonymous"
  ></script>
    <link rel="stylesheet" type="text/css"
href="{{url_for('static',filename='styles/homestyle.css')}}" />
    <title>signup</title>
</head>
<body>
    <form action="{{url_for('sendhomePage')}}" class="sign-up-form">
        <h2 class="title">Sign up</h2>
        <div class="input-field">
          <i class="fas fa-user"></i>
          <input type="text" placeholder="Username" />
        </div>
        <div class="input-field">
          <i class="fas fa-envelope"></i>
          <input type="email" placeholder="Email" />
        </div>
        <div class="input-field">
          <i class="fas fa-lock"></i>
          <input type="password" placeholder="Password" />
        </div>
        <input type="submit" class="btn" value="Sign up" />
        <p class="social-text">Or Sign up with social platforms</p>
        <div class="social-media">
```

```
     <a href="https://www.facebook.com/login/" class="social-icon">
      <i class="fab fa-facebook-f"></i>
     </a>
     <a href="https://twitter.com/login" class="social-icon">
      <i class="fab fa-twitter"></i>
     </a>

     <a href="https://www.linkedin.com/login" class="social-icon">
      <i class="fab fa-linkedin-in"></i>
     </a>
    </div>
   </form>
</body>
</html>
```

HTML INDEX PAGE

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">     <title>FLIGHT DELAY
PREDICTION</title>

</head>
<body>
 <style>
   body{
    background-image:
url('http://tripplanners.co.in/blogs/wpcontent/uploads/2014/12/
flight.jpg');     background-size: 100% 150%;
    background-repeat: no-repeat;
  }
  h1{
    color:white;
  }
 label{
    color:white;
  }



    </style>

   <center><h1>FLIGHT DELAY PREDICTION</h1></center>
   <style>
    div {
      margin-bottom: 20px;
    }
    label {
      display: inline-block;
      width: 200px;
    }
  </style>

<form action="{{url_for('predict')}}" method="POST">

  <label for="name">Enter Flight name:</label>
    <input type="number" id="name" name="name" required>     <br><br>

  <label for="month">Month:</label>
  <input type="number" name="month" required>
 <br><br>

 <label for="dayofmonth">Day of Month:</label>
 <input type="number"  name="dayofmonth" required>
<br><br>

 <label for="dayofweek">Day of Week:</label>
 <input type="number"  name="dayofweek" required>
<br><br>



  <label for="origin">origin:</label>
<select name="origin" id="og" required>
  <option value="SEA">SEA</option>
  <option value="MSP">MSP</option>
  <option value="DTW">DTW</option>
  <option  value="ATL">ATL</option>      <option
value="JFK">JFK</option>
</select> <br><br>

<label >Destination:</label>
<select name="destination" id="des" required>
  <option value="SEA">SEA</option>
  <option value="MSP">MSP</option>
  <option value="DTW">DTW</option>
  <option  value="ATL">ATL</option>      <option
value="JFK">JFK</option>
</select>
<br><br>

<label >Scheduled Departure Time:</label>
<input type="number" id="sdt" name="dept" required>
<br><br>

<label >Scheduled Arrival Time:</label>
<input type="number" id="sat" name="arrtime" required>
<br><br>

<label for="acttime">Actual Departure Time:</label>
<input type="number" id="adt" name="actdept" required>
<br><br>
<style>
  .block {
    display: block;
    width: 50%;
    border: none;
    background-color: #04AA6D;
    color: white;
    padding: 14px 28px;
    font-size: 16px;
    cursor: pointer;
    text-align: center;
  }

  .block:hover {
    background-color: #ddd;
    color: black;
  }
  </style>

<button class="block">Submit</button>

</form>
<h1>{{showcase}}</h1>

</body>
</html>
```

HTML PREDICT PAGE

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>PREDICTIONS</title>
    <style>
        body{

        background-image:
url('data:image/jpeg;base64,/9j/4AAQSkZJRgABAQAAAQABAAD/2wCEAAkGBxIQEhU
         background-size: 100% 600%;
       background-repeat: no-repeat;
       }

       h1{
```

```
    color:rgb(214, 32, 32);
   }

     </style>
</head>
<body >

   <center>
   <h1>{{showcase}}</h1>
   <a color:green href ="{{url_for('sendindexPage')}}"> Go back </a>
    </center>
</body>
</html>
```

```
{
 "cells": [
  {
   "cell_type": "code",
   "execution_count": 5,
   "metadata": {},
   "outputs": [
    {
     "data": {
      "text/html": [
       "<div>\n",
       "<style scoped>\n",
       "    .dataframe tbody tr th:only-of-type {\n",
       "        vertical-align: middle;\n",
       "    }\n",
       "\n",
       "    .dataframe tbody tr th {\n",
       "        vertical-align: top;\n",
       "    }\n",
       "\n",
       "    .dataframe thead th {\n",
       "        text-align: right;\n",
       "    }\n",
"</style>\n",
"<table border=\"1\" class=\"dataframe\">\n",
"  <thead>\n",
"    <tr style=\"text-align: right;\">\n",
"      <th></th>\n",
"      <th>YEAR</th>\n",
"      <th>QUARTER</th>\n",
"      <th>MONTH</th>\n",
"      <th>DAY_OF_MONTH</th>\n",
"      <th>DAY_OF_WEEK</th>\n",
"      <th>UNIQUE_CARRIER</th>\n",
"      <th>TAIL_NUM</th>\n",
"      <th>FL_NUM</th>\n",
"      <th>ORIGIN_AIRPORT_ID</th>\n",
"      <th>ORIGIN</th>\n",
"      <th>...</th>\n",
"      <th>CRS_ARR_TIME</th>\n",
"      <th>ARR_TIME</th>\n",
"      <th>ARR_DELAY</th>\n",
"      <th>ARR_DEL15</th>\n",
"      <th>CANCELLED</th>\n",
"      <th>DIVERTED</th>\n",
"      <th>CRS_ELAPSED_TIME</th>\n",
"      <th>ACTUAL_ELAPSED_TIME</th>\n",
"      <th>DISTANCE</th>\n",
"      <th>Unnamed: 25</th>\n",
"    </tr>\n",
"  </thead>\n",
"  <tbody>\n",
"    <tr>\n",
"      <th>0</th>\n",
"      <td>2016</td>\n",
"      <td>1</td>\n",
"      <td>1</td>\n",
"      <td>1</td>\n",
"      <td>5</td>\n",
"      <td>DL</td>\n",
"      <td>N836DN</td>\n",
"      <td>1399</td>\n",
"      <td>10397</td>\n",
"      <td>ATL</td>\n",
"      <td>...</td>\n",
"      <td>2143</td>\n",
"      <td>2102.0</td>\n",
"      <td>-41.0</td>\n",
"      <td>0.0</td>\n",
"      <td>0.0</td>\n",
"      <td>0.0</td>\n",
"      <td>338.0</td>\n",
"      <td>295.0</td>\n",
"      <td>2182.0</td>\n",
"      <td>NaN</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>1</th>\n",
"      <td>2016</td>\n",
"      <td>1</td>\n",
"      <td>1</td>\n",
"      <td>1</td>\n",
"      <td>5</td>\n",
"      <td>DL</td>\n",
"      <td>N964DN</td>\n",
"      <td>1476</td>\n",
"      <td>11433</td>\n",
"     <td>DTW</td>\n",       "
<td>...</td>\n",
"      <td>1435</td>\n",
"      <td>1439.0</td>\n",
"      <td>4.0</td>\n",
"      <td>0.0</td>\n",
"      <td>0.0</td>\n",
"      <td>0.0</td>\n",
"      <td>110.0</td>\n",
"      <td>115.0</td>\n",
"      <td>528.0</td>\n",
"      <td>NaN</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>2</th>\n",
"      <td>2016</td>\n",
"      <td>1</td>\n",
"      <td>1</td>\n",
"      <td>1</td>\n",
"      <td>5</td>\n",
"      <td>DL</td>\n",
"      <td>N813DN</td>\n",
"      <td>1597</td>\n",
"      <td>10397</td>\n",
"      <td>ATL</td>\n",
"      <td>...</td>\n",
"      <td>1215</td>\n",
"      <td>1142.0</td>\n",
"      <td>-33.0</td>\n",
"      <td>0.0</td>\n",
"      <td>0.0</td>\n",
"      <td>0.0</td>\n",
"      <td>335.0</td>\n",
"      <td>300.0</td>\n",
"      <td>2182.0</td>\n",
"      <td>NaN</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>3</th>\n",
"      <td>2016</td>\n",
"      <td>1</td>\n",
"      <td>1</td>\n",
"      <td>1</td>\n",
"      <td>5</td>\n",
"      <td>DL</td>\n",
"      <td>N587NW</td>\n",
"      <td>1768</td>\n",
"      <td>14747</td>\n",
"      <td>SEA</td>\n",
"      <td>...</td>\n",
"      <td>1335</td>\n",
"      <td>1345.0</td>\n",
"      <td>10.0</td>\n",
"      <td>0.0</td>\n",
"      <td>0.0</td>\n",
"      <td>0.0</td>\n",
"      <td>196.0</td>\n",
"      <td>205.0</td>\n",
"      <td>1399.0</td>\n",
"      <td>NaN</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>4</th>\n",
"      <td>2016</td>\n",
"      <td>1</td>\n",
"      <td>1</td>\n",
"      <td>1</td>\n",
"      <td>5</td>\n",
"      <td>DL</td>\n",
"      <td>N836DN</td>\n",
"      <td>1823</td>\n",
"      <td>14747</td>\n",
"      <td>SEA</td>\n",
"      <td>...</td>\n",
"      <td>607</td>\n",
"      <td>615.0</td>\n",
"      <td>8.0</td>\n",
"      <td>0.0</td>\n",
"      <td>0.0</td>\n",
"      <td>0.0</td>\n",
"      <td>247.0</td>\n",
"      <td>259.0</td>\n",
"      <td>1927.0</td>\n",
"      <td>NaN</td>\n",
"    </tr>\n",
"  </tbody>\n",
"</table>\n",
"<p>5 rows × 26 columns</p>\n",
     "</div>"
    ],
    "text/plain": [
     "   YEAR  QUARTER  MONTH  DAY_OF_MONTH  DAY_OF_WEEK
UNIQUE_CARRIER TAIL_NUM  \\\n",
     "0  2016       1      1             1            5             DL   N836DN   \n",
     "1  2016       1      1             1            5             DL   N964DN   \n",
     "2  2016       1      1             1            5             DL   N813DN   \n",
     "3  2016       1      1             1            5             DL   N587NW   \n",      "4  2016
1      1             1            5             DL   N836DN   \n",      "\n",
     "   FL_NUM  ORIGIN_AIRPORT_ID ORIGIN  ...  CRS_ARR_TIME ARR_TIME  ARR_DELAY  \\\n",
     "0    1399             10397    ATL  ...          2143   2102.0      -41.0   \n",
     "1    1476             11433    DTW  ...          1435   1439.0        4.0   \n",
     "2    1597             10397    ATL  ...          1215   1142.0      -33.0   \n",
     "3    1768             14747    SEA  ...          1335   1345.0       10.0   \n",
     "4    1823             14747    SEA  ...           607    615.0        8.0   \n",      "\n",
     "   ARR_DEL15  CANCELLED  DIVERTED  CRS_ELAPSED_TIME
ACTUAL_ELAPSED_TIME  \\\n",
     "0        0.0        0.0       0.0             338.0               295.0   \n",
     "1        0.0        0.0       0.0             110.0               115.0   \n",
     "2        0.0        0.0       0.0             335.0               300.0   \n",
     "3        0.0        0.0       0.0             196.0               205.0   \n",      "4        0.0
0.0       0.0             247.0               259.0   \n",
     "\n",
     "   DISTANCE  Unnamed: 25  \n",
     "0    2182.0          NaN  \n",
     "1     528.0          NaN  \n",
     "2    2182.0          NaN  \n",
     "3    1399.0          NaN  \n",
     "4    1927.0          NaN  \n",
     "\n",
     "[5 rows x 26 columns]"
    ]
   },
   "execution_count": 5,
   "metadata": {},
   "output_type": "execute_result"
  }
```

```
   ],
   "source": [
    "\n",
    "import os, types\n",
    "import pandas as pd\n",
    "from botocore.client import Config\n",
    "import ibm_boto3\n",
    "\n",
    "def __iter__(self): return 0\n",
    "\n",
    "# @hidden_cell\n",
    "# The following code accesses a file in your IBM Cloud Object Storage. It includes your credentials.\n",
    "# You might want to remove those credentials before you share the notebook.\n",
    "cos_client = ibm_boto3.client(service_name='s3',\n",
    "    ibm_api_key_id='iUZj_xocQAxwGUba0IEvgqEwHXcaMCT3EkhVOVXJ60yk',\n",
    "    ibm_auth_endpoint=\"https://iam.cloud.ibm.com/oidc/token\",\n",
    "    config=Config(signature_version='oauth'),\n",
    "    endpoint_url='https://s3.private.eu.cloud-object-storage.appdomain.cloud')\n",
    "\n",
    "bucket = 'flightdelay-donotdelete-pr-k6u3ulqavon8e1'\n",
    "object_key = 'flightdata.csv'\n",
    "\n",
    "body = cos_client.get_object(Bucket=bucket,Key=object_key)['Body']\n",
    "# add missing __iter__ method, so pandas accepts body as file-like object\n",
    "if not hasattr(body, \"__iter__\"): body.__iter__ = types.MethodType( __iter__, body )\n",
    "\n",
    "data= pd.read_csv(body)\n",
    "data.head()\n"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 6,
   "metadata": {
    "id": "l0p7DMk4jV3r"
   },
   "outputs": [],
  "source": [
   "import numpy as np\n",
   "import pandas as pd\n",
   "import matplotlib.pyplot as plt\n",
   "import seaborn as sns\n",
   "import pickle\n",
   "%matplotlib inline\n",
   "from sklearn.preprocessing import LabelEncoder\n",
   "from sklearn.preprocessing import OneHotEncoder\n",
   "from sklearn.model_selection import train_test_split\n",
   "from sklearn.preprocessing import StandardScaler\n",
   "from sklearn.tree import DecisionTreeClassifier\n",
   "from sklearn.metrics import accuracy_score\n",
   "import sklearn.metrics as metrics"
  ]
 },
 {
  "cell_type": "markdown",
  "metadata": {
   "id": "YThn7gb0lXHG"
  },
  "source": [
   "**Importing the dataset**"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 7,
  "metadata": {
   "colab": {
    "base_uri": "https://localhost:8080/",
    "height": 474
   },
   "id": "zy-9QZQBldsU",
   "outputId": "738ed34c-3608-4176-9c59-3cfc3ee6b088"
  },
  "outputs": [
   {
```

```
"data": {
 "text/html": [
  "<div>\n",
  "<style scoped>\n",
  "    .dataframe tbody tr th:only-of-type {\n",
  "        vertical-align: middle;\n",
  "    }\n",
  "\n",
  "    .dataframe tbody tr th {\n",
  "        vertical-align: top;\n",
  "    }\n",
  "\n",
  "    .dataframe thead th {\n",
  "        text-align: right;\n",
  "    }\n",
  "</style>\n",
  "<table border=\"1\" class=\"dataframe\">\n",
  "  <thead>\n",
  "    <tr style=\"text-align: right;\">\n",
  "      <th></th>\n",
  "      <th>YEAR</th>\n",
  "      <th>QUARTER</th>\n",
  "      <th>MONTH</th>\n",
  "      <th>DAY_OF_MONTH</th>\n",
  "      <th>DAY_OF_WEEK</th>\n",
  "      <th>UNIQUE_CARRIER</th>\n",
  "      <th>TAIL_NUM</th>\n",
  "      <th>FL_NUM</th>\n",
  "      <th>ORIGIN_AIRPORT_ID</th>\n",
  "      <th>ORIGIN</th>\n",
  "      <th>...</th>\n",
  "      <th>CRS_ARR_TIME</th>\n",
  "      <th>ARR_TIME</th>\n",
  "      <th>ARR_DELAY</th>\n",
  "      <th>ARR_DEL15</th>\n",
  "      <th>CANCELLED</th>\n",
  "      <th>DIVERTED</th>\n",
  "      <th>CRS_ELAPSED_TIME</th>\n",
"      <th>ACTUAL_ELAPSED_TIME</th>\n",
"      <th>DISTANCE</th>\n",
"      <th>Unnamed: 25</th>\n",
"    </tr>\n",
"  </thead>\n",
"  <tbody>\n",
"    <tr>\n",
"      <th>0</th>\n",
"      <td>2016</td>\n",
"      <td>1</td>\n",
"      <td>1</td>\n",
"      <td>1</td>\n",
"      <td>5</td>\n",
"      <td>DL</td>\n",
"      <td>N836DN</td>\n",
"      <td>1399</td>\n",
"      <td>10397</td>\n",
"      <td>ATL</td>\n",
"      <td>...</td>\n",
"      <td>2143</td>\n",
"      <td>2102.0</td>\n",
"      <td>-41.0</td>\n",
"      <td>0.0</td>\n",
"      <td>0.0</td>\n",
"      <td>0.0</td>\n",
"      <td>338.0</td>\n",
"      <td>295.0</td>\n",
"      <td>2182.0</td>\n",
"      <td>NaN</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>1</th>\n",
"      <td>2016</td>\n",
"      <td>1</td>\n",
"      <td>1</td>\n",
"      <td>1</td>\n",
"      <td>5</td>\n",
"      <td>DL</td>\n",
    "     <td>N964DN</td>\n",
    "     <td>1476</td>\n",
    "     <td>11433</td>\n",
    "    <td>DTW</td>\n",      "
<td>...</td>\n",
    "     <td>1435</td>\n",
    "     <td>1439.0</td>\n",
    "     <td>4.0</td>\n",
    "     <td>0.0</td>\n",
    "     <td>0.0</td>\n",
    "     <td>0.0</td>\n",
    "     <td>110.0</td>\n",
    "     <td>115.0</td>\n",
    "     <td>528.0</td>\n",
    "     <td>NaN</td>\n",
    "   </tr>\n",
    "   <tr>\n",
    "     <th>2</th>\n",
    "     <td>2016</td>\n",
    "     <td>1</td>\n",
    "     <td>1</td>\n",
    "     <td>1</td>\n",
    "     <td>5</td>\n",
    "     <td>DL</td>\n",
    "     <td>N813DN</td>\n",
    "     <td>1597</td>\n",
    "     <td>10397</td>\n",
    "     <td>ATL</td>\n",
    "     <td>...</td>\n",
    "     <td>1215</td>\n",
    "     <td>1142.0</td>\n",
    "     <td>-33.0</td>\n",
    "     <td>0.0</td>\n",
    "     <td>0.0</td>\n",
    "     <td>0.0</td>\n",
    "     <td>335.0</td>\n",
    "     <td>300.0</td>\n",
    "     <td>2182.0</td>\n",
"      <td>NaN</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>3</th>\n",
"      <td>2016</td>\n",
"      <td>1</td>\n",
"      <td>1</td>\n",
"      <td>1</td>\n",
"      <td>5</td>\n",
"      <td>DL</td>\n",
"      <td>N587NW</td>\n",
"      <td>1768</td>\n",
"      <td>14747</td>\n",
"      <td>SEA</td>\n",
"      <td>...</td>\n",
"      <td>1335</td>\n",
"      <td>1345.0</td>\n",
"      <td>10.0</td>\n",
"      <td>0.0</td>\n",
"      <td>0.0</td>\n",
"      <td>0.0</td>\n",
"      <td>196.0</td>\n",
"      <td>205.0</td>\n",
"      <td>1399.0</td>\n",
"      <td>NaN</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>4</th>\n",
"      <td>2016</td>\n",
"      <td>1</td>\n",
"      <td>1</td>\n",
"      <td>1</td>\n",
"      <td>5</td>\n",
"      <td>DL</td>\n",
"      <td>N836DN</td>\n",
"      <td>1823</td>\n",
"      <td>14747</td>\n",
"      <td>SEA</td>\n",
"     <td>...</td>\n",
"     <td>607</td>\n",
"     <td>615.0</td>\n",
"     <td>8.0</td>\n",
"     <td>0.0</td>\n",
"     <td>0.0</td>\n",
"     <td>0.0</td>\n",
"     <td>247.0</td>\n",
"     <td>259.0</td>\n",
"     <td>1927.0</td>\n",
"     <td>NaN</td>\n",
"   </tr>\n",
"   <tr>\n",
"     <th>5</th>\n",
"     <td>2016</td>\n",
"     <td>1</td>\n",
"     <td>1</td>\n",
"     <td>1</td>\n",
"     <td>5</td>\n",
"     <td>DL</td>\n",
"     <td>N936DL</td>\n",
"     <td>1975</td>\n",
"     <td>13487</td>\n",
"    <td>MSP</td>\n",      "
<td>...</td>\n",
"     <td>1459</td>\n",
"     <td>1441.0</td>\n",
"     <td>-18.0</td>\n",
"     <td>0.0</td>\n",
"     <td>0.0</td>\n",
"     <td>0.0</td>\n",
"     <td>150.0</td>\n",
"     <td>134.0</td>\n",
"     <td>907.0</td>\n",
"     <td>NaN</td>\n",
"   </tr>\n",
"   <tr>\n",
"     <th>6</th>\n",
    "     <td>2016</td>\n",
    "     <td>1</td>\n",
    "     <td>1</td>\n",
    "     <td>2</td>\n",
    "     <td>6</td>\n",
    "     <td>DL</td>\n",
    "     <td>N983DL</td>\n",
    "     <td>2074</td>\n",
    "     <td>10397</td>\n",
    "     <td>ATL</td>\n",
    "     <td>...</td>\n",
    "     <td>1931</td>\n",
    "     <td>1920.0</td>\n",
    "     <td>-11.0</td>\n",
    "     <td>0.0</td>\n",
    "     <td>0.0</td>\n",
    "     <td>0.0</td>\n",
    "     <td>166.0</td>\n",
    "     <td>155.0</td>\n",
    "     <td>907.0</td>\n",
    "     <td>NaN</td>\n",
    "   </tr>\n",
    "   <tr>\n",
    "     <th>7</th>\n",
    "     <td>2016</td>\n",
    "     <td>1</td>\n",
    "     <td>1</td>\n",
    "     <td>2</td>\n",
    "     <td>6</td>\n",
    "     <td>DL</td>\n",
    "     <td>N589NW</td>\n",
    "     <td>2151</td>\n",
    "     <td>13487</td>\n",
    "    <td>MSP</td>\n",      "
<td>...</td>\n",
    "     <td>1929</td>\n",
    "     <td>1908.0</td>\n",
    "     <td>-21.0</td>\n",
"     <td>0.0</td>\n",
"     <td>0.0</td>\n",
"     <td>0.0</td>\n",
"     <td>229.0</td>\n",
"     <td>197.0</td>\n",
"     <td>1399.0</td>\n",
"     <td>NaN</td>\n",
"   </tr>\n",
"   <tr>\n",
"     <th>8</th>\n",
"     <td>2016</td>\n",
"     <td>1</td>\n",
"     <td>1</td>\n",
"     <td>2</td>\n",
"     <td>6</td>\n",
"     <td>DL</td>\n",
"     <td>N804DN</td>\n",
"     <td>2221</td>\n",
"     <td>13487</td>\n",
"    <td>MSP</td>\n",      "
<td>...</td>\n",
"     <td>1305</td>\n",
"     <td>1255.0</td>\n",
"     <td>-10.0</td>\n",
"     <td>0.0</td>\n",
"     <td>0.0</td>\n",
"     <td>0.0</td>\n",
"     <td>230.0</td>\n",
"     <td>220.0</td>\n",
"     <td>1399.0</td>\n",
"     <td>NaN</td>\n",
"   </tr>\n",
"   <tr>\n",
"     <th>9</th>\n",
"     <td>2016</td>\n",
"     <td>1</td>\n",
"     <td>1</td>\n",
"     <td>2</td>\n",
    "      <td>6</td>\n",
    "      <td>DL</td>\n",
    "      <td>N965DN</td>\n",
    "      <td>2291</td>\n",
    "      <td>13487</td>\n",
    "      <td>MSP</td>\n",
    "      <td>...</td>\n",
    "      <td>1801</td>\n",
    "      <td>1800.0</td>\n",
    "      <td>-1.0</td>\n",
    "      <td>0.0</td>\n",
    "      <td>0.0</td>\n",
    "      <td>0.0</td>\n",
    "      <td>151.0</td>\n",
    "      <td>137.0</td>\n",
    "      <td>907.0</td>\n",
    "      <td>NaN</td>\n",
    "    </tr>\n",
    "  </tbody>\n",
    "</table>\n",
    "<p>10 rows × 26 columns</p>\n",
    "</div>"
   ],
   "text/plain": [
    "   YEAR  QUARTER  MONTH  DAY_OF_MONTH  DAY_OF_WEEK
UNIQUE_CARRIER TAIL_NUM  \\\n",
    "0  2016        1      1             1            5             DL   N836DN   \n",
    "1  2016        1      1             1            5             DL   N964DN   \n",
    "2  2016        1      1             1            5             DL   N813DN   \n",
    "3  2016        1      1             1            5             DL   N587NW   \n",
    "4  2016        1      1             1            5             DL   N836DN   \n",
    "5  2016        1      1             1            5             DL   N936DL   \n",
    "6  2016        1      1             2            6             DL   N983DL   \n",
    "7  2016        1      1             2            6             DL   N589NW   \n",
    "8  2016        1      1             2            6             DL   N804DN   \n",
    "9  2016        1      1             2            6             DL   N965DN   \n",       "\n",
    "   FL_NUM  ORIGIN_AIRPORT_ID ORIGIN  ...  CRS_ARR_TIME ARR_TIME
ARR_DELAY  \\\n",
    "0   1399          10397   ATL  ...       2143   2102.0     -41.0  \n",
    "1   1476          11433   DTW  ...       1435   1439.0      4.0  \n",
    "2   1597          10397   ATL  ...       1215   1142.0     -33.0  \n",
    "3   1768          14747   SEA  ...       1335   1345.0     10.0  \n",
    "4   1823          14747   SEA  ...        607   615.0      8.0  \n",
    "5   1975          13487   MSP  ...       1459   1441.0     -18.0  \n",
    "6   2074          10397   ATL  ...       1931   1920.0     -11.0  \n",
    "7   2151          13487   MSP  ...       1929   1908.0     -21.0  \n",
    "8   2221          13487   MSP  ...       1305   1255.0     -10.0  \n",      "9
2291          13487   MSP  ...       1801   1800.0     -1.0  \n",      "\n",
    "   ARR_DEL15  CANCELLED  DIVERTED  CRS_ELAPSED_TIME
ACTUAL_ELAPSED_TIME  \\\n",
    "0       0.0       0.0       0.0           338.0            295.0  \n",
    "1       0.0       0.0       0.0           110.0            115.0  \n",
    "2       0.0       0.0       0.0           335.0            300.0  \n",
    "3       0.0       0.0       0.0           196.0            205.0  \n",
    "4       0.0       0.0       0.0           247.0            259.0  \n",
    "5       0.0       0.0       0.0           150.0            134.0  \n",
    "6       0.0       0.0       0.0           166.0            155.0  \n",
    "7       0.0       0.0       0.0           229.0            197.0  \n",
    "8       0.0       0.0       0.0           230.0            220.0  \n",      "9
0.0       0.0      0.0           151.0            137.0  \n",      "\n",
    "   DISTANCE  Unnamed: 25  \n",
    "0   2182.0        NaN  \n",
    "1    528.0        NaN  \n",
    "2   2182.0        NaN  \n",
    "3   1399.0        NaN  \n",
    "4   1927.0        NaN  \n",
    "5    907.0        NaN  \n",
    "6    907.0        NaN  \n",
    "7   1399.0        NaN  \n",
    "8   1399.0        NaN  \n",
    "9    907.0        NaN  \n",
    "\n",
    "[10 rows x 26 columns]"
   ]
  },
  "execution_count": 7,
     "metadata": {},
     "output_type": "execute_result"
    }
   ],
   "source": [
    "data.head(10)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 8,
   "metadata": {
    "colab": {
     "base_uri": "https://localhost:8080/"
    },
    "id": "bKyoX91-lezL",
    "outputId": "a45706ec-ea8a-4420-e891-d66da4c35f71"
   },
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "<class 'pandas.core.frame.DataFrame'>\n",
      "RangeIndex: 11231 entries, 0 to 11230\n",
      "Data columns (total 26 columns):\n",
      " #   Column               Non-Null Count  Dtype  \n",
      "---  ------               --------------  -----  \n",
      " 0   YEAR                 11231 non-null  int64  \n",
      " 1   QUARTER              11231 non-null  int64  \n",
      " 2   MONTH                11231 non-null  int64  \n",
      " 3   DAY_OF_MONTH         11231 non-null  int64  \n",
      " 4   DAY_OF_WEEK          11231 non-null  int64  \n",
      " 5   UNIQUE_CARRIER       11231 non-null  object \n",
      " 6   TAIL_NUM             11231 non-null  object \n",
      " 7   FL_NUM               11231 non-null  int64  \n",
      " 8   ORIGIN_AIRPORT_ID    11231 non-null  int64  \n",
      " 9   ORIGIN               11231 non-null  object \n",
      " 10  DEST_AIRPORT_ID      11231 non-null  int64  \n",
      " 11  DEST                 11231 non-null  object \n",
      " 12  CRS_DEP_TIME         11231 non-null  int64  \n",
      " 13  DEP_TIME             11124 non-null  float64\n",
      " 14  DEP_DELAY            11124 non-null  float64\n",
      " 15  DEP_DEL15            11124 non-null  float64\n",
      " 16  CRS_ARR_TIME         11231 non-null  int64  \n",
      " 17  ARR_TIME             11116 non-null  float64\n",
      " 18  ARR_DELAY            11043 non-null  float64\n",
      " 19  ARR_DEL15            11043 non-null  float64\n",
      " 20  CANCELLED            11231 non-null  float64\n",
      " 21  DIVERTED             11231 non-null  float64\n",
      " 22  CRS_ELAPSED_TIME     11231 non-null  float64\n",
      " 23  ACTUAL_ELAPSED_TIME  11043 non-null  float64\n",
      " 24  DISTANCE             11231 non-null  float64\n",
      " 25  Unnamed: 25          0 non-null      float64\n",
      "dtypes: float64(12), int64(10), object(4)\n",
      "memory usage: 2.2+ MB\n"
     ]
    }
   ],
   "source": [
    "data.info()"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 9,
   "metadata": {
    "id": "eZ2gYfSJmUR9"
   },
   "outputs": [],
   "source": [
    "data=data.drop('Unnamed: 25',axis=1)"
   ]
  },
  {
   "cell_type": "code",
"execution_count": 10,
"metadata": {
 "colab": {
  "base_uri": "https://localhost:8080/"
 },
 "id": "FD8frcdMmdJf",
 "outputId": "57489ec1-020b-4e56-bb47-a763bcf096ab"
},
"outputs": [
 {
  "name": "stdout",
  "output_type": "stream",
  "text": [
   "<class 'pandas.core.frame.DataFrame'>\n",
   "RangeIndex: 11231 entries, 0 to 11230\n",
   "Data columns (total 25 columns):\n",
   " #   Column             Non-Null Count  Dtype  \n",
   "---  ------             --------------  -----  \n",
   " 0   YEAR               11231 non-null  int64  \n",
   " 1   QUARTER            11231 non-null  int64  \n",
   " 2   MONTH              11231 non-null  int64  \n",
   " 3   DAY_OF_MONTH       11231 non-null  int64  \n",
   " 4   DAY_OF_WEEK        11231 non-null  int64  \n",
   " 5   UNIQUE_CARRIER     11231 non-null  object \n",
   " 6   TAIL_NUM           11231 non-null  object \n",
   " 7   FL_NUM             11231 non-null  int64  \n",
   " 8   ORIGIN_AIRPORT_ID  11231 non-null  int64  \n",
   " 9   ORIGIN             11231 non-null  object \n",
   " 10  DEST_AIRPORT_ID    11231 non-null  int64  \n",
   " 11  DEST               11231 non-null  object \n",
   " 12  CRS_DEP_TIME       11231 non-null  int64  \n",
   " 13  DEP_TIME           11124 non-null  float64\n",
   " 14  DEP_DELAY          11124 non-null  float64\n",
   " 15  DEP_DEL15          11124 non-null  float64\n",
   " 16  CRS_ARR_TIME       11231 non-null  int64  \n",
   " 17  ARR_TIME           11116 non-null  float64\n",
   " 18  ARR_DELAY          11043 non-null  float64\n",
   " 19  ARR_DEL15          11043 non-null  float64\n",
    " 20  CANCELLED            11231 non-null  float64\n",
    " 21  DIVERTED             11231 non-null  float64\n",
    " 22  CRS_ELAPSED_TIME     11231 non-null  float64\n",
    " 23  ACTUAL_ELAPSED_TIME  11043 non-null  float64\n",
    " 24  DISTANCE             11231 non-null  float64\n",
    "dtypes: float64(11), int64(10), object(4)\n",
    "memory usage: 2.1+ MB\n"
   ]
  }
 ],
 "source": [
  "data.info()"
 ]
},
{
 "cell_type": "code",
 "execution_count": 11,
 "metadata": {
  "colab": {
   "base_uri": "https://localhost:8080/",
   "height": 394
  },
  "id": "LVxbOlmp_7V-",
  "outputId": "2b749b0b-ec23-41e9-b1fa-9f231ab1bd7f"
 },
 "outputs": [
  {
   "data": {
    "text/html": [
     "<div>\n",
     "<style scoped>\n",
     "    .dataframe tbody tr th:only-of-type {\n",
     "        vertical-align: middle;\n",
     "    }\n",
     "\n",
     "    .dataframe tbody tr th {\n",
     "        vertical-align: top;\n",
     "    }\n",
"\n",
"    .dataframe thead th {\n",
"        text-align: right;\n",
"    }\n",
"</style>\n",
"<table border=\"1\" class=\"dataframe\">\n",
"  <thead>\n",
"    <tr style=\"text-align: right;\">\n",
"      <th></th>\n",
"      <th>YEAR</th>\n",
"      <th>QUARTER</th>\n",
"      <th>MONTH</th>\n",
"      <th>DAY_OF_MONTH</th>\n",
"      <th>DAY_OF_WEEK</th>\n",
"      <th>FL_NUM</th>\n",
"      <th>ORIGIN_AIRPORT_ID</th>\n",
"      <th>DEST_AIRPORT_ID</th>\n",
"      <th>CRS_DEP_TIME</th>\n",
"      <th>DEP_TIME</th>\n",
"      <th>...</th>\n",
"      <th>DEP_DEL15</th>\n",
"      <th>CRS_ARR_TIME</th>\n",
"      <th>ARR_TIME</th>\n",
"      <th>ARR_DELAY</th>\n",
"      <th>ARR_DEL15</th>\n",
"      <th>CANCELLED</th>\n",
"      <th>DIVERTED</th>\n",
"      <th>CRS_ELAPSED_TIME</th>\n",
"      <th>ACTUAL_ELAPSED_TIME</th>\n",
"      <th>DISTANCE</th>\n",
"    </tr>\n",
"  </thead>\n",
"  <tbody>\n",
"    <tr>\n",
"      <th>count</th>\n",
"      <td>11231.0</td>\n",
"      <td>11231.000000</td>\n",
"      <td>11231.000000</td>\n",
"      <td>11231.000000</td>\n",
"      <td>11231.000000</td>\n",
"      <td>11231.000000</td>\n",
"      <td>11231.000000</td>\n",
"      <td>11231.000000</td>\n",
"      <td>11231.000000</td>\n",
"      <td>11124.000000</td>\n",
"      <td>...</td>\n",
"      <td>11124.000000</td>\n",
"      <td>11231.000000</td>\n",
"      <td>11116.000000</td>\n",
"      <td>11043.000000</td>\n",
"      <td>11043.000000</td>\n",
"      <td>11231.000000</td>\n",
"      <td>11231.000000</td>\n",
"      <td>11231.000000</td>\n",
"      <td>11043.000000</td>\n",
"      <td>11231.000000</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>mean</th>\n",
"      <td>2016.0</td>\n",
"      <td>2.544475</td>\n",
"      <td>6.628973</td>\n",
"      <td>15.790758</td>\n",
"      <td>3.960199</td>\n",
"      <td>1334.325617</td>\n",
"      <td>12334.516695</td>\n",
"      <td>12302.274508</td>\n",
"      <td>1320.798326</td>\n",
"      <td>1327.189410</td>\n",
"      <td>...</td>\n",
"      <td>0.142844</td>\n",
"      <td>1537.312795</td>\n",
"      <td>1523.978499</td>\n",
"      <td>-2.573123</td>\n",
"      <td>0.124513</td>\n",
"      <td>0.010150</td>\n",
    "     <td>0.006589</td>\n",
    "     <td>190.652124</td>\n",
    "     <td>179.661233</td>\n",
    "     <td>1161.031965</td>\n",
    "   </tr>\n",
    "   <tr>\n",
    "    <th>std</th>\n",      "
<td>0.0</td>\n",
    "     <td>1.090701</td>\n",
    "     <td>3.354678</td>\n",
    "     <td>8.782056</td>\n",
    "     <td>1.995257</td>\n",
    "     <td>811.875227</td>\n",
    "     <td>1595.026510</td>\n",
    "     <td>1601.988550</td>\n",
    "     <td>490.737845</td>\n",
    "     <td>500.306462</td>\n",
    "     <td>...</td>\n",
    "     <td>0.349930</td>\n",
    "     <td>502.512494</td>\n",
    "     <td>512.536041</td>\n",
    "     <td>39.232521</td>\n",
    "     <td>0.330181</td>\n",
    "     <td>0.100241</td>\n",
    "     <td>0.080908</td>\n",
    "     <td>78.386317</td>\n",
    "     <td>77.940399</td>\n",
    "     <td>643.683379</td>\n",
    "   </tr>\n",
    "   <tr>\n",
    "     <th>min</th>\n",
    "     <td>2016.0</td>\n",
    "     <td>1.000000</td>\n",
    "     <td>1.000000</td>\n",
    "     <td>1.000000</td>\n",
    "     <td>1.000000</td>\n",
    "     <td>7.000000</td>\n",
    "     <td>10397.000000</td>\n",
"     <td>10397.000000</td>\n",
"     <td>10.000000</td>\n",
"     <td>1.000000</td>\n",
"     <td>...</td>\n",
"     <td>0.000000</td>\n",
"     <td>2.000000</td>\n",
"     <td>1.000000</td>\n",
"     <td>-67.000000</td>\n",
"     <td>0.000000</td>\n",
"     <td>0.000000</td>\n",
"     <td>0.000000</td>\n",
"     <td>93.000000</td>\n",
"     <td>75.000000</td>\n",
"     <td>509.000000</td>\n",
"   </tr>\n",
"   <tr>\n",
"     <th>25%</th>\n",
"     <td>2016.0</td>\n",
"     <td>2.000000</td>\n",
"     <td>4.000000</td>\n",
"     <td>8.000000</td>\n",
"     <td>2.000000</td>\n",
"     <td>624.000000</td>\n",
"     <td>10397.000000</td>\n",
"     <td>10397.000000</td>\n",
"     <td>905.000000</td>\n",
"     <td>905.000000</td>\n",
"     <td>...</td>\n",
"     <td>0.000000</td>\n",
"     <td>1130.000000</td>\n",
"     <td>1135.000000</td>\n",
"     <td>-19.000000</td>\n",
"     <td>0.000000</td>\n",
"     <td>0.000000</td>\n",
"     <td>0.000000</td>\n",
"     <td>127.000000</td>\n",
"     <td>117.000000</td>\n",
"     <td>594.000000</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>50%</th>\n",
"      <td>2016.0</td>\n",
"      <td>3.000000</td>\n",
"      <td>7.000000</td>\n",
"      <td>16.000000</td>\n",
"      <td>4.000000</td>\n",
"      <td>1267.000000</td>\n",
"      <td>12478.000000</td>\n",
"      <td>12478.000000</td>\n",
"      <td>1320.000000</td>\n",
"      <td>1324.000000</td>\n",
"      <td>...</td>\n",
"      <td>0.000000</td>\n",
"      <td>1559.000000</td>\n",
"      <td>1547.000000</td>\n",
"      <td>-10.000000</td>\n",
"      <td>0.000000</td>\n",
"      <td>0.000000</td>\n",
"      <td>0.000000</td>\n",
"      <td>159.000000</td>\n",
"      <td>149.000000</td>\n",
"      <td>907.000000</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>75%</th>\n",
"      <td>2016.0</td>\n",
"      <td>3.000000</td>\n",
"      <td>9.000000</td>\n",
"      <td>23.000000</td>\n",
"      <td>6.000000</td>\n",
"      <td>2032.000000</td>\n",
"      <td>13487.000000</td>\n",
"      <td>13487.000000</td>\n",
"      <td>1735.000000</td>\n",
"      <td>1739.000000</td>\n",
"      <td>...</td>\n",
"      <td>0.000000</td>\n",
"      <td>1952.000000</td>\n",
"      <td>1945.000000</td>\n",
"      <td>1.000000</td>\n",
"      <td>0.000000</td>\n",
"      <td>0.000000</td>\n",
"      <td>0.000000</td>\n",
"      <td>255.000000</td>\n",
"      <td>236.000000</td>\n",
"      <td>1927.000000</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>max</th>\n",
"      <td>2016.0</td>\n",
"      <td>4.000000</td>\n",
"      <td>12.000000</td>\n",
"      <td>31.000000</td>\n",
"      <td>7.000000</td>\n",
"      <td>2853.000000</td>\n",
"      <td>14747.000000</td>\n",
"      <td>14747.000000</td>\n",       "      <td>2359.000000</td>\n",
"      <td>2400.000000</td>\n",
"      <td>...</td>\n",
"      <td>1.000000</td>\n",
"      <td>2359.000000</td>\n",
"      <td>2400.000000</td>\n",
"      <td>615.000000</td>\n",
"      <td>1.000000</td>\n",
"      <td>1.000000</td>\n",
"      <td>1.000000</td>\n",
"      <td>397.000000</td>\n",
"      <td>428.000000</td>\n",
"      <td>2422.000000</td>\n",
"    </tr>\n",
"  </tbody>\n",
"</table>\n",
"<p>8 rows × 21 columns</p>\n",
"</div>"
```

```
     ],
     "text/plain": [
      "        YEAR      QUARTER        MONTH  DAY_OF_MONTH   DAY_OF_WEEK
\\\n",
      "count  11231.0  11231.000000  11231.000000  11231.000000  11231.000000   \n",
      "mean    2016.0     2.544475     6.628973    15.790758     3.960199   \n",
      "std       0.0     1.090701     3.354678     8.782056     1.995257   \n",
      "min     2016.0     1.000000     1.000000     1.000000     1.000000   \n",
      "25%     2016.0     2.000000     4.000000     8.000000     2.000000   \n",
      "50%     2016.0     3.000000     7.000000    16.000000     4.000000   \n",
      "75%     2016.0     3.000000     9.000000    23.000000     6.000000   \n",
      "max     2016.0     4.000000    12.000000    31.000000     7.000000   \n",
      "\n",
      "          FL_NUM  ORIGIN_AIRPORT_ID  DEST_AIRPORT_ID  CRS_DEP_TIME
\\\n",
      "count  11231.000000      11231.000000    11231.000000  11231.000000   \n",
      "mean    1334.325617      12334.516695    12302.274508   1320.798326   \n",
      "std      811.875227       1595.026510     1601.988550    490.737845   \n",
      "min        7.000000      10397.000000    10397.000000     10.000000   \n",
      "25%      624.000000      10397.000000    10397.000000    905.000000   \n",
      "50%     1267.000000      12478.000000    12478.000000   1320.000000   \n",
      "75%      2032.000000       13487.000000     13487.000000   1735.000000    \n",         "max
2853.000000      14747.000000    14747.000000   2359.000000   \n",
      "\n",
      "         DEP_TIME  ...    DEP_DEL15  CRS_ARR_TIME     ARR_TIME  \\\n",
      "count  11124.000000  ...  11124.000000  11231.000000  11116.000000   \n",
      "mean    1327.189410  ...     0.142844   1537.312795   1523.978499   \n",
      "std      500.306462  ...     0.349930   502.512494   512.536041   \n",
      "min        1.000000  ...     0.000000     2.000000     1.000000   \n",
      "25%      905.000000  ...     0.000000   1130.000000   1135.000000   \n",
      "50%     1324.000000  ...     0.000000   1559.000000   1547.000000   \n",
      "75%     1739.000000  ...     0.000000   1952.000000   1945.000000   \n",
      "max     2400.000000  ...     1.000000   2359.000000   2400.000000   \n",
      "\n",
      "        ARR_DELAY    ARR_DEL15    CANCELLED     DIVERTED  \\\n",
      "count  11043.000000  11043.000000  11231.000000  11231.000000   \n",
      "mean      -2.573123     0.124513     0.010150     0.006589   \n",
      "std       39.232521     0.330181     0.100241     0.080908   \n",
       "min     -67.000000     0.000000     0.000000     0.000000   \n",
       "25%     -19.000000     0.000000     0.000000     0.000000   \n",
       "50%     -10.000000     0.000000     0.000000     0.000000   \n",
       "75%       1.000000     0.000000     0.000000     0.000000   \n",
       "max     615.000000     1.000000     1.000000     1.000000   \n",
       "\n",
       "       CRS_ELAPSED_TIME  ACTUAL_ELAPSED_TIME      DISTANCE  \n",
       "count      11231.000000         11043.000000  11231.000000  \n",
       "mean         190.652124           179.661233   1161.031965  \n",
       "std           78.386317            77.940399    643.683379  \n",
       "min           93.000000            75.000000    509.000000  \n",
       "25%          127.000000           117.000000    594.000000  \n",
       "50%          159.000000           149.000000    907.000000  \n",
       "75%          255.000000           236.000000   1927.000000  \n",
       "max          397.000000           428.000000   2422.000000  \n",
       "\n",
       "[8 rows x 21 columns]"
      ]
     },
     "execution_count": 11,
     "metadata": {},
     "output_type": "execute_result"
    }
   ],
   "source": [
    "data.describe()"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {
    "id": "eQ0SsMAymESc"
   },
   "source": [
    "**Handling Missing Values**"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 12,
   "metadata": {
    "id": "1BIxOSi5lx5O"
   },
   "outputs": [],
   "source": [
    "data=data.dropna()"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 13,
   "metadata": {
    "colab": {
     "base_uri": "https://localhost:8080/"
    },
    "id": "iTwoByHLmJq5",
    "outputId": "39b90ac2-81fc-4cf8-f4e7-f6eda882aef7"
   },
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "<class 'pandas.core.frame.DataFrame'>\n",
      "Int64Index: 11043 entries, 0 to 11230\n",
      "Data columns (total 25 columns):\n",
      " #   Column               Non-Null Count  Dtype  \n",
      "---  ------               --------------  -----  \n",
      " 0   YEAR                 11043 non-null  int64  \n",
      " 1   QUARTER              11043 non-null  int64  \n",
      " 2   MONTH                11043 non-null  int64  \n",
      " 3   DAY_OF_MONTH         11043 non-null  int64  \n",
      " 4   DAY_OF_WEEK          11043 non-null  int64  \n",
      " 5   UNIQUE_CARRIER       11043 non-null  object \n",
      " 6   TAIL_NUM             11043 non-null  object \n",
      " 7   FL_NUM               11043 non-null  int64  \n",
     " 8   ORIGIN_AIRPORT_ID    11043 non-null  int64  \n",
     " 9   ORIGIN               11043 non-null  object \n",
     " 10  DEST_AIRPORT_ID      11043 non-null  int64  \n",
     " 11  DEST                 11043 non-null  object \n",
     " 12  CRS_DEP_TIME         11043 non-null  int64  \n",
     " 13  DEP_TIME             11043 non-null  float64\n",
     " 14  DEP_DELAY            11043 non-null  float64\n",
     " 15  DEP_DEL15            11043 non-null  float64\n",
     " 16  CRS_ARR_TIME         11043 non-null  int64  \n",
     " 17  ARR_TIME             11043 non-null  float64\n",
     " 18  ARR_DELAY            11043 non-null  float64\n",
     " 19  ARR_DEL15            11043 non-null  float64\n",
     " 20  CANCELLED            11043 non-null  float64\n",
     " 21  DIVERTED             11043 non-null  float64\n",
     " 22  CRS_ELAPSED_TIME     11043 non-null  float64\n",
     " 23  ACTUAL_ELAPSED_TIME  11043 non-null  float64\n",      " 24  DISTANCE             11043 non-null
float64\n",
     "dtypes: float64(11), int64(10), object(4)\n",
     "memory usage: 2.2+ MB\n"
    ]
   }
  ],
  "source": [
   "data.info()"
  ]
 },
 {
  "cell_type": "markdown",
  "metadata": {
   "id": "V7vJSkDeml7w"
  },
  "source": [
   "**DATA VISUALIZATION**"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 14,
  "metadata": {
   "colab": {
    "base_uri": "https://localhost:8080/",
    "height": 298
   },
   "id": "2aIs4yWwmlUV",
   "outputId": "8bc1a2b5-524a-4e9c-a88b-9a2dbf293fbe"
  },
  "outputs": [
   {
    "data": {
     "text/plain": [
      "Text(0.5, 1.0, 'Distribution of the Arrival Time')"
     ]
    },
    "execution_count": 14,
    "metadata": {},
    "output_type": "execute_result"
   },
   {
    "data": {
     "image/png":
"iVBORw0KGgoAAAANSUhEUgAAAYsAAAEICAYAAACuxNj9AAAAOXRFWHRTb2Z0
d2FyZQBNYXRwbG90bGliIHZlcnNpb24zLjUuMCwgaHR0cHM6Ly9tYXRwbG90bGliLm
9yZy8/fFQqAAAACXBIWXMAAAsTAAALEwEAmpwYAAA/yklEQVR4nO29e5wcZZno/3
259GQ7359nQx8qKLM5uG2AaNOOnWz0agTEfkt3I6wcmkCVX0BdynqQ9469b0i8iNVv
RuImmFUm3nMC3yejzt7+QVwEPBHtv4IP9gBVot3L24nEoz7CG6Hc1KVe4P8wkvTycAj
N8BfqCqf5xZKhpAf3+/Dg0NJb5+cMuwHalqTDkKMMkmoSCuxl3Yb0b7IMCnVyxKfa6Fi
DykqvqV2V4YmGMqpaBr4H3Aw8BLw51dPbgIHFfTaSMqYcYcLANysxQdHeZL


      "<Figure size 432x288 with 1 Axes>"
     ]
    },
    "metadata": {
     "needs_background": "light"
    },
    "output_type": "display_data"
    }
   ],
   "source": [
    "plt.scatter(data.index,data['ARR_TIME'])\n",
    "plt.ylabel('Arrival Time')\n",
    "plt.title('Distribution of the Arrival Time')"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 15,
   "metadata": {
    "colab": {
     "base_uri": "https://localhost:8080/",
     "height": 352
    },
    "id": "0RO320_-mKQy",
    "outputId": "812d4f0e-a880-4f84-8eff-1af88258e06f"
   },
   "outputs": [
    {
     "data": {
      "text/plain": [
       "(array([ 989., 1487., 1340., 1382.,  924.,  957., 1058., 1111.,  987.,\n",          "
808.]),\n",
       " array([   7. ,  291.6,  576.2,  860.8, 1145.4, 1430. , 1714.6, 1999.2,\n",
       "       2283.8, 2568.4, 2853. ]),\n",
       " <BarContainer object of 10 artists>)"
      ]
     },
     "execution_count": 15,
     "metadata": {},
     "output_type": "execute_result"
    },
    {
     "data": {
      "image/png":
```

```
"iVBORw0KGgoAAAANSUhEUgAAAX0AAAD4CAYAAAAAczaOAAAAOXRFWHRTb2Z
0d2FyZQBNYXRwbG90bGliIHZlcnNpb24zLjUuMCwgaHR0cHM6Ly9tYXRwbG90bGliLL
gLzwL4k24Bbq+qZqirgS8CDKzQGSdJNWolz+n8feKotbwcuDa1baLXtbXlpfaQkh5PMJZ
lbXFxcgS5KkmDK0E/yGeAN4MtvlUY0q+vUR6qqE1U1W1WzMzMz03RRkjTklkk3THII+
Fng/nbKBgYz+J1DzXYAl1t9x4i6JGkVTTTTT7If+GXg41X1/4ZWnQEOJtmaZDeDD2yfra
orwOtJ7mtX7XwSeHLKvkuSxnTDmX6SJ4CPAHckWQA+y+Bqna3A0+3Ky9+tqn9QVee
SnAZeYnDa50hVvdne6lMMrgR6N4PPAJ5CkrSqbhj6VfWJEeUvXKf9ceD4iPoccM9YvZ
MkrSjvyJWkjhj6ktQRQ1+SOmLoS1JHDH1J6oihL0kdmfiOXEn92XX0N9dkv6888rE12e9
m5Exfkjpi6EtSRwx9SeqIoS9JHTH0Jakjhr4kdcRLNqUNaK0undTG50xfkjpi6EtSRwx9Se
qIoS9JHblh6Cd5PMnVJC8O1W5P8nSSl9vzbUPrjiWZT3IhyQND9Q8meaGt+1z7A+mS
pFV0MzP9LwL7l9SOAmerag9wtr0myV7gIHB32+bRJFvaNo8Bh4E97bH0PSVJ77Abhn
5VfQP4wZLyAeBkWz4JPDhUP1VV16rqIjAP7EuyDbi1qp6pqgK+NLSNJGmVTHpO/66q
ugLSNYeLQT3Ir8GHgCwBV9SdV9b+AA8DJ1uwk8GBbPg
      "text/plain": [
       "<Figure size 432x288 with 1 Axes>"
      ]
     },
     "metadata": {
      "needs_background": "light"
     },
     "output_type": "display_data"
    }
   ],
   "source": [
    "plt.hist(data['FL_NUM'])"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 16,
   "metadata": {
    "id": "wPPYeGrNnim_"
   },
   "outputs": [],
   "source": [
    "columns=list(data.columns)"
   ]
  },
 {
  "cell_type": "code",
  "execution_count": 17,
  "metadata": {
   "colab": {
    "base_uri": "https://localhost:8080/",
    "height": 386
   },
   "id": "2UNMGd6uAVU0",
   "outputId": "8e939ae8-bce3-41dd-ef61-155f463be2a6"
  },
  "outputs": [
   {
    "data": {
     "text/plain": [
      "<seaborn.axisgrid.FacetGrid at 0x7f5bf6c57d90>"
     ]
    },
    "execution_count": 17,
    "metadata": {},
    "output_type": "execute_result"
   },
   {
    "data": {
     "image/png":
"iVBORw0KGgoAAAANSUhEUgAAAWAAAAFgCAYAAACFYaNMAAAAOXRFWHRTb2
Z0d2FyZQBNYXRwbG90bGliIHZlcnNpb24zLjUuMCwgaHR0cHM6Ly9tYXRwbG90bGliL
m9yZy8/fFQqAAAACXBIWXMAAAsTAAALEwEAmpwYAAAWAklEQVR4nO3df7RlZX3f
     "text/plain": [
      "<Figure size 360x360 with 1 Axes>"
     ]
    },
    "metadata": {
     "needs_background": "light"
    },
    "output_type": "display_data"
   }
  ],
  "source": [
   "sns.catplot(x='ARR_DELAY',y='ARR_DEL15',data=data,kind='bar')"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 18,
  "metadata": {
   "colab": {
    "base_uri": "https://localhost:8080/"
   },
   "id": "qCSXZMrZnoqh",
   "outputId": "02e886e1-148f-4458-f323-fa3b83a4efa1"
  },
  "outputs": [
   {
    "data": {
     "text/plain": [
      "2"
     ]
    },
    "execution_count": 18,
    "metadata": {},
    "output_type": "execute_result"
   }
  ],
  "source": [
   "data['ARR_DEL15'].nunique()"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 19,
  "metadata": {
   "colab": {
    "base_uri": "https://localhost:8080/"
   },
   "id": "1tg8rffzoEKB",
    "outputId": "0697f47c-3aee-4823-99e8-e3b0c3044aef"
   },
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "The skew of YEAR is 0\n",
      "The skew of QUARTER is -0.07076578197761728\n",
      "The skew of MONTH is -0.06478404626399789\n",
      "The skew of DAY_OF_MONTH is -0.0039212208433480715\n",      "The skew of DAY_OF_WEEK is
0.02611051500245296\n",
      "\n",
      "\n",
      "The skew of FL_NUM is 0.17421702805934322\n",
      "The skew of ORIGIN_AIRPORT_ID is 0.1781563550685908\n",
      "\n",
      "The skew of DEST_AIRPORT_ID is 0.20849436349039438\n",
      "\n",
      "The skew of CRS_DEP_TIME is 0.06149721892776374\n",
      "The skew of DEP_TIME is 0.03000709701894307\n",
      "The skew of DEP_DELAY is 7.1602009024202795\n",
      "The skew of DEP_DEL15 is 2.0463588064693035\n",
      "The skew of CRS_ARR_TIME is -0.40688020169034556\n",
      "The skew of ARR_TIME is -0.4130536257588298\n",
      "The skew of ARR_DELAY is 5.898519655640514\n",
      "The skew of ARR_DEL15 is 2.274840717112184\n",
      "The skew of CANCELLED is 0\n",
      "The skew of DIVERTED is 0\n",
      "The skew of CRS_ELAPSED_TIME is 0.9028927753685997\n",
      "The skew of ACTUAL_ELAPSED_TIME is 0.8903973027244532\n",
      "The skew of DISTANCE is 0.7844649071893438\n"
     ]
    }
   ],
   "source": [
    "for i in columns:\n",
    "    try:\n",
   "      skew1=data[i].skew()\n",
   "      print(\"The skew of {} is {}\".format(i,str(skew1)))\n",
   "      if skew1 > 3:\n",
   "        median = float(data[i].median())\n",
   "        data[i] = np.where(data[i] > 0.45, median, data[i])\n",
   "    except:\n",
   "      print()"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 20,
  "metadata": {
   "colab": {
    "base_uri": "https://localhost:8080/",
    "height": 394
   },
   "id": "Lma1Kiw-oXXi",
   "outputId": "8e5389f2-cfd3-485a-da6d-9557cbda49a3"
  },
  "outputs": [
   {
    "data": {
     "text/html": [
      "<div>\n",
      "<style scoped>\n",
      "    .dataframe tbody tr th:only-of-type {\n",
      "        vertical-align: middle;\n",
      "    }\n",
      "\n",
      "    .dataframe tbody tr th {\n",
      "        vertical-align: top;\n",
      "    }\n",
      "\n",
      "    .dataframe thead th {\n",
      "        text-align: right;\n",
      "    }\n",
      "</style>\n",
"<table border=\"1\" class=\"dataframe\">\n",
"  <thead>\n",
"    <tr style=\"text-align: right;\">\n",
"      <th></th>\n",
"      <th>YEAR</th>\n",
"      <th>QUARTER</th>\n",
"      <th>MONTH</th>\n",
"      <th>DAY_OF_MONTH</th>\n",
"      <th>DAY_OF_WEEK</th>\n",
"      <th>FL_NUM</th>\n",
"      <th>ORIGIN_AIRPORT_ID</th>\n",
"      <th>DEST_AIRPORT_ID</th>\n",
"      <th>CRS_DEP_TIME</th>\n",
"      <th>DEP_TIME</th>\n",
"      <th>...</th>\n",
"      <th>DEP_DEL15</th>\n",
"      <th>CRS_ARR_TIME</th>\n",
"      <th>ARR_TIME</th>\n",
"      <th>ARR_DELAY</th>\n",
"      <th>ARR_DEL15</th>\n",
"      <th>CANCELLED</th>\n",
"      <th>DIVERTED</th>\n",
"      <th>CRS_ELAPSED_TIME</th>\n",
"      <th>ACTUAL_ELAPSED_TIME</th>\n",
"      <th>DISTANCE</th>\n",
"    </tr>\n",
"  </thead>\n",
"  <tbody>\n",
"    <tr>\n",
"      <th>count</th>\n",
"      <td>11043.0</td>\n",
"      <td>11043.000000</td>\n",
"      <td>11043.000000</td>\n",
"      <td>11043.000000</td>\n",
"      <td>11043.000000</td>\n",
"      <td>11043.000000</td>\n",
"      <td>11043.000000</td>\n",
"      <td>11043.000000</td>\n",
"    <td>11043.000000</td>\n",
"    <td>11043.000000</td>\n",
"    <td>...</td>\n",
"    <td>11043.000000</td>\n",
"    <td>11043.000000</td>\n",
"    <td>11043.000000</td>\n",
"    <td>11043.000000</td>\n",
"    <td>11043.000000</td>\n",
"    <td>11043.0</td>\n",
"    <td>11043.0</td>\n",
"    <td>11043.000000</td>\n",
"    <td>11043.000000</td>\n",
"    <td>11043.000000</td>\n",
"  </tr>\n",
"  <tr>\n",
"    <th>mean</th>\n",
"    <td>2016.0</td>\n",
"    <td>2.548402</td>\n",
"    <td>6.643847</td>\n",
"    <td>15.813185</td>\n",
"    <td>3.964774</td>\n",
"    <td>1337.185276</td>\n",
"    <td>12332.628271</td>\n",
"    <td>12302.496785</td>\n",
"    <td>1321.007154</td>\n",
"    <td>1327.140723</td>\n",
"    <td>...</td>\n",
"    <td>0.142443</td>\n",
"    <td>1537.425428</td>\n",
"    <td>1524.224758</td>\n",
"    <td>-14.226569</td>\n",
"    <td>0.124513</td>\n",
"    <td>0.0</td>\n",
"    <td>0.0</td>\n",
"    <td>190.595581</td>\n",
"    <td>179.661233</td>\n",
"    <td>1160.944490</td>\n",
"  </tr>\n",
"    <tr>\n",
"     <th>std</th>\n",      "
<td>0.0</td>\n",
"      <td>1.091655</td>\n",
"      <td>3.353072</td>\n",
"      <td>8.789698</td>\n",
"      <td>1.990953</td>\n",
"      <td>810.832998</td>\n",
"      <td>1596.321443</td>\n",
"      <td>1602.485742</td>\n",
"      <td>490.705288</td>\n",
"      <td>500.631611</td>\n",
"      <td>...</td>\n",
"      <td>0.349520</td>\n",
"      <td>502.495992</td>\n",
"      <td>510.861392</td>\n",
"      <td>8.687823</td>\n",
"      <td>0.330181</td>\n",
"      <td>0.0</td>\n",
"      <td>0.0</td>\n",
"      <td>78.425024</td>\n",
"      <td>77.940399</td>\n",
"      <td>643.830437</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>min</th>\n",
"      <td>2016.0</td>\n",
"      <td>1.000000</td>\n",
"      <td>1.000000</td>\n",
"      <td>1.000000</td>\n",
"      <td>1.000000</td>\n",
"      <td>7.000000</td>\n",
"      <td>10397.000000</td>\n",
"      <td>10397.000000</td>\n",
"      <td>10.000000</td>\n",
"      <td>1.000000</td>\n",
"      <td>...</td>\n",
"      <td>0.000000</td>\n",
"      <td>2.000000</td>\n",
"      <td>1.000000</td>\n",
"      <td>-67.000000</td>\n",
"      <td>0.000000</td>\n",
"      <td>0.0</td>\n",
"      <td>0.0</td>\n",
"      <td>93.000000</td>\n",
"      <td>75.000000</td>\n",
"      <td>509.000000</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>25%</th>\n",
"      <td>2016.0</td>\n",
"      <td>2.000000</td>\n",
"      <td>4.000000</td>\n",
"      <td>8.000000</td>\n",
"      <td>2.000000</td>\n",
"      <td>629.000000</td>\n",
"      <td>10397.000000</td>\n",
"      <td>10397.000000</td>\n",
"      <td>905.000000</td>\n",
"      <td>905.000000</td>\n",
"      <td>...</td>\n",
"      <td>0.000000</td>\n",
"      <td>1130.000000</td>\n",
"      <td>1135.000000</td>\n",
"      <td>-19.000000</td>\n",
"      <td>0.000000</td>\n",
"      <td>0.0</td>\n",
"      <td>0.0</td>\n",
"      <td>126.000000</td>\n",
"      <td>117.000000</td>\n",
"      <td>594.000000</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>50%</th>\n",
"      <td>2016.0</td>\n",
"      <td>3.000000</td>\n",
"      <td>7.000000</td>\n",
"      <td>16.000000</td>\n",
"      <td>4.000000</td>\n",
"      <td>1287.000000</td>\n",
"      <td>12478.000000</td>\n",        "      <td>12478.000000</td>\n",
"      <td>1320.000000</td>\n",
"      <td>1324.000000</td>\n",
"      <td>...</td>\n",
"      <td>0.000000</td>\n",
"      <td>1559.000000</td>\n",
"      <td>1546.000000</td>\n",
"      <td>-10.000000</td>\n",
"      <td>0.000000</td>\n",
"      <td>0.0</td>\n",
"      <td>0.0</td>\n",
"      <td>159.000000</td>\n",
"      <td>149.000000</td>\n",
"      <td>907.000000</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>75%</th>\n",
"      <td>2016.0</td>\n",
"      <td>3.000000</td>\n",
"      <td>9.000000</td>\n",
"      <td>23.000000</td>\n",
"      <td>6.000000</td>\n",
"      <td>2032.000000</td>\n",
"      <td>13487.000000</td>\n",
"      <td>13487.000000</td>\n",
"      <td>1735.000000</td>\n",
"      <td>1739.000000</td>\n",
"      <td>...</td>\n",
"      <td>0.000000</td>\n",
"      <td>1952.000000</td>\n",
"      <td>1944.000000</td>\n",
"      <td>-10.000000</td>\n",
"      <td>0.000000</td>\n",
"      <td>0.0</td>\n",
    "     <td>0.0</td>\n",
    "     <td>255.000000</td>\n",
    "     <td>236.000000</td>\n",
    "     <td>1927.000000</td>\n",
    "   </tr>\n",
    "   <tr>\n",
    "     <th>max</th>\n",
    "     <td>2016.0</td>\n",
    "     <td>4.000000</td>\n",
    "     <td>12.000000</td>\n",
    "     <td>31.000000</td>\n",
    "     <td>7.000000</td>\n",
    "     <td>2853.000000</td>\n",
    "     <td>14747.000000</td>\n",
    "     <td>14747.000000</td>\n",
    "     <td>2359.000000</td>\n",
    "     <td>2400.000000</td>\n",
    "     <td>...</td>\n",
    "     <td>1.000000</td>\n",
    "     <td>2359.000000</td>\n",
    "     <td>2400.000000</td>\n",
    "     <td>0.000000</td>\n",
    "     <td>1.000000</td>\n",
    "     <td>0.0</td>\n",
    "     <td>0.0</td>\n",
    "     <td>397.000000</td>\n",
    "     <td>428.000000</td>\n",
    "     <td>2422.000000</td>\n",
    "   </tr>\n",
    "  </tbody>\n",
    "</table>\n",
    "<p>8 rows × 21 columns</p>\n",
    "</div>"
   ],
   "text/plain": [
    "         YEAR      QUARTER        MONTH  DAY_OF_MONTH   DAY_OF_WEEK  
\\\n",
    "count  11043.0  11043.000000  11043.000000  11043.000000  11043.000000   \n",
    "mean   2016.0    2.548402    6.643847    15.813185    3.964774  \n",
    "std      0.0    1.091655    3.353072    8.789698    1.990953  \n",
    "min   2016.0    1.000000    1.000000    1.000000    1.000000  \n",
    "25%    2016.0    2.000000    4.000000    8.000000    2.000000  \n",
    "50%    2016.0    3.000000    7.000000   16.000000    4.000000  \n",
    "75%    2016.0    3.000000    9.000000   23.000000    6.000000  \n",
    "max    2016.0    4.000000   12.000000   31.000000    7.000000  \n",
    "\n",
    "          FL_NUM  ORIGIN_AIRPORT_ID  DEST_AIRPORT_ID  CRS_DEP_TIME
\\\n",
    "count  11043.000000      11043.000000    11043.000000  11043.000000   \n",
    "mean    1337.185276      12332.628271    12302.496785   1321.007154  \n",
    "std      810.832998        1596.321443      1602.485742    490.705288   \n",       "min
7.000000      10397.000000    10397.000000    10.000000   \n",
    "25%     629.000000      10397.000000    10397.000000   905.000000  \n",
    "50%    1287.000000      12478.000000    12478.000000  1320.000000  \n",
    "75%     2032.000000       13487.000000     13487.000000   1735.000000   \n",       "max
2853.000000      14747.000000    14747.000000  2359.000000  \n",
    "\n",
    "        DEP_TIME  ...    DEP_DEL15  CRS_ARR_TIME     ARR_TIME  \\\n",
    "count  11043.000000  ...  11043.000000  11043.000000  11043.000000   \n",
    "mean    1327.140723  ...     0.142443   1537.425428   1524.224758  \n",
    "std     500.631611  ...     0.349520   502.495992   510.861392  \n",
    "min       1.000000  ...     0.000000     2.000000     1.000000  \n",
    "25%     905.000000  ...     0.000000   1130.000000   1135.000000  \n",
    "50%    1324.000000  ...     0.000000   1559.000000   1546.000000  \n",
    "75%     1739.000000  ...      0.000000   1952.000000   1944.000000   \n",       "max
2400.000000  ...     1.000000   2359.000000   2400.000000  \n",
    "\n",
    "       ARR_DELAY    ARR_DEL15  CANCELLED  DIVERTED
CRS_ELAPSED_TIME  \\\n",
    "count  11043.000000  11043.000000   11043.0   11043.0     11043.000000  \n",
    "mean    -14.226569     0.124513      0.0      0.0      190.595581  \n",
    "std       8.687823     0.330181      0.0      0.0       78.425024  \n",
    "min     -67.000000     0.000000      0.0      0.0       93.000000  \n",
    "25%     -19.000000     0.000000      0.0      0.0      126.000000  \n",
    "50%     -10.000000     0.000000      0.0      0.0      159.000000  \n",
    "75%     -10.000000     0.000000      0.0      0.0      255.000000  \n",
       "max       0.000000     1.000000      0.0      0.0       397.000000  \n",
       "\n",
       "       ACTUAL_ELAPSED_TIME     DISTANCE  \n",
       "count         11043.000000  11043.000000  \n",
       "mean            179.661233   1160.944490  \n",
       "std              77.940399    643.830437  \n",
       "min              75.000000    509.000000  \n",
       "25%             117.000000    594.000000  \n",
       "50%             149.000000    907.000000  \n",
       "75%             236.000000   1927.000000  \n",
       "max             428.000000   2422.000000  \n",
       "\n",
       "[8 rows x 21 columns]"
      ]
     },
     "execution_count": 20,
     "metadata": {},
     "output_type": "execute_result"
    }
   ],
   "source": [
    "data.describe()"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {
    "id": "TtesWXQEqXvY"
   },
   "source": [
    "**Dropping off unnecessary columns**"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 21,
   "metadata": {
    "colab": {
```

```
  "base_uri": "https://localhost:8080/"
 },
 "id": "CPNSN8OhqRnp",
 "outputId": "75c39545-90ec-457e-9a85-19ec4adf23b4"
},
"outputs": [
 {
  "data": {
   "text/plain": [
    "YEAR                        NaN\n",
    "QUARTER                0.023102\n",
    "MONTH                  0.030161\n",
    "DAY_OF_MONTH          -0.000597\n",
    "DAY_OF_WEEK           -0.012063\n",
    "FL_NUM                -0.002378\n",
    "ORIGIN_AIRPORT_ID     -0.006250\n",
    "DEST_AIRPORT_ID        0.029525\n",
    "CRS_DEP_TIME           0.086057\n",
    "DEP_TIME               0.127593\n",
    "DEP_DELAY              0.200721\n",
    "DEP_DEL15              0.658511\n",
    "CRS_ARR_TIME           0.078282\n",
    "ARR_TIME               0.042298\n",
    "ARR_DELAY              0.183476\n",
    "ARR_DEL15              1.000000\n",
    "CANCELLED                   NaN\n",
    "DIVERTED                    NaN\n",
    "CRS_ELAPSED_TIME       0.015676\n",
    "ACTUAL_ELAPSED_TIME    0.077741\n",
    "DISTANCE               0.002870\n",
    "Name: ARR_DEL15, dtype: float64"
   ]
  },
  "execution_count": 21,
  "metadata": {},
  "output_type": "execute_result"
 }
],
   "source": [
    "data.corr()['ARR_DEL15']"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 22,
   "metadata": {
    "colab": {
     "base_uri": "https://localhost:8080/",
     "height": 395
    },
    "id": "GdGBTQ2VAhdZ",
    "outputId": "00df90ba-58b7-4309-bca9-5e64d591c06b"
   },
   "outputs": [
    {
     "data": {
      "text/plain": [
       "<AxesSubplot:>"
      ]
     },
     "execution_count": 22,
     "metadata": {},
     "output_type": "execute_result"
    },
    {
     "data": {
      "image/png":
"iVBORw0KGgoAAAANSUhEUgAAAb8AAAFpCAYAAAAMdCZAAAAAOXRFWHRTb2Z
0d2FyZQBNYXRwbG90bGliIHZlcnNpb24zLjUuMCwgaHR0cHM6Ly9tYXRwbG90bGliL
m9yZy8/fFQqAAAACXBIWXMAAAsTAAALEwEAmpwYAABhYklEQVR4nO2dd7wkRbm
/ny9LDpIERECWqCKSXERFFFAUFQlKWq8iynV/3gsqmDBdc7qCioppVQSVCwYyAga
CgCCywBIVRDKIgAFQFtjw/f1RNbu9vd1zpvvMOTt75n349Ifuqnq7qufMzttV9QbZJgiCIA
iGiaUW9wCCCIAiCYLwJ5RcEQRAMHaH8giAIgqEjlF8QBEEwdITyC4IgCIaOUH5BEATB
0BHKLwiCIFhsSDpO0gOSbqipl6SvSrpV0nWStutHv6H8giAIgsXJ8cDuAN/vRaSi/IAiCYLFh+2Lg712a7AX8wInfAatJWne0/YbyC4IgCAaZ9YC7C9f35LJRsfRobz
```

DsSBJwCfAZ2+fmsv2BtwK7AdcXmp9s+/O5zVrAfcBhtr9duN8dwKOAgX8AB9m+c4Rh
RIy6IAh6RaO9weyHbuv5N2fZtTb5f6Tlyg7TbU9v0F3VeEf9mxfKb5TYtqS3Az+VdCEwCf
gMaQ37Wtvb1IjuB/wOmAp8u1S3i+2HJH0C+AjwtjEZfBAEQRvmze25aVZ0TZRdmXuA
DQrX65MmDqMiIj37gO0bgLOAI4GPkdan/zyC2FTgPcD6kuqm8JfTh+l9EARBX/G83o/R
cyZwULb6fAHwsO2/jPamMfPrH58ArgaeBKbkshUkzSy0+ZztH0vaAHia7d9L+glwAPClin
vuDpw+dkMOgiBowby+KDUAJJ0E7Aw8VdI9pAnEMgC2vwWcA7wauBV4DHhLX/qNlE
b9Q9IngX/Z/kK+/pftlSvavQ9YzfaHJW0FfM/29rnuDtKe3zrAA8ALbP9rhK7jjxgEQa+Mes/
vyXuu733Pb/3njrq/sSCWPfvLvHyMxFTg4KzozgS2lrRZoX4XYEPgRuCTVTeQNE3SDE
kzpk8fzXJ6EARBQ8Z32XNMiGXPcUbSM4GVbK9XKPsEcCDwqU6Z7VmSDgeul/Rp2w
v5wZQ2kWPmFwTB+NHA4GVQCeU3tpT3/M4DHgdOK7U7BTiZgvIDsP2XvB5+aLkuCI
JgsTHAM7peiT2/iUH8EYMg6JXR7/nd9vve9/w2fv5A7vnFzC8IgiBohOfOWdxDGDWh/Ia
tvXL47xzB0H59NJ1YaqXXnccxrLPNpCpmm4sln3XdK4j/+aMtKqen+Y3TCM2CPznmzc
x8pLLdtYpg0rL1X+J9Kd5VuEwxsvXviMvzRq3yaW04P3rTxyo7GgT/57tudIOoy0BTYJO
M72jZLenuu/ZfsPks4DriNFgfuu7RtG2/cSo/yyg/n3F/c42mK7dg8yCIJgiaKP787ZJuScUt
m3StdHsSDyVl9YYpRfEARBMBj0sJw58ITyC4IgCJoRyi8IgiAYNjwnlF8QBEEwbAxusoa
eCeUXBEEQNCL2/IIgCILhI2Z+QRAEwbAxwDlqeyaUXxAEdCIFnElBo5QfkPKpDbhJ
Boyu8Xr4dItInbcM7ecb3hkVlp2+Ubt20Rr+eaMLzSW8axHG8u8a6fPNGq/5lIrNe7jozv8t
bHMvCeb//3fclU5g1h31lSziDDjybS/NPuOtUneMsuPjdyoxEWNJSqYADO/gYntKelpkk7
OSW5vknSOpM0lzcqZpycFd9TgVdLmmP79J4HUcGEVH4Atm/LM6u1c9FOkmYWmrz
e9p87F5KWBl4FnDdWfVQwFTiJtFz6WUnL2J4NHA78HviT7R/2Op4gCILxoI+xPa8ENp
O0EXAvcCDwhoX78kadc0nHA2ePVvHBBFZ+meLrSd2S5AoFhXUJ8L0x6GNRoZR9/tX
AEbYflXQF8Arg57bvk3QBcHbDsQRBEIw5PRiy9lTtOZIOI1lxTgKOs32jpLFn+m/1paMKJ
qzyk7QxMJdkHfTsLk3n7+ONYR9V7E7aL7w+T+dXBB4Dfp7r5+Wjru/56+jf/va3mTZtWl3
TIAiCvtLP2J62zwHOKZVVvKj3bB/er3wmp/CStBXwLONa2+7lJ2sc+pgL/afukfL+VgNslrV
hnbVqktI4+ARKMBEGwpNCvmd/iZCIpv87yZccN4YcsbLVZ3o/7tO2fjVMfK0q6p1D+DeC
VJxpjWT4IgmDQmDPAy5m9MhDKLwiCIFhyGOTlzF6ZMMovG7acX1H1H1Mtt/62M/byE52
Rf5re1D+9XHeDDXg7kgMXecFko8Dv3Me7L5T8RSyzbfiZjT8FlmtQg71yZU2dLP3bWxz
DL6WaP2//acxn2MF082VBFtFMrsxaSGHDO/wSEruG3GoZ/vA98f636CIAgGlX6qXEm7
A18BJgHftf35Uv1/sMDY8F/Af9m+drT9ThjlFwRBEIwP/VJ+kiYVGYBXwd2A2wRBEIwP/VJ+kiYBXwd2A+4h+WyfabsY8v
F24KW2/yHpVcB0UkrRob7sy2z/I1/+jhS
DedTEzC8IgiBoRB+tPdejEMeZNPvrNqs7BDi3Hx2H8guCIAga0WTZU9IZ9I0oBiJY7rtTliqKj
VaOWGUtAtJ+b24qr4pofyCIAiCRjSxL86Kri4G4z3ABoXr9Vk0wAmStgK+C7yqX9b7sec
XBEEQNKKPsT2vBDaTtFEOVvaRnAKcb8qhJvtCzPyCIAiCRvTL2tP2HEmHH
Ab8guTocZ/tGSW/P9d8CPgqsCXwj5yOYY3vKaPsO5RcEQRA0op8hImyfA5xTKismN/9
P4D/72CUQym9oaRMZQpV70/Us1bA9wFw3H9d6y6zWWObe2f9s1H7lpZZr3Mdbrlqlsc
wkNd+JeOMTyzRqf+8yzZ/l1W+qyrncnabRWgB+fs03GrX/4vM+2riP8eJXj9zcqP3GK6zT
uI97n/jHyI3GgKZRhQaRCbXnJ2kfSZb0rHw9WdKsnB3hJkk/kLRMrttZ0sOSrpH0R0lHj3
DvgyU9mNv/SdIvJL2oUH+8pNsL2RguK8gdW3PPpSU9JOlz+foVWkfK8XVckH

QRAsDiZCVocJpfxI6YsuJW2adugkq30uyZKomD3iEtvbAtsCe0jacYT7/9j2trY3Az4PnFr
KIP++Qgb2XhTWK4Cbgf0lyfYvSXkJD8n17wCutH1ZD/cKgiAYFyKf3wAhaWVgR5LiOLB
cb3su8Hsq0hLldEUzq+rqsH0hyXx3NJlkp5Ji2t0FvCCXHQF8UNJzgMPoUwLdIAiCfhGZ
3AeLvYHzsins3yVtV6yUtDwpcsB5ZcGcFX4z4OKGfV4NPKtwfVRh2fPEboKSVgBeBpw
NnERShJ2M8scAlwOftv33hmMKgiAYU+bhno9BZSIpv6mkuHDk/0/N55tImgn8DbjL9nU
FmZ0kXQfcD5xt+/6GfZbfa4rLnv8xguwewIW2HyNllt8nB3mFFOh1Uk6EW92xNE3SDEk
zpk+v8x8NgiDoP3MbHIPKhLD2zLn8dgW2lGSSv4iBb5D3/CStC1wkaU/bHSfKS2zvIWlz
4FJJp9me2aDrbYE/tBz2VGBHSXfk6zWBXYBf256Xn6OWUtSEwX29CoJgwjHIM7pem
Sgzv32BH9je0PZk2xuQ0mDMj/6dlxM/AHywLJyXSj9Hg/01SS8l7fd9p+lgJT2FFJ/uGXm
8k4FDWTBbDYlgGFjC2nNwmAqcVio7BfhQqex0YEVJO1Xc41vASyRt1KWfA/J+3i353q
+3XZz5Fff8ZuZwPQAHS7qncwDvBC6w/URB9gxgT0nNnbCCIAjGkYlg7Tkhlj1t71xR9lX
gq6UyA1sXii4q1M2ii7Vn3n87vkv9wTVVXeUK8n8H1ipcrzySTBAEweJgIix7TgjlFzRn2fm
2Nb3TdJngMc9p3EebN8W5LaTWWbpZ9JXlW3xea7aYxK/a4p/kncs2+8tMe/OTjfu49oer
Npb5d4u/f9OILe+56pON+xgvnrfy5Ebt11pq+cZ9PGVSc5l+sOSrvlB+iyDpLcC7SsW/tX3
o4hhPEATBoDF3Aqi/UH4lbH8f+P7iHkcQBMGgMsh7eb0Syi8IgiBoROz5BUEQBEPHkq
/6QvkFQRAEDYmZXxAEQTB0hMFLEARBMHSEwUsQBEEwdDhmfkEQBMGwETO/IAi
CYOiY5yV/5icv4Q8h6Wmk5K/bA08AdwCH275F0hGkbA3r2H44t98ZuBDY0/ZZuexs4Gj
bF0laBvgU8Pp8v8eAj9k+N6cfepQFaaoutv1OSceT8gH+rDCuyblsy9J4jwdeCjycix6z/SJ
JBwNHAfcAKwO3AZ+wfVkPH8OS/UcMgmA8GXV+9Tdu+Lqef3N+dOepA5nPfYme+Uk
SKZvDCbYPzGXbAOsAt5CyPVwJ7MPCwaXvAT4MnFVx208B6wJb2n5C0jokZdVhF9s
PjXLo7ysqygI/tn1Yfo5dgFMl7VLKHBEEQbBYaRNPd9BY0lMa7QLMtv2tToHtmbYvkbQJ
aQb1ERbNk3ct8LCk3YqFklYE3ga8o5NuyPZfkLB+iCtsXkpLVThvvvoMgCLoxEVIaLe
nKb0vgqpq6qcBJwCXAMyWtXar/NEkxFtkUuMv2I136vLCQr++INoNm4bx/J3ZpdzXwrJ
Z9BEEQjAnzcM/HoLJEL3uOwIHAPrbnSToV2A/4eqcyzw6pSWzbjFc9iwzkGvlQRAMN
xPB1WFJn/ndCDyvXChpK2Az4FfZSOVAFl36BPgMae+vw63AMyQ1S/Y2dmwLVO73S
ZomaYakGdOnTx/nYQVBMMz0c9lT0u6SbpZ0q6QPVNRL0ldz/XWStuvHMyzpyu8CYDl
Jb+sUSNoe+ArwcduT8/F0YD1JGxaFbf8SWJ2c3d32Y8D3gK9KWjbfb11Jbxyfx1mApJe
S9vu+U1Vve7rtKbanTJsW24JBEIwftns+uiFpEmlF7lXAFsBUSVuUmr2KNJnZjPSb+M1+
PMMSrfycPtl9gN0k/VnSjcDHgZ1JVqBFTiPNAMt8Bli/cP0R4EHgJkk3AKfn6w7FPb8fFM
q/LemefFyey55ZKLtH0n65vLjnN7OjaIED8vUtwIeA14elZxAEg8Yc3PMxAs8HbrV9m+0n
gZOBvUpt9gJ+7gkYaZ1ji9/xs3wfs30O7dxcuLyqUn0lhby3/Ad6fj/I9Jtfc++Cabp
epKPtpTdvjWdgdIwiCYCDp457fesDdhet7gB16aLMe8JfRdLxEz/yCIAiC8aeJtWfRPiEfx
X2aKqO+smbtpU1jlviZXxAEQTC+NIkMZns6yWe5inuADQDQrX6wP3tWjTmJj5BUEQBI3o
o7XnlcBmkjbKtg8HAmeWZM4E3pyfwV8vCtm46kw4BfAJ
OA42zfKOntuf5bwDnAq0muaI8Bb+lH30t8YOsAiMDWQRD0zqiDZ7xs/Vf0/Jtz/j2/HMhg
HTHzC4IgCBoxyGHLeiWUXxAEQdCIiRDeLJRfEARB0IiJkMw2lF8QBEHQiCVf9YXyC4I
gCBoyZ6Az9fVGKL8gCIKgERPBSyCU35Cy74Z7jnkfj3tuY5lJLaywnzI/Lnjv7PnkCo3av/
AZzX1qp/1l+cYybVi14fPPbvHWvqaWayzzZIt+fvXIzY3aP2/lyY37OOOusxvLtGH2Q7c1F
HiicR9288942ac/p7FMmYlg7dlThBdJT5N0cs6ccJOkcyRtLmlWzkJwk6QfSFomt19R0o

mSrpd0g6RLJa3c5f5zS1kOPpDLL5I0pUZmH0mW9KxC2eTSmL4laal8fDWP5XpJV0ra
KMvckcs6fX81lx8v6XZJ10q6JT/fel2e4Yosf5ekBwv3m5z7eGpuZ0k/LMgtndufna8PLsnPr
EjxEQRBsNhwg/8GlRFnfpJESgd0gu0Dc9k2wDrAn21vk3My/YqUXeFE4F3AX20/N7d/Jj
C7SzezbG/TcOxTgUtJ4XA+XijvjGlpUr6/vYHlgKcDW+XM7usD/y7I1GVnf5/tn+XP4HBS
OqMtc+aHhbC9AyTlBUyxfVinLonP59/AlpJWsD0L2A24t3S7HxflgyAIBomJsOzZy8xvF2
B2DjMDgO2ZFFJM2J4L/J6UZgJgXQo/6LZvtt18Tl9DnkXuCBxCdY4+bM8BLgM2zeP5i/
Mage17bP+j1/5yHqkvA/eTEiuOlnOB1+TzqcBJfbhnEATBuNAkq8Og0ovy2xK4qlsDScuT
cjCdl4uOA46UdLmkT0vabIQ+Vigt8x0wQvu9gfNs3wL8XRVp7SWtCLwMuB74CfDafO8v
Stq21LyYoPaILv1eDTyrS32vnAwcmD+3rYArSvUHlD6PZhtUQRAEY8hcz+v5GFRGa/C
yiaSZpPTyP7N9HaSZoaSNgVcALweulPTCLlnJmy57TgWOyecn5+urS2MycIbtc2H+0uu
u+Thf0n62z88ydcueZfoSo872dZIm53GfU9Eklj2DIBhYBnkvr1d6UX43AvvW1HX219YFL
pK0Z86Mju1/AacCp0qaR4rKXaf8ekbSmiQFtqUkkyKBW1In8/qfqxRpXnY9FzhX0l9Js8fz
y+1GYNsWMnWcCRwN7Ays2VRYKSHkNIBvf/vbTJs2bQSJIAiC/jARIrz0sux5AbCcpLd
1CiRtD2zYuc65lT4AfDDX7yhp9Xy+LLAFcGefxrwv8APbG9qebHsD4HbgxXUCkraT9PR
8vhRpqbHn8SjxTtLe4Xkjte+R44BP2r6+jbDt6ban2J4Sii8IgvFkIlh7jqj8nMx69gF2U3J1uJ
FkXVnOpHs6sKKknYBNgN9Iuh64BpgBnNKlm/Ke3+cLdT+XdE8+fkpaKjytJH8K8IYu918
bOEvSDcB1wBzg2EJ9cc/vB4XyoyRdC9wCbE9aHl3E0rMN2ejmKzXV5T2/F/WjzyAIgn4
wz+75GFQin9/EoPEfMZzcw8m9CeHk3pwBdnIfte3C5mtN6fk355YHZ0Q+vyAIgmDJZ5
CXM3tl3JRfNlSpMhZ5me2/jdc4+oGkK0iO80Xe1Hb/bnGgFjOspl/4Nj4+bWZ+S6m5TNP
3ZfUUC2lhxm1VZRzeq9sYrLeR2XiFdRq1X2up8Zldt6LpTG6Z5rNrZj3aXKYPDPJyZq+
Mm/LLCm6b8epvLOlEcwmCIBhGYuYXBEEQDB1t9hoHjVB+QRAEQSMGOWxZr4TyC
4IgCBoxyGHLeiWUXxAEQdCIieAiF8ovCIIgaERYewZBEARDR1h7BkEQBEPHRFj2bO
G6GwRBEAwz45XMVtIakn4l6U/5/6tXtNlA0oWS/iDpRknv6uXeQ6zHzkzSXlNR2GVJQ6x
OAY2zPk7Qz8F7be0haB/gesEFuewJPDDfKtnAA/n4yHbL8+Jca8Gdrf9i0KfBr5k+z35
+r3AyrY/nq8PAt5s5Pis8h4DjbR0s6Hnhp7gPgMdtDE9h6qRbhSlZkUmOZZy7bLDLGg/et3L
iPWX6ssUybvZSnLNUsMsjsFpZ6s2gep7VNDNF7n/hHo/ZPmTS4EV4a+8K1iNaiFVZpL
NMP5s4bN2vPDwDn2/68pA/k6yNLbeYA77F9taRVgKsk/cr2Td1uPBTKj0KyXElrA/8HrA
p8rNTuk8CvOtkWJG2VQ5Z1ZI8Hzrb9s4LMVODS/P9fFMqfAF4n6XPlRLmSXgUcDrzC
9n05o/ubCl3gYBjHZc+9SDlPIU1aLqKk/HJKvb/k80cl/QFYD+iq/IZu2dP/Qk
sIdJiwSFXBe4p9D2um73yvL7AgcDr8hKrMMcYDpwRIXoB0mzzftyP4/b/k7DRwmCIFgsj
NeyJ7BOVm4dJbd2t8aSJpOSjl8x0o2HTvkB2L6N9Oxrt8pqN22h3ZCZpyZ8PZtSb+WdE
PFsVeTMUlamZTb9XDbj4zUfliWPatYZHPJ9i8kbQzsDrwKuEbSlrYfrLnHVODkfH4
1ML9HsnJcd8JzGowtlj2DIJgYGmyN217OmkVrK7+5XV1kv4qaV3bf5G0LvBATZYd
NunVrUpM5Qzv6zg5lLxQdr+u+3/s/0m4ErgJTX3mAS8HviopLsA4AS8HviopDuArwGvyhuuRY4BDgFW
KpTdCDxvlM8w5Z5++l7wB9sf6nXGw+d8zcAZ5QZ5++l7wB9sf6nXGw+d
8pO0FvAt4FiXdm0l7SppxXy+CrAJcpLwY8OXGt7A9uTbW9uTbW9IevPYu9jI9t+Bn5AUYPAV
+Q9LTc13KS3tnkOWxPtz3F9pRp06aNLBAEQdAnmix7jpLPA7tJ+hOwW+4jaaqks3KbFwH/BDRMO7ms5
Ehaddu1sF1U3oJahGFZ9lxB0kwWuDr8EKh6Q3gecKykOaQXg+/avrLmnlOB00plpwD/x

QLXiA5fBA7rXNg+J7tV/Dq/tRg4rtD+KEkfKVw/3/aTXZ4vCIJg3BivCC85D+zLKsrvI9tY2L
6UFimdh0L52a51BLN9EclYBdtHAUd1aXtw1Xmh7EzSNB3bKxfK/wqsWGr7feD73foIgiA
YRCZChJehUH5BEARB/wjlFwRBEAwdS77qo9nGZRxL1gFMmygygzquYX+WYX/+QR
1XW5lhOobO2nPIaGMGOqgygzquNjKDOq7xkhnUcbWRGdRxtZUZGkL5BUEQBENHK
L8gCIJg6AjlN7FpE/plUGUGdVxtZAZ1XOMlM6jjaiMzqONqKzM0KG+MBkEQBMHQED
O/IAiCYOgI5RcEQRAMHaH8JjCSVpP04cU9jn4jafUxuOdaOeh5v+/7lC51z+i33ERA0gs
W9xgmEpIimEkFofwmAJI2kDRd0tmS/lPSipK+CNzCCJmPa+63m6Rf1dS9u9tRI3OWp
DPrjhqZ79Y9K3BJTd11Ncf1kq6raC9JH5f0EPBH4BZJD0r6aM1H05F7pqQvSvp5Po6W
9Mya5hcV5M4v1Z3epZtGcpK+0U1h9oKkjSS9TtKzauqPKZy/q1R3fL9kgG9K+rak1XoZd
1sa/h2LcqtJ2j4f5UTV5bbPKpwvV6pbRMlLen/hfL9S3We79HNp4bwcWP/33cY4rITymxj
8ALiPlFPwOcDvgKcDW9l+V51QTuF0i6R/SfqRpC0kzSClDflmjdgqheO9petyLsMOR5M
yW3wJeHY+Lx5VLJ3HNP87KunZwMX5flXMI+Vp/CGwP/DafOyR/1/mcFI6lO1tr2l7dWA
HYEdJR1R1IOmFJMX0KMma7jvAv4ELa2YsxWjza3SpG63cHcBVkt7Q5Z4L30Q6vXC+
F3AB6XM6Q9LBFSLF3JZvLtVtVdNNG5nnAX8Afi/pTTVt5iPprYXz9SWdL+mfki6TtHmN
TNO/l5KWzQr7joLMHZKOk7RszfD+r3B+eanuGxXtDyycf7BUt3tNH7BwvtDnlOoaZZzwY
ChZ3iJk4Rn+Q8goWr/8KLNeD3DXAzsBypDyEjwDvatDvNS3G2pMM6R/sdOCnwCTgR
cDdwGtGkHsW8AngauBHpLQnS3d5/qdWlK9VN07gXGDnivKXAudWlF9ddV51PVo5Y
D3gx8D5wL7A6zrHSH8L4DJgo3z+1PJ3qqL9NT2OqbFMoX4L4GGSgnqk8/8RPqufAP+
P9GK/D3B+P/6Oue6TwInAKoWyVUgvn59q8fyLfMeath/t92yYj1gLniDkfbDOG979wIqSV
oL5CXWrsFNKJ4DTJT1o+ysNum3jJ9OTjNO/2mmSvkJ6Q98Q2M/270aQ+yPwMeBjkg4
g/TD9L9Wpqpax/VDFPR6UtExNF5sUPrOizG8kVflVrZ2Xg1U4J19322NsLGf7Xkk/Bz5D
msF10mgbOLVKpHC+tO3b830eklSVgnup/D1bqnDe+c7VpQ1rI4OkQ4APAB8Gvp6/D72
wue398/lpXZawm/4dIb1IPN/2Y4X2j0r6b9Jqy/9UyLjmvOq6TfsOq0nah/Q5rybpdblcQNel
2WEllN/EYFXgKhZe3rg6/9/AxjVyxX8kkLbB5B5l/brvrBbIyk4rLdpNIPYKVylvQ10thFmgFcD
byhs6xn+501fa1HWjraB/gHcASLJh3u0C1BcF3do11k/I1R9h0WLAcXzwEq9zXbyEl6Dm
mp+j7SD/Rfuty7w9aSHiF9xstJeprt+/MSXpViKn/Pri7U1f0wN5aRdBlpaXEn2/eX6paxPbs
ksr6kr+Y+1iq1qXuJafp3BJhXVHwdbP9LUt3zF8fWOSdfr1fRvvg3WSGfd9ov32XMvwH2
LJwXl/kv7iI3tIST+xCT9y/qvgC2/dZyoaTrCzKbArd2qrLMIvs4km5ngSKr6mcR5SypvD9U
FjqhQuY3JCXxE+BnwN9LMn8vtZ9L9Q+dgOVtL/LDKekB4OQamf1d9Ah9v
+Ral8eeC1tn/a4F6rAc+2Xd6jGhck7Wb7V4VrAbsAbyA9yzql9uXvypm2/yHpacA7bX+oo
o/Gf0dJ15K2Caq+xxfa3rpCpvH3OBgfQvlNUCRtCtRtkv8Mtp9R0/9dG8m8BUuTpNbvWmJ79v3vCFD8PUn3pKXbQG4yFSqk2LqNfQFtHgBtFl6ZsDq+ULDfYdUXLSBDY9AsdpiU/Zl3zSDTmNjtA7Txh3DQWEpm8WxQ2o5QLGvh3mYQsYLmtm7ia6VPk9bRVPA3knKp5bypzwVXpHTOqkwKV6vRlwW6jfIC73mS32G6UqAX
bnlNRfgcLFHPxi91RzHUz4CZ9N/oxK7zp17Wvm8E2kpO0nO0n8vkk4BXAVOCVwCW2
9+12v16QtN0IY7q6XNZGpiC7A0nh7UMy+jmUrNh6GnAXWr5c3UFaSu75Ja7FuMrGTe
VOKrcw8rL4w7a/Vyp/BzDJ9jGjHdtEI5TfBELSusABpB+MrYDPAafavr6m/TG/n7yru9
0k63vbBFTK/tP2KhuO62nbXH8EKmUttvzif/9D2mwp1je9X00erH5qGfTwJ3ECajd5H6Ye
z7s2/jZykl5D+9q8hmbfvCGxctVTXw7ivt/3cUtk84EbgwU7RwkPyrhX3aSPzGZK17l3ASa
Rl6xm2N2rxHB+1/cmGMpUvV21QjStPB9t7Fq/z53UP0Om//HlVKlhJNwDb2X6yVL4ccG
XVisywE3t+EwBJbyO95Qn9P+rH8T+AM258YQbRLpDMiCDOlaT3JkHmmY5R0Gu2mbvDJz2tL8q3QyLLvWsJoTsKbpgdD3E/7MMM7TcvXTRMPbAaSkysNFf8y35Zb7z67J3Vb8/1U3nftG9ZQ+Vbm4tNYC8ks9fbe8f8hnyYyr9

m6Kr7Tfu1AV8LSK8vcArwdmkZYMT7P9rxGeo43MNOBm0rOcbfvxLntqI/GfJCvNhej2ck
V6cVjk5UrSG23/KJ/vaPu3hbrDbB9b0f8LSVbKJwFXMPK/ha+RllZ/m2Uu7dHYx2XFlwuf
yMvGQYmY+U0A8izhcuA9tmfksttGWoaRdI3tbcvn+bpydiXpNpJ/XyVVRjJd9lc6Moss/R
X7L4+ly9gurLj9GsCypOXfmXVj6Iak59i+MZ+/tFtb27/pcp/1SC8p7waOtF12Rm4tp2QVuz
dwPcm37Azg+m7fAUmzSab7VT8C+9qu9NuUtFEez17AncBnR/psm8iUlm13BS4EXg5s
ULPc/Ui5rFMFrGB7kZf80ne//P1a6N9CobzNd3ISsFt+lq2AnwMndb5PNc8vkgKcCjwf+C
XwTWdr3BqZ64GX2/5rqXwd4NflWXwQM7+JwtNJs4Qv5S/7T6i3civSxgx9VZLTeN1sqc
pCdBZpltWExqbbtnepKpc0BfgqC890m/BD8kygm3Ir9XmK7dcXrrcj/ZjtRvIx6+nz6FXO9r
skHU4yDJlKcu14iqT9gXNqZlvXAUfbvqGi35fXjcn27ZLOAFYA3gRsDszs9hxNZGzPJT3r
udlgZw9gReBeSefbLjvy/5MUqOCvpXIk1c36u73119Wp5rzqOt0oPct5wHl5CXIqcJGkT9r
+Wo2MSc7215D27T8F/Ilk9VvHUcDPJb2HBRa1zwO+QH1QiKEmlN8EwMlX7ZuksFDrk
/7BPJAtAE9zhbVbpo3p+p2usAIdgb/V7W114WL6ZLpte4aklRv2X6TNstHGAJI+Qfrx/gN
p9vvBXvaT2sjlH80LgAuU/BR3J/3YfoPkuF7mcJLzeBX7VIxpY9J3ay/SUt7JwGdsP97lO
RrLlJ7pcZLl7s+UwrcdVtHsByQ/0EWUHwtHWCnSxi+ulQ9eVnqvIf0tJpNexCrdiJR8c/ciL
Xevldtt12XpPnVu/0DSg6Ql3o6B2w3Ax2yf2012WIllzwmMUpzCA+o2/NtYYdYtCY0g8zv
biy1YcZ4Nn2P7eS3l2xjsXG17u2zAcBtp9gsLfiRrXUOyfCu5mnt90PbnGo7/cJcsBPOYri
MtqT5C6Qff9pcq7tNYZoRx3WW758Dekp5u+76K8u93k7P9lgqZx0iuPQI2YWE3n41tr1
QhcwJJGZ0LnFw1yy61/zdplndSvn/58+qL720QM78JgaT32/5CPt/P2afL9s152aiO06jY2
B+BN7YY4v7qkonA9l3lspaWqB3H+CJrkEKjvavFuPtBYwvFUcpV8V8ky98mvBs4plT2S
RZ8vr3OpNvIdKPpLPx3wCLfvSrlNr8D6fU1Vc9u2DekJd5/k5Z531mwPem8xJSDkf+U9
Hk9Kx9F6rYVUPdg7Lb9qYbjnvDEzG8C0GYjPte1mcV1HNbnFxWubXuTCpmOY/xCZtu
kZZ21bS+yv9jSuKDsu2XgbyRT7wcq2j+jSvFWtGs8c23z2Y4Vku62vcFYy4wHLWZ+bZ6
9sg+1cPMZL/JeX5mVgEOANW3348VjQhEzv4lB4434zHrq4kxdZYUJTCldL0XyyXovKV
B01X3K/mKTgSNJFnx1aVq6PVMlLfYVT6eHmW9R8dXNOis4Mrd/lOr9oLo3/04/reRq6E
sM1pqZ9QKBaqvdfsoIWK3uXnVdNGzf6aeKxm4+auhLqpq0YIX2lcvEtudnR5G0Cmml4
y2kPda6zClDTSi/iUHbYLiNrTBt/w1AKdXQm4D3kaz2XmP7pm6ykjYjBSregfQP8p1eNE
5jh8aWqNnVoVu4tpeVRbqNt4ae9tps/zKfrtHlGbvRSG4EZblCn2Rm9DqeMZRZpK7Pyp
KaewGsqnrfyLr9uKa+pHVpwUYkK9p3A/8BnEAylBl1NJyJSii/icFWahcMt7EVZrYifCspY
PSlwF62/zyCzJYkpfcckun1IdkEvBttLFGr/A9fALwfWGTZk3Yz3xUlbUuN4vSi4bquoPm+
amM51/jk9VnmmV0sh/sm0+07qeqs5I2UZb5PMUbtQlVUO/hDOzefnRsalf3N1c7yXZF0
FCnrxHTguTWuLUGB2PObAEg6F/hvd3GCrZGr3MvKjrkH2j6xoq4TeukYUkSRhah6+1U
KIH03ycF3EaVXo2RGhZIz+v+QchV+tsrcW9KdQK2hQNWPcJ4tXUnND6BL4bra7v01lZ
O0q+0L8vlGxe+CpNfV/F0ayYzG6rWhTN9C26k+DmybGLVj/vxt+shy84AnSP82q+LaNlk
mHwpi5jcxOI7kRHsCcFSD5bJXSvogKbXKmcCvSH5U7yUtZS6i/IBfk/5xbZ2PInVvv039
AlHKMv5bUpLVK10RuqlG7pUkpfc4yZ+sKupLhzb+h7eWFdwIrNVtH6eLqX9TuaNZMFM
8hYVnjR+h+u/SVGaRdFSlMVXFQm0j0yi0XTdlSU2osm6zMUm/JcVFXaSqi8yGNfccl9B
itpcaj34mEqH8JgC2fyrpHNIsZoakH7IgkWm3H9gfkHLeXU6Kgfg+UiiwvVwTeqpHY4+yz
HwFo+Rsbtt1OdM6fJfkovAZ0rLuH1mgDC9zdTSPK0lGCUeRnqkTIaUzjvKSZKVC7Tbz
7YaklSqeaxLJxL/pj2BTuTZGT01lnsWieSM71OWNbCPTNPpK4ziwI1BnTfomSS8kvSxe

```
bPsBSVuRku7uBFRZlTZdWu9sYZRpPINTcpjfG3iD7df0KjcshPKbOMwm+RMtR9o0r8rE
XWbjjiWmpO8CDwHPsN0t0WdnD+99pB8aAzeRwmRVZo/IMv8FfJD8QyXpX8D/2v5GV
XvbZwNn57aTgG1J8Q6PIvnAVRm9/Bv4F7AvKZhy2bWiPGNrM/M9UinW5rrAdbaflLQ2
KVrKwaRQc0X+4oZZBVrKtTF6aipzU4sl3DYyTaOvtAlV1o06mTeT9vxmkr4HZwP/TbJYrl
vdaGpUdv1oXGSUEhG/mpTdY3fSjP5bbe83kQnlNwGQtDvwJdIP+HbuPYXN/OVR23O
VsgCMpPj2Ii2XfY5ksSlSDMFTJb3X9hkVMh8hzeJ2tn1bLtsY+IqkNWx/uqavp2a5F5EM
V5YnLbtWJlm1vXP3x51/306y1MYzX1JW+RNJ0TeWUwoo/aV8r6oIMj3NPCStXrLMayq
3sVL6HBXOO/epc5hvIzMe/IZmoe0ahyrrYrVZax1LClG2rVOWidVJqaa2sv2n2idpt7TeG
Emd4NmvJAUC/yHwfHdx5h92wuBlAiDpEuDt7hIpvkaumMm884/+MbossShls97L9h2l8s
mkNErlfUAk3Qxs7VI8R0krANfa3rxC5k/Aw6Q319+R9v36YsGmBaHH5uesy7PLEWe+k
m4CXmz770pRa24FXmL7dzXt16jZ16ocU1s5tcg20VRG0sG2j+9hTF+z/Y62Mk1Ru1Blb
WSuciFEnqSZtrcZYWw9BUhQzhwi6UO263xfi+0XClmXDV4uAQ7uGC6ph8wuw0wov6
ARkm6yvUWTOkk3235mjcwfbZfDOJGXI19AWpK8hTQ7uxy4xiO7SYz0DNfY3rZC4Yxo
aVchc4PtLbvJNBlTP+QkrQVg+8Fqqcr7NJbpcq9RW0UqxaWdxoIQX38Aptu+ZbTja4ukf
7LwzPMlxWsvmsuxyb1HZRWq5H5zIGnJ/zaSc/tHbXe1ah1mYtkzWIgeNslnqyIsWDYdr8
s6cI+kl9k+vySzK/CXKoHSW+3mpKXPtwE7SXrQdtdZywh03vi21gL/SFjgI9nNuGD9kgH
D2sXrCgOGpmNqJSdJJIOnd5DGv5SkOcDX6vYO28iMB9mo5FSSz9r0PLZtSamAXlc1
y26qLNUidiwp20KRfkZOaWqYs1B729eQIiwdKWlH0hLoskpuUKfZnt6fYU4cQvkFTTfJP
wb8WtJnWRC9YnuSxduRNTLvBM6QdGlJZkcW/UEpj21jUkLPHUgzwbVIb7ajxhUxRXv
gfaXrpnkKx4rDgReT8tp1lr02JqW5OsL2l/skMx58lJR8+KJC2emSLiB9/15VbNxGWbJwb
sc3A18pXNdF8bnGdmUKKHUJ3N4jTV9+FmqvQtxRpwzzv5X0TlIeyANJn0tQxHYcQ3q
Q/mEcB9wL/IhkWHBHD3Jbkww8riJFXvkhaU+vm8zyJIu4L5IMRA4Blu/S/jTSrPCPwPdJ
BilbjOJZdyicn1qq24Vk5XkoySin13uuDKzUp7/FNaORI731P7Wifq26e7eRGatnKcoAt3R
pd3NF2blVfzfgpcC5PfR3Tanu6hqZqwvn5/ci0+D5G8lXjLn132tYj5j5DTe/IG2Sv9gL3vy/0l
0EbF8LHNSkIydjl+MaiHwfeJtTot5aCpabI/FTsv+W7ddl2fVIM4bHWeCLtn82xNnH9r01ffb
stqGUUurtwKbA9cD3XJ2U9mWjlFum6rOy/aBSSLoq2shUooWdvEf8Do0g083iuMo/dB
MvPEsEksGOpLoZT+PYsSy81FgOWD1aZ/aegjgU+Gnpuk3c0aEmlN9w8zzSksivJXU2
ybsuBRbM4Stxxaa/WmQosN21nwL/S/LPG4mqH6djgW+6ZI0o6SBS9vNFlmTV3G3jBJJ
LySWk5botqMgt6EUtO5vKdfvxrKtrLKMenLwrPs+mMhuo2jFc+T5lmipLaBc7tlUAeaV4p
K9i4f3I84ovM144c8gupH3YZxbaH1tU8F7UIrRN3NGhJqw9AwAKm+SvJznxnuaKTXJJ
D5LidJ5ECr5c3nivMqk/nRQs+FTgx26YPX6EcV/jHqwkVZGjbQQr1Mo6NXTbKLlTLA38
3j1Y9TWV08JuKwtVkZaXF5nJNZVRCp7ccfLelBSEoOPk/e3yZzIKmTfXPScsGnNV0gO
kF7eq59jf9jrd7tcrSnFtv5Tve0Q+7/RzuCvyBkp6Osnv7i+kZebOfuTTgF1cyjIv6TWkl7JPk
hSySOHZPgIcZvucmrG1igk6zMTMLwAabZI/LddPJRnI/Bw4yV18DG3vLWlVctT5vKT3Y
+DkihlP46F3TiSdRf0Mc82K8rrUSEvV1cH8Jdxy2azsa1WmGEhgjtTz6lgjOfdovKOCM30
LmTZO3o1lysqty7g6voFlI6QidVkdXgmsYvtnpfI3AA/WLKV/hwUph4rnkMLxVfFZ0urCMa
V+3kkKFFFW9O8D9s5bCx1mSpoBfA2oVH6Mftl1+Fjcm45xLL6D9EZZPLYFNmggvxwp
pNeDwDt6lFmKpDgfAt7dh2coGiG8tNtRIftl0o/YSoWylUhK/6s1/Z0PvKyifFfgworyucAj+Xi
U5A7SOX+ky3O1kmvyeTWVAa4qlc/sQbaxzFg9C8mFo3P+O2CtijZPAy6vkZ/SYox/7FJ
```

XZbzTrX23ui1L12sC+wDP69fnPdGOmPkNN1V+Smtk14cDvfDb53wkLUd6o58KTAa+y
gh7CpJelNvvRMoDuI/tS9oPfT53FM5vd8n/cATeT3r7vlMpvZGBDUn7bXU56Bq5bbidO0
VruR5oM0PoyGxS2vOdXLx2tZN3G5mxopipYUVXOPTbvj/7ulbxHaXA7CeRVi26Jm/Oz
OpSVxWGsFvA9251n5f0Ads3SFqXtGQ6g/T5T3dp5hnEnl9QgaQpwJdsv6Si7gRgS5J5
+cm2b+jhfncA/yTty1xAyRnei2ZbQNJnnZOg9mrRWdz3kHSK7dePJJPbrkDajxIpZVHX2
Kh52fYNpMDeAm4ETnT1/lXjPHujkRuJNntDGscQam3G1aa9pFtIrjNzSm2WIQXj3qzmH
s8kbQkcQDIK6ijCyn3sbEhWlWRZwBdsb1Jq/0+q45eKZJW9ek0/N9p+Tj7/EPAs2wdJW
gX4re0638WhJZRfUEndD0ve1+q8gRa/PN3igV5UaGsWnn3YFfnxSj9UPf3IFY1fejGEkb
QZKUj3JiR3gve6xr2hJLc32QXB9i9GaFv7HN2eq61cD2Mfk4SsknZ02jduct/GMiX5Ef/Gp
fbFz/TzwDokI5J/57KVSKsYD9muC9hQvN/WJEW4P3C/7UVyAEo6ni6WoC7FEG37sqB
CnFFJ5wPfsX1yuS5YQCx7BosgaR1q/sG6RdJMN8+20JZupuhVHEdy1r+YlEXgaySjnF
okfYM047sM+JSk59v+VDeRmvOq637IjUTrZU+l4N/7k9wNzstLbHuQlohXIO0ZLyzYQqb
rQFr4E5afI/MR4NMsWPIWye3ie6RkyCONYylgbZICXYm0770Ibp7/8hq3iyJzt6R3APeQ
9u/PyzIrAI18NoeFUH5DjKSvsaiSWIPkx7aIT9k4UPTZW1spk7kK5/NxdYLeYqzOTpxOq
J+VrmL7O/n8KEmLLL9W8BKSq8NcSSuS/PC6Kb9WvmFN5dTCmb6FzPdICuL3wFez0
ngh8AHbp9c8RxuZVv6ENfepVJb5OT8g6ROk54e05N1tjw5JO5H2rvcGbiAt5R9h++Ga9
se4WQzRi8iZ5yWdb7sY/OB0KrLSZw4huUe8HDjA9j9z+QtIASOCEqH8hpuOGfhKpO/C
qqSoL++2/cBiGE/xzbybWXklLYxElleKhl8MbD3/umovEnjSOauE7cc0su9C25x5TeXaON
M3lZlCclOYlxXnQ8Cmtu/v8hyNZbSwb2BPSWObKktJi+xnA9t3/py2F9l3k3Q3cBdJ4X3C
9l+7PHeHpjFEW0WRyf9e315RfiHJzzAoEXt+Q0y26vwCKVTZHeRZFskk/POStnWKFj9
e4xlXR11J3X4U6vYiHyPl8IP0eW2Srzuzy61K7dvu4TSSUwtn+qYybfYeW8rcRErK3JNv
oNo50p9VcSuT4tauX/UiVZpF9kS3feiqz2IUe8R1Pq7pwcbXqnaJIGZ+w83RpH2XDZ0Tu
Ep6CnC0pG+SMjx0m52MKZJeRYqhuQXpH/ZNpBiadY6+jbC9SwuxZzfsY76SUoOceS3
k2jjTN5V5lqTrOsMimdFfR43i70FmniuSHwOzOgrL9j+Uou3025G+mB0eSS8GPkyKxHJ
YjdixkpoqmKYxROuW+0UKOF7H0V3qggpC+Q03rwY2c2H6b/sRpcDND1FKHTMO3N
E5kfQ24P+RfPE6y7NTSP5M63sM85NJ2g14v+3dynV1b/7ZsONA4M5SuWiRM6+FXGe
/Exbe86y1wm0h00jxd5ERsD71vpRNfQObKssFA5FeRjJwMfDZEQyu2iiYpjFE20SRaf2S
NdR4ADzt41g8B91Tx9TWtejns4Xz3XqUuQlYo6J8TeAPfRrXrqQs8f8ipXTagqRorwJeVy
PzFNJs9FjgFaQftXeQlN4ZFe2PIBnxbFQo25i0t3pEl7G1kltM36Mdga/30G4b0jL7HaR9q
MNq2jWN1PNP4MzCsdB1TR+vIVnsngvs2OKZ16IiQsxi/jt8jPTS+jfgHyQL1I8u7nEN6rH
YBxDHYvzjJ+uxgyrK31j1Qz6Kfq6uOh9BplbB9VH5XQPsTArTtjcpfNi7RpA5AzieNCv9S
VZQvwG26dJH45x5TeWAXQvnG5Xq6hR5Y5lCfVmRVYa3AzYnzWD/QIrs8w7gzlH8zR
ZRVE2VZZaZRzJeOaukOGsVZpZrpGBIEYNWLVzvQjJ6OQJYtqL920irMZBerI4DHgauI
y3t1vWzxLwsDcqx2AcQx2L84yfruCtI5tVfJC3r/IZklr5eH/tpo/yuoCJBLskg4ff9Hle+/nMPM
tcXziflH8BVurS/YTzqun3GdZ95U5k2iiwrmd+QLDw7ZbeNIDOJ5E7wXnLMSpJBy2U0T
NpapSxzeRuF2VjB5O/x0/P5Nllxvodkafvdqr8tKc8ipChCV5FWO14OXNLlOVu9ZA3zEXt
+Q4xTNJMdJO3KglBd59o+v89dtfHZew9wpqTvs3AMzTeTZqb9YDUtnABUxWtXhxArG
onMlXS7s7FQDW3y7LWpa+MU31TmjyS3iNfavhVA0hFdxgkpRdaBwIWSziO5CYxkWd
PIN7CNI73rrWw3yOOtqj+ItGw/PwGw7dskvRH4JSlQepkVvCBt0RuB42x/MTvJz6xoP8d

25zu2B/AD238j5dz8QtWYM31LTDwshPILcIohecEYdtHGZ+9SSc8HDiVljujE0HyBu/uV
NeE3wGtrrk11sO6mRiLF9kUELN9lbE3lXHNedd1WprEis30acJpS6LC9SbOndbI18Wm2
f1kh1tQ3sJUjfQdJTwX2I8021wNOq2naRsEUP59dSfvF5Geraj9PKTD1P0jBBT5TqFuhy
2O0fZEaWsLPL1iiUYMA1oOMCnn2Wsr/kxSmTSTH7o6TtqgJiNxGJst1FNlU0g/6CdQrsi
r5NUjK5gCPENe16rqi/Q00d6RfhZTy5w2k5dzT8njW7yLTUyzWUvlXgHVJLhR7Apvbnp0
V3Fm2p5Ta7wF8m7T0e5btt+Xyl5IskF9T03/jZMbDTii/YFwYK5+9suNwQ9l3d6uvWo7N
P9zdZFol5x2tg38bZ/q2DvileyyiyPqgyLsFEljEN7ClI/0s0kzxI8Clti3pNtsbd5FprGCyy8oBJ
AX4k7zVgFIkobVdERg9BxxYpfgZ5hcO2f5Xt+cKeieWPYMxZ4x99kbz9tZt+bXuvp39x6o
1K5MMINrQk1d6HW7h59VGpuIefyfNVL5dKD6f+hiUvdDUN7CNI/2HSEu43wT+T9KPR
xqUW+RYzEr1cdJMbkvg3lx+TVV7pUwjRwGbSpqfacQ580TQP2LmF4w5SuGqXlyeFUl
ak/TW3cZ5unOPMQmJJml721f2+75d+hvtzG8Rp3hS3sRaZ/o2Mj2OpfVsvOJe25CWJv
cHbgdOsX1sqc2GVaJkZWn71V3uvzFp+fZAYDPS53G67Vv6NP5vklY7LiPt4Z3lLllAJF3
CwplGXmi7a6aRoB2N09MEQQtUtRyYrdhGfe8+3CPdSNpC0icl/Yk0I+hVbhNJH857T4u
Lw4EXA9vbXjPv1+0A7NjFIrONTC+M6o1a0uaSPirpD6RgAneTvkO7lBUfpKg7nQNYnW
QkdREp20blsrqkTZXyCd5m+zNOMU6fTwrp94fRjL/ETiR/yg+SfEr3HqH9Kra/Y/tm20cBk
/s4lqBAKL9gPHhEKfHnQuSySjcBSe/PJuwjMWLS0W5I2lDSByRdC/yQFBB5t7IhQoXcu
pIOl/R7khXq0qQZROuhjEIWkhn+VBcyvtu+jWRef1AfZcaDP5JmSa+1/WLbXwPm1jVuqi
wzx1D67tm+jvR9OrcPz9BhoSwgjPx3Xl7StpK2k7QdOdNI4TroE7HnF4wHbXz2NgSukn
Sou2T77tXCsApJl5FiL54M7Gv7T9lv744uMm8jKbn1SRFe/pMUDecTNe1XBGZ3fLckPZ
MUU/VOL+xH+LIq+Qa0McMfK9+w0Srypi4VbfwPJ2dltxC2r6xZRm1L02DgfwGKhlb3F6
5Nsq4N+kAov2DMcQufPduH5jfdr0n6I2kZcl6hvpfEsyPxIEmJrUOKhPEnRl6y+zpwOfAG
2zMA1CXSPymj9iHAnyRtmmVPBPbI+4odv69WVqIF2vh5NZIZL0Xu5r6BbRzpu/lYdvOn
a0rTLCC1mUZG+UISlAiDl2BgqPLZk7QzcAop03jny+oq/7CWfa5K+vGcSsoFtxrwStu/r2lf
dIhehzT7O9j2BjXtiznzPkUK1n2oUi7Fqzp1fXiONmb4jWQkXQwckmfIm5JcBU4kGXT8v
qPIx4Iql4qKNj37H0o6CbjA9ndK5YcAr7B9QH+fYJH+dyS9QB06QjuR4oG+gTSzXWcsx
zVMhPILBgYtnPhzbVK80Y2B/7Z97Tj0vzbJJ2sqsEGdQiu0X58045gKrEj6of1Qqc11naUt
Sb8FjnKOOiLp2hoz/IFkvBR5PxhJWUpah+TY/iRpKR6S+82ywD51KxKjHNM2LGy5emr
ez6xqu0Nuuw8po/uhpIDbrf0ng4UJ5RcMDFo4i/VtwOeB77j0JR0PNwQVsnZL+prtd4zQ/
pnAgZ29P0m72f6VpB+R9m3uBT5ACor8mKTVgN8sYcpvwijyDpJ2IfnfAdzoFOqvn/ffnA
UvSH8Dfkzy3avcV5T0GZJyvAs4iaSgZ9jeqJ/jCkL5BQNESfmt5YLDtaQtWPAj8vBI1phjN
a6mMpJWAN5FivBxXGcGK+lFwCa2f9j/EY8NE0mRjxeS5pGMcQ4pGOPURpKR9CBw
M8ka9Wyn7PRdI88E7QiDl2CQmG+kkC0ONyQpu6kk5+sNgSndrDEHCAHYnkWawS6
E7cvyfs6SxNtIinwyaV/ssVy+Be2ynA8DTY1xnkZKkjwVOEbShSR3h6Vtzxnz0Q4RMfML
xhxJ7we+2PF36tLuFR3jhJIbwskFN4RxX/4Z5cyva7od9ykSyuImO4zXuqQMO02McQoy
y5PSGk0lBSM43/Ybxn60w0E4uQfjQcdnb8dujUo/BA+SYm923BBglJFDRsFoZmjfI/kCrkl
Kt/N90izpC0ua4pM0SdJUSe+VtGUu2yO/qNQ5kweA7X/bPtH2HiT3mpmkpeNuMo/b/lm
2gN6MlOg26BMx8wvGhY7PHskhuSefvaZuCGOFpINtH5/PX2D7dz3InGr7dWqRbmdQ
kXQ8C/Lm7QA0ypsXLIykHzdxqZB0l+1njOWYholQfsG4MRqfveyGcGA+RnRD6HE836
d+Nmnbh1TINFoCLbdvs4Q6KEwkRT4INFVmku7ux/c+SITBSzDmlHz2dm3qs6eUbse2v

0paOuxX+KmzK8qeQQr43Dh9TQ114a0A8KLhrQaZJ23Pg7QkJ+mWUHzjSsxU+kgov2
A8+B3J4vGgXn32siXkx4DDSHvTkjTqdDtFbJ9S6G9jkhHKS/JYv1cjtrGkM7vcc89S0dakf
cu7S+UbAvc1HfNiZiIp8nFB9cGoBVRF3fka1UpOpGX/oE+E8gvGgx26+eyRImuUORzY
kZRu5/YstzHwTUlH2P5yPwYm6dnAh4FtSUlE3z6CSfmDpFlsr3yZlFPuzlK/a+W61zYb8
WJlIiny8aLbd+WPFWUzKsp6qQsaEnt+wbjQ1GdP0jWk1EIPlcrXAn7ZD0tJST8lKd6jST
E6F3LFcEWwaTVM1CrpBttb1tTNDxe2JCDpbJIiv65UPgX4mO0lSZEvcYSvX38JV4dgz
Mmm8OeQlnn2tf084NERnNVr0+1QsVzUku3z/98LXEGK8dg56t6yb68pr2O8sgeMB3Vp
gGYQSVcryT6unfP9SnWfrWh/aeG8HP1nXK2cJzqh/ILxoI3PXpsUPY2wPdn2RvnYuHC
+UZdwUp+T9LTOhaSDJJ0h6as5mHKZK5VyAC6EUvaAqyraDzITSZGPFwcWzstZL3av
aL9S4fw5pbolLSLQQBN7fsGYY3uvgs/eJ5TS4awm6fldfPa2lvRIRbno/iPcM12MEYBa/8
NvAy/P8h3jmHcA2wDTgX1L7Q8n5ab7DyqyB7Qc+uLiSklvc3UaoCVNkY8XqjmvuobuL4
WxR9VHQvkF44Lth4HjgOMKPnvHSKr02bPdL1eDbnQzRqjLmj2psBd4ADA9W42eImn
mIjex/wq8qJQ94Of9zh4wThzOxFHk44VrzquuIb0U7kNalVtN0utyuUjh/oI+EQYvwbiSDV
Y6e3cLpQ5aEsiO3tvYnqOUYX6a7Ys7dXXGLRMJjXEaoImEFiQMFmlpuBMMvC5h8Pe
73c/2W8ZinMNIKL9gzKny2SNZfPbNZ6/luJ4CrGP7T/l6PxbsXf0iz9rKMh8GXk2KbvIMY
Dvbzku5J9juGr80CKqQtLobJKqV9GbbJ4zlmCY6ofyCMUfSESSFMa3ss0fKdNAXn70W
45oOXFaI23krcC5JAc6x/fYauReQ8vP90va/c9nmwMp1cUqDoBujDZsXNCesPYPx4CB
gakfxAdi+DXhjrltcbE9KLdPhUdvvsP2fLFjWWwhJu9r+ne3TgLU75bZvIcz9g/Y0teQMy89
REsovGA/Gw2evDUuXwq29qXC+Wo1MMWnrKaW6j/RjUMFQ0nQJLpbsRkkov2A8GH
OfvZbMK/rs2b4BQNJ6FFIulWhquh4EY0F810ZJKL9gPNha0iMVx6PA4gzvdRRwlqSXS
FolHy8FTs91VTQ1XQ+CXhhRmUl6feHyt2M4lqEgDF6CoUbS7qRsDp1oGjcAn7d9bk37
fwIXk36sdsrn5OsX2159TAccLFFIWhGYbXt2vn4myfjrTtunFtqtURVLtnSvSGbbR0L5BU
ED8sywFtu/Ga+xBIOPpIuBQ2z/KbvD/B44EdgC+L3tcsizbveKZLZ9JJRfMLR0yZ0GgO1
3jiC/kMN+EJQpZu6Q9ClgDduHSloWuKpJVo+Y+fWXCG8WDDON86Nlh/2PkuJ5Cliq30
l2gwlF8eVqV/Jesu0nJS1iVCXpeqpfyEQKDB/0iVB+wdDSLUJGzj9YxeHAixnjJLvBhOE6
SUcD9wKbAr8EkLRaTfs9xmlcQ08sewZDjaQXAusBF9t+QNJWwAeAnar2V8YjyW4wcZ
C0AvAuUkSg42xfm8tfBGxiu5yzr+4+OwJvsH3omA12yIiZXzC0SDqK9KY9EzgyZyr/b+C
zwFtrxGod9iUtTof9YACxPYuU9qpcflleQq9F0jbAG4D9SUmUT+3WPmhGKL9gmHkNsK
3txyWtDtwHbNUJdF3DoDrsBwOIpEkk5bUeKY7tDZL2ILnXrABsW2q/OSnd11Tgb8CPS
St0u4zrwIeAWPYMhhZJV9l+XuF6pu1tRpDppKhZpIqKFDXBcCPpeGADkovDDsCdwAu
BD9g+vaL9POASknvErbnsNtsbj9eYh4WY+QXDzCaSzixcT87XAmx7z7LAOCXZDSYO
U0irCfMkLU9KhbWp7ftr2r+eNPO7UNJ5wMlEKLMxIWZ+wdBScFhfAdiMFM/zz8AsCIf1
YPSUUw/1mopI0krA3qTlz11J2UdOs/3LsRrrsBHKLxhasoHKZ0jGLXeR3rDXB44HPtQJ
SRUEbZH0GHBr5xLYpHCN7a16uMcawH7AAbZ3HYtxDiOh/IKhRdKXgZWBd9t+NJc9h
ZS26DHbhy/G4QUTAEmbkZzT7y5VbQjc19nXK7Rfo9TOwD8dP9R9J5RfMLRI+hOwef
mHJVvo/dH2ZotnZMFEIbvPfMj2daXyKcDHbL+2VH47SeEV9/lWIbnjHGL7zrEd8fAQBi/
BMOOqN2rbcyXFW2HQDyaXFR+A7RmSJleUb1R1E0mvA74N7N73EQ4pkc8vGGZuk
nRQuVDSG4E/LobxBBOP5bvUrdDrTXL6o7VHP5ygQ8z8gmHmUOBUSW8FriItN21P+l
HaZ3EOLJgwXCnpbba/UyyUdAjpO9cTklYmJit9Jfb8gqFH0q6kZLYCbrR9/mIeUjBBkLQ

```
OcBop+k9H2U0BlgX2Kfv7SXp3xW1WB/YEji0r0aA9ofyCIAjGGEm7AFvmyxttX1DT7mO
lIpPCnF1s+/oxHOLQEcovCIJgQJD0WdsfWtzjGAZiDTkIgmBwCGvOcSIMXoIgCAaHST
nDSGU8T9t/H+fxTFhi2TMIgmBAkPQEKet7lfJzZHfoHzHzC4IgGBxusr3tyM2C0RJ7fkE
QBMHQEcovCIJgcPiOpLXKhZLWzvkAgz4Ryi8IgmBw2AbYqaJ8N+DL4zuUiU0YvARBE
AwIkm6yvUVN3Y22nzPeY5qoxMwvCIJgcKh0ccjE73UfiQ8zCIJgcHhA0vPLhbnswcUwn
glLLHsGQRAMCFnJ/QQ4noUDYR8EHGj7isU0tAlHKL8gCIIBImeC+G9SIGwDNwLnAwf
YPnRxjm0iEcovCIJgAJG0LTAV2B+4HTjF9rGLd1QTh4jwEgRBMCBI2hw4kKT0/gb8mD
RJ2WWxDmwCEjO/IAiCAUHSPOAS4BDbt+ay2yKmZ/8Ja88gCILB4fXA/cCFkr4j6WV0
d38IWhIzvyAIggFD0krA3qTlz12BE4DTbP9ycY5rIhHKLwiCYICRtAawH8nac4fs
/IAiCYOiIPb8gCIJg6AjlFwRBEAwdofyCIAiCoSOUXxAEQTB0hPILgiAIho7/D2xrSxSSH9
2rAAAAAElFTkSuQmCC\n",
      "text/plain": [
       "<Figure size 432x288 with 2 Axes>"
      ]
     },
     "metadata": {
      "needs_background": "light"
     },
     "output_type": "display_data"
    }
   ],
   "source": [
    "sns.heatmap(data.corr())"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 23,
   "metadata": {
    "id": "DbN9x7Y3qcuD"
   },
   "outputs": [],
   "source": [

"new_data=data.drop(['ORIGIN_AIRPORT_ID','DEST_AIRPORT_ID','FL_NUM','YEAR','
CANCELLED','DIVERTED','DISTANCE','DAY_OF_MONTH','QUARTER','MONTH','DAY
_OF_WEEK','UNIQUE_CARRIER','TAIL_NUM'],axis=1)"
   ]
```

```
 },
 {
  "cell_type": "code",
  "execution_count": 24,
  "metadata": {
   "colab": {
    "base_uri": "https://localhost:8080/",
    "height": 206
   },
   "id": "cIGccKFI9SFR",
   "outputId": "93a575fc-28c2-4d7a-a51c-1aabab185171"
  },
  "outputs": [
   {
    "data": {
     "text/html": [
      "<div>\n",
      "<style scoped>\n",
      "    .dataframe tbody tr th:only-of-type {\n",
      "        vertical-align: middle;\n",
      "    }\n",
      "\n",
      "    .dataframe tbody tr th {\n",
      "        vertical-align: top;\n",
      "    }\n",
      "\n",
      "    .dataframe thead th {\n",
      "        text-align: right;\n",
      "    }\n",
      "</style>\n",
      "<table border=\"1\" class=\"dataframe\">\n",
      "  <thead>\n",
      "    <tr style=\"text-align: right;\">\n",
      "      <th></th>\n",
      "      <th>ORIGIN</th>\n",
      "      <th>DEST</th>\n",
      "      <th>CRS_DEP_TIME</th>\n",
      "      <th>DEP_TIME</th>\n",
    "     <th>DEP_DELAY</th>\n",
    "     <th>DEP_DEL15</th>\n",
    "     <th>CRS_ARR_TIME</th>\n",
    "     <th>ARR_TIME</th>\n",
    "     <th>ARR_DELAY</th>\n",
    "     <th>ARR_DEL15</th>\n",
    "     <th>CRS_ELAPSED_TIME</th>\n",
    "     <th>ACTUAL_ELAPSED_TIME</th>\n",
    "   </tr>\n",
    "  </thead>\n",
    "  <tbody>\n",
    "   <tr>\n",
    "     <th>0</th>\n",
    "     <td>ATL</td>\n",
    "     <td>SEA</td>\n",
    "     <td>1905</td>\n",
    "     <td>1907.0</td>\n",
    "     <td>-1.0</td>\n",
    "     <td>0.0</td>\n",
    "     <td>2143</td>\n",
    "     <td>2102.0</td>\n",
    "     <td>-41.0</td>\n",
    "     <td>0.0</td>\n",
    "     <td>338.0</td>\n",
    "     <td>295.0</td>\n",
    "   </tr>\n",
    "   <tr>\n",
    "     <th>1</th>\n",
    "     <td>DTW</td>\n",
    "    <td>MSP</td>\n",      "
<td>1345</td>\n",
    "     <td>1344.0</td>\n",
    "     <td>-1.0</td>\n",
    "     <td>0.0</td>\n",
    "     <td>1435</td>\n",
    "     <td>1439.0</td>\n",
    "     <td>-10.0</td>\n",
    "     <td>0.0</td>\n",
"      <td>110.0</td>\n",
"      <td>115.0</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>2</th>\n",
"      <td>ATL</td>\n",
"      <td>SEA</td>\n",
"      <td>940</td>\n",
"      <td>942.0</td>\n",
"      <td>-1.0</td>\n",
"      <td>0.0</td>\n",
"      <td>1215</td>\n",
"      <td>1142.0</td>\n",
"      <td>-33.0</td>\n",
"      <td>0.0</td>\n",
"      <td>335.0</td>\n",
"      <td>300.0</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>3</th>\n",
"      <td>SEA</td>\n",
"      <td>MSP</td>\n",
"      <td>819</td>\n",
"      <td>820.0</td>\n",
"      <td>-1.0</td>\n",
"      <td>0.0</td>\n",
"      <td>1335</td>\n",
"      <td>1345.0</td>\n",
"      <td>-10.0</td>\n",
"      <td>0.0</td>\n",
"      <td>196.0</td>\n",
"      <td>205.0</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>4</th>\n",
"      <td>SEA</td>\n",
"     <td>DTW</td>\n",      "
<td>2300</td>\n",
 "      <td>2256.0</td>\n",
 "      <td>-4.0</td>\n",
 "      <td>0.0</td>\n",
 "      <td>607</td>\n",
 "      <td>615.0</td>\n",
 "      <td>-10.0</td>\n",
 "      <td>0.0</td>\n",
 "      <td>247.0</td>\n",
 "      <td>259.0</td>\n",
 "    </tr>\n",
 "  </tbody>\n",
 "</table>\n",
 "</div>"
],
"text/plain": [
      "  ORIGIN DEST  CRS_DEP_TIME  DEP_TIME  DEP_DELAY  DEP_DEL15
CRS_ARR_TIME  \\\n",
      "0   ATL  SEA        1905    1907.0      -1.0       0.0        2143   \n",
      "1   DTW  MSP        1345    1344.0      -1.0       0.0        1435   \n",
      "2   ATL  SEA         940     942.0      -1.0       0.0        1215   \n",
      "3   SEA  MSP         819     820.0      -1.0       0.0        1335   \n",
      "4   SEA  DTW        2300    2256.0      -4.0       0.0         607   \n",      "\n",
      "   ARR_TIME  ARR_DELAY  ARR_DEL15  CRS_ELAPSED_TIME
ACTUAL_ELAPSED_TIME  \n",
      "0   2102.0     -41.0       0.0           338.0             295.0  \n",
      "1   1439.0     -10.0       0.0           110.0             115.0  \n",
      "2   1142.0     -33.0       0.0           335.0             300.0  \n",
      "3   1345.0     -10.0       0.0           196.0             205.0  \n",      "4
615.0     -10.0       0.0           247.0             259.0  "      ]
     },
     "execution_count": 24,
     "metadata": {},
     "output_type": "execute_result"
    }
   ],
   "source": [
    "new_data.head()"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {
    "id": "Z97RZytR-fOH"
   },
   "source": [
    "**Label Encoding**"
   "execution_count": 25,
   "metadata": {
   ]
  },
  {
   "cell_type": "code",
    "id": "kz83TDLW9Taq"
   },
   "outputs": [],
   "source": [
    "cities=new_data['ORIGIN'].unique()"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 26,
   "metadata": {
    "colab": {
     "base_uri": "https://localhost:8080/"
    },
    "id": "g4ku2MwD-nkr",
    "outputId": "8b891c9e-01a0-4670-a456-c936978c350e"
   },
   "outputs": [
    {
     "data": {
      "text/plain": [
       "array(['ATL', 'DTW', 'SEA', 'MSP', 'JFK'], dtype=object)"
      ]
     },
     "execution_count": 26,
     "metadata": {},
     "output_type": "execute_result"
    }
   ],
   "source": [
    "cities"
   "execution_count": 27,
   "metadata": {
   ]
  },
  {
   "cell_type": "code",
   "colab": {
    "base_uri": "https://localhost:8080/"
   },
   "id": "y9z7hnqf9-wP",
   "outputId": "918a1582-b37b-45a7-cdf2-f2149ce128b9"
  },
  "outputs": [
   {
    "data": {
     "text/plain": [
      "array(['SEA', 'MSP', 'DTW', 'ATL', 'JFK'], dtype=object)"
     ]
    },
    "execution_count": 27,
    "metadata": {},
    "output_type": "execute_result"
   }
  ],
  "source": [
   "new_data['DEST'].unique()"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 28,
  "metadata": {
   "id": "AsadBJsE-aVg"
  },
  "outputs": [],
  "source": [
   "city_map={cities[i]:i for i in range(0,len(cities))}"


  ]
 },
 {
  "cell_type": "code",
   "execution_count": 29,
   "metadata": {
    "colab": {
     "base_uri": "https://localhost:8080/"
    },
    "id": "ZCuN152b-xes",
    "outputId": "5127e75a-30f7-4bb9-d96c-566a579e0186"
   },
   "outputs": [
    {
     "data": {
      "text/plain": [
       "{'ATL': 0, 'DTW': 1, 'SEA': 2, 'MSP': 3, 'JFK': 4}"
      ]
     },
     "execution_count": 29,
     "metadata": {},
     "output_type": "execute_result"
    }
   ],
   "source": [
    "city_map"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 30,
   "metadata": {
    "id": "SYJp2IdL-yzX"
   },
   "outputs": [],
   "source": [
    "def encode(c):\n",
    "  return city_map[c]"
   ]
  },
{
"cel
```

```
l_ty
pe":
"co
de",
  "execution_count": 31,
  "metadata": {
   "id": "TylfsmCe_Dcm"
  },
  "outputs": [],
  "source": [
   "new_data['ORIGIN']=new_data['ORIGIN'].apply(encode)"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 32,
  "metadata": {
   "id": "WdpEDHLX_KUM"
  },
  "outputs": [],
  "source": [
   "new_data['DEST']=new_data['DEST'].apply(encode)"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 33,
  "metadata": {
   "colab": {
    "base_uri": "https://localhost:8080/",
    "height": 206
   },
   "id": "aD7rcYOA_NWJ",
   "outputId": "eca5abd0-68fb-4517-aa77-8cd562d69e5e"
  },
  "outputs": [
   {
    "data": {
```

```
"text/html": [
 "<div>\n",
 "<style scoped>\n",
 "    .dataframe tbody tr th:only-of-type {\n",
 "        vertical-align: middle;\n",
 "    }\n",
 "\n",
 "    .dataframe tbody tr th {\n",
 "        vertical-align: top;\n",
 "    }\n",
 "\n",
 "    .dataframe thead th {\n",
 "        text-align: right;\n",
 "    }\n",
 "</style>\n",
 "<table border=\"1\" class=\"dataframe\">\n",
 "  <thead>\n",
 "    <tr style=\"text-align: right;\">\n",
 "      <th></th>\n",
 "      <th>ORIGIN</th>\n",
 "      <th>DEST</th>\n",
 "      <th>CRS_DEP_TIME</th>\n",
 "      <th>DEP_TIME</th>\n",
 "      <th>DEP_DELAY</th>\n",
 "      <th>DEP_DEL15</th>\n",
 "      <th>CRS_ARR_TIME</th>\n",
 "      <th>ARR_TIME</th>\n",
 "      <th>ARR_DELAY</th>\n",
 "      <th>ARR_DEL15</th>\n",
 "      <th>CRS_ELAPSED_TIME</th>\n",
 "      <th>ACTUAL_ELAPSED_TIME</th>\n",
 "    </tr>\n",
 "  </thead>\n",
 "  <tbody>\n",
 "    <tr>\n",
 "      <th>0</th>\n",
 "      <td>0</td>\n",
 "      <td>2</td>\n",
"      <td>1905</td>\n",
"      <td>1907.0</td>\n",
"      <td>-1.0</td>\n",
"      <td>0.0</td>\n",
"      <td>2143</td>\n",
"      <td>2102.0</td>\n",
"      <td>-41.0</td>\n",
"      <td>0.0</td>\n",
"      <td>338.0</td>\n",
"      <td>295.0</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>1</th>\n",
"      <td>1</td>\n",
"      <td>3</td>\n",
"      <td>1345</td>\n",
"      <td>1344.0</td>\n",
"      <td>-1.0</td>\n",
"      <td>0.0</td>\n",
"      <td>1435</td>\n",
"      <td>1439.0</td>\n",
"      <td>-10.0</td>\n",
"      <td>0.0</td>\n",
"      <td>110.0</td>\n",
"      <td>115.0</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>2</th>\n",
"      <td>0</td>\n",
"      <td>2</td>\n",
"      <td>940</td>\n",
"      <td>942.0</td>\n",
"      <td>-1.0</td>\n",
"      <td>0.0</td>\n",
"      <td>1215</td>\n",
"      <td>1142.0</td>\n",
"      <td>-33.0</td>\n",
"      <td>0.0</td>\n",
 "      <td>335.0</td>\n",
 "      <td>300.0</td>\n",
 "    </tr>\n",
 "    <tr>\n",
 "      <th>3</th>\n",
 "      <td>2</td>\n",
 "      <td>3</td>\n",
 "      <td>819</td>\n",
 "      <td>820.0</td>\n",
 "      <td>-1.0</td>\n",
 "      <td>0.0</td>\n",
 "      <td>1335</td>\n",
 "      <td>1345.0</td>\n",
 "      <td>-10.0</td>\n",
 "      <td>0.0</td>\n",
 "      <td>196.0</td>\n",
 "      <td>205.0</td>\n",
 "    </tr>\n",
 "    <tr>\n",
 "      <th>4</th>\n",
 "      <td>2</td>\n",
 "      <td>1</td>\n",
 "      <td>2300</td>\n",
 "      <td>2256.0</td>\n",
 "      <td>-4.0</td>\n",
 "      <td>0.0</td>\n",
 "      <td>607</td>\n",
 "      <td>615.0</td>\n",
 "      <td>-10.0</td>\n",
 "      <td>0.0</td>\n",
 "      <td>247.0</td>\n",
 "      <td>259.0</td>\n",
 "    </tr>\n",
 "  </tbody>\n",
 "</table>\n",
 "</div>"
],
"text/plain": [
      "   ORIGIN  DEST  CRS_DEP_TIME  DEP_TIME  DEP_DELAY  DEP_DEL15  
CRS_ARR_TIME  \\\n",
      "0      0    2        1905    1907.0      -1.0       0.0        2143   \n",
      "1      1    3        1345    1344.0      -1.0       0.0        1435   \n",
      "2      0    2         940     942.0      -1.0       0.0        1215   \n",
      "3      2    3         819     820.0      -1.0       0.0        1335   \n",
      "4      2    1        2300    2256.0      -4.0       0.0         607   \n",      "\n",
      "   ARR_TIME  ARR_DELAY  ARR_DEL15  CRS_ELAPSED_TIME  
ACTUAL_ELAPSED_TIME  \n",
      "0   2102.0     -41.0       0.0           338.0              295.0  \n",
      "1   1439.0     -10.0       0.0           110.0              115.0  \n",
      "2   1142.0     -33.0       0.0           335.0              300.0  \n",
      "3   1345.0     -10.0       0.0           196.0              205.0  \n",      "4 
615.0     -10.0       0.0           247.0              259.0  "      ]
     },
     "execution_count": 33,
     "metadata": {},
     "output_type": "execute_result"
    }
   ],
   "source": [
    "new_data.head()"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 34,
   "metadata": {
    "colab": {
     "base_uri": "https://localhost:8080/"
    },
    "id": "CiklsaWl_Qc8",
    "outputId": "fa1af054-a33e-4634-929c-484a57078670"
   },
   "outputs": [
    {
     "data": {
      "text/plain": [
     "ORIGIN                 0.034813\n",
     "DEST                   0.039938\n",
     "CRS_DEP_TIME           0.086057\n",
     "DEP_TIME               0.127593\n",
     "DEP_DELAY              0.200721\n",
     "DEP_DEL15              0.658511\n",
     "CRS_ARR_TIME           0.078282\n",
     "ARR_TIME               0.042298\n",
     "ARR_DELAY              0.183476\n",
     "ARR_DEL15              1.000000\n",
     "CRS_ELAPSED_TIME       0.015676\n",
     "ACTUAL_ELAPSED_TIME    0.077741\n",
     "Name: ARR_DEL15, dtype: float64"
    ]
   },
   "execution_count": 34,
   "metadata": {},
   "output_type": "execute_result"
  }
 ],
 "source": [
  "new_data.corr()['ARR_DEL15']"
 ]
},
{
 "cell_type": "code",
 "execution_count": 35,
 "metadata": {
  "colab": {
   "base_uri": "https://localhost:8080/"
  },
  "id": "mOsfqqzGAzeK",
  "outputId": "9326f074-b2f0-4262-e59a-ad8a8b1dc449"    },
 "outputs": [
  {
   "data": {
    "text/plain": [
     "YEAR                   0\n",
     "QUARTER                0\n",
     "MONTH                  0\n",
     "DAY_OF_MONTH           0\n",
     "DAY_OF_WEEK            0\n",
     "UNIQUE_CARRIER         0\n",
     "TAIL_NUM               0\n",
     "FL_NUM                 0\n",
     "ORIGIN_AIRPORT_ID      0\n",
     "ORIGIN                 0\n",
     "DEST_AIRPORT_ID        0\n",
     "DEST                   0\n",
     "CRS_DEP_TIME           0\n",
     "DEP_TIME               0\n",
     "DEP_DELAY              0\n",
     "DEP_DEL15              0\n",
     "CRS_ARR_TIME           0\n",
     "ARR_TIME               0\n",
     "ARR_DELAY              0\n",
     "ARR_DEL15              0\n",
     "CANCELLED              0\n",
     "DIVERTED               0\n",
     "CRS_ELAPSED_TIME       0\n",
     "ACTUAL_ELAPSED_TIME    0\n",
     "DISTANCE               0\n",
     "dtype: int64"
    ]
   },
   "execution_count": 35,
   "metadata": {},
   "output_type": "execute_result"
  }
 ],
 "source": [
  "\n",
  "data.isnull().sum()"
 ]
},
{
   "cell_type": "code",
   "execution_count": 36,
   "metadata": {
    "colab": {
     "base_uri": "https://localhost:8080/"
    },
    "id": "ZO_3AXS7CJd6",
    "outputId": "0f06c49f-e09f-45bc-9a8a-7a08f6dbedde"
   },
   "outputs": [
    {
     "data": {
      "text/plain": [
       "FL_NUM          0\n",
       "MONTH           0\n",
       "DAY_OF_MONTH    0\n",
       "DAY_OF_WEEK     0\n",
       "ORIGIN          0\n",
       "DEST            0\n",
       "CRS_ARR_TIME    0\n",
       "DEP_DEL15       0\n",
       "ARR_DEL15       0\n",
       "dtype: int64"
      ]
     },
     "execution_count": 36,
     "metadata": {},
     "output_type": "execute_result"
    }
   ],
   "source": [
"data=data[[\"FL_NUM\",\"MONTH\",\"DAY_OF_MONTH\",\"DAY_OF_WEEK\",\"ORIGIN
\",\"DEST\",\"CRS_ARR_TIME\",\"DEP_DEL15\",\"ARR_DEL15\"]]\n",
    "data.isnull().sum()"
   ]
  },
  {
```

```
 "cell_type": "markdown",
 "metadata": {
  "id": "HAxUNHKUBcXw"
 },
 "source": []
},
{
 "cell_type": "code",
 "execution_count": 37,
 "metadata": {
  "colab": {
   "base_uri": "https://localhost:8080/",
   "height": 300
  },
  "id": "meJwlbX9DdAj",
  "outputId": "a3dde882-8e97-4980-ea2c-0984cc07ce5e"
 },
 "outputs": [
  {
   "data": {
    "text/html": [
     "<div>\n",
     "<style scoped>\n",
     "    .dataframe tbody tr th:only-of-type {\n",
     "        vertical-align: middle;\n",
     "    }\n",
     "\n",
     "    .dataframe tbody tr th {\n",
     "        vertical-align: top;\n",
     "    }\n",
     "\n",
     "    .dataframe thead th {\n",
     "        text-align: right;\n",
     "    }\n",
     "</style>\n",
     "<table border=\"1\" class=\"dataframe\">\n",
     "  <thead>\n",
     "    <tr style=\"text-align: right;\">\n",
    "      <th></th>\n",
    "      <th>FL_NUM</th>\n",
    "      <th>MONTH</th>\n",
    "      <th>DAY_OF_MONTH</th>\n",
    "      <th>DAY_OF_WEEK</th>\n",
    "      <th>ORIGIN</th>\n",
    "      <th>DEST</th>\n",
    "      <th>CRS_ARR_TIME</th>\n",
    "      <th>DEP_DEL15</th>\n",
    "      <th>ARR_DEL15</th>\n",
    "    </tr>\n",
    "  </thead>\n",
    "  <tbody>\n",
    "    <tr>\n",
    "      <th>178</th>\n",
    "      <td>2839</td>\n",
    "      <td>1</td>\n",
    "      <td>9</td>\n",
    "      <td>6</td>\n",
    "      <td>DTW</td>\n",
    "      <td>JFK</td>\n",
    "      <td>1724</td>\n",
    "      <td>0.0</td>\n",
    "      <td>0.0</td>\n",
    "    </tr>\n",
    "    <tr>\n",
    "      <th>180</th>\n",
    "      <td>87</td>\n",
    "      <td>1</td>\n",
    "      <td>10</td>\n",
    "      <td>7</td>\n",
    "      <td>DTW</td>\n",
    "     <td>MSP</td>\n",      "
<td>1649</td>\n",
    "      <td>1.0</td>\n",
    "      <td>0.0</td>\n",
    "    </tr>\n",
    "    <tr>\n",
    "      <th>181</th>\n",
    "      <td>423</td>\n",
    "      <td>1</td>\n",
    "      <td>10</td>\n",
    "      <td>7</td>\n",
    "     <td>JFK</td>\n",       "
<td>ATL</td>\n",
    "      <td>1600</td>\n",
    "      <td>0.0</td>\n",
    "      <td>0.0</td>\n",
    "    </tr>\n",
    "    <tr>\n",
    "      <th>182</th>\n",
    "      <td>440</td>\n",
    "      <td>1</td>\n",
    "      <td>10</td>\n",
    "      <td>7</td>\n",
    "     <td>JFK</td>\n",       "
<td>ATL</td>\n",
    "      <td>849</td>\n",
    "      <td>0.0</td>\n",
    "      <td>0.0</td>\n",
    "    </tr>\n",
    "    <tr>\n",
    "      <th>183</th>\n",
    "      <td>485</td>\n",
    "      <td>1</td>\n",
    "      <td>10</td>\n",
    "      <td>7</td>\n",
    "      <td>JFK</td>\n",
    "      <td>SEA</td>\n",
    "      <td>1945</td>\n",
    "      <td>1.0</td>\n",
    "      <td>0.0</td>\n",
    "    </tr>\n",
    "    <tr>\n",
    "      <th>185</th>\n",
    "      <td>589</td>\n",
 "      <td>1</td>\n",
 "      <td>10</td>\n",
 "      <td>7</td>\n",
 "      <td>MSP</td>\n",
 "      <td>SEA</td>\n",
 "      <td>1100</td>\n",
 "      <td>0.0</td>\n",
 "      <td>0.0</td>\n",
 "    </tr>\n",
 "    <tr>\n",
 "      <th>186</th>\n",
 "      <td>744</td>\n",
 "      <td>1</td>\n",
 "      <td>10</td>\n",
 "      <td>7</td>\n",
 "      <td>MSP</td>\n",
 "      <td>ATL</td>\n",
 "      <td>1334</td>\n",
 "      <td>0.0</td>\n",
 "      <td>0.0</td>\n",
 "    </tr>\n",
 "    <tr>\n",
 "      <th>187</th>\n",
 "      <td>789</td>\n",
 "      <td>1</td>\n",
 "      <td>10</td>\n",
 "      <td>7</td>\n",
 "      <td>SEA</td>\n",
 "      <td>MSP</td>\n",
 "      <td>1837</td>\n",
 "      <td>0.0</td>\n",
 "      <td>0.0</td>\n",
 "    </tr>\n",
 "  </tbody>\n",
 "</table>\n",
 "</div>"
],
"text/plain": [
      "    FL_NUM  MONTH  DAY_OF_MONTH  DAY_OF_WEEK ORIGIN DEST
CRS_ARR_TIME  \\\n",
      "178   2839     1          9          6   DTW  JFK       1724   \n",
      "180     87     1         10          7   DTW  MSP       1649   \n",
      "181    423     1         10          7   JFK  ATL       1600   \n",
      "182    440     1         10          7   JFK  ATL        849   \n",
      "183    485     1         10          7   JFK  SEA       1945   \n",
      "185    589     1         10          7   MSP  SEA       1100   \n",
      "186    744     1         10          7   MSP  ATL       1334   \n",
      "187    789     1         10          7   SEA  MSP       1837   \n",      "\n",
      "    DEP_DEL15  ARR_DEL15  \n",
      "178       0.0       0.0  \n",
      "180       1.0       0.0  \n",
      "181       0.0       0.0  \n",
      "182       0.0       0.0  \n",
      "183       1.0       0.0  \n",
      "185       0.0       0.0  \n",
      "186       0.0       0.0  \n",
      "187       0.0       0.0  "
     ]
    },
    "execution_count": 37,
    "metadata": {},
    "output_type": "execute_result"
   }
  ],
  "source": [
   "data=data.fillna({'ARR_DEL15': 1})\n",
   "data=data.fillna({'DEP_DEL15': 0})\n",
   "data.iloc[177:185]"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 38,
  "metadata": {
   "colab": {
    "base_uri": "https://localhost:8080/",
  "height": 206
 },
 "id": "keObS1THECZK",
 "outputId": "92645709-278e-41a0-c408-77d8d1ff9f48"
},
"outputs": [
 {
  "data": {
   "text/html": [
    "<div>\n",
    "<style scoped>\n",
    "    .dataframe tbody tr th:only-of-type {\n",
    "        vertical-align: middle;\n",
    "    }\n",
    "\n",
    "    .dataframe tbody tr th {\n",
    "        vertical-align: top;\n",
    "    }\n",
    "\n",
    "    .dataframe thead th {\n",
    "        text-align: right;\n",
    "    }\n",
    "</style>\n",
    "<table border=\"1\" class=\"dataframe\">\n",
    "  <thead>\n",
    "    <tr style=\"text-align: right;\">\n",
    "      <th></th>\n",
    "      <th>FL_NUM</th>\n",
    "      <th>MONTH</th>\n",
    "      <th>DAY_OF_MONTH</th>\n",
    "      <th>DAY_OF_WEEK</th>\n",
    "      <th>ORIGIN</th>\n",
    "      <th>DEST</th>\n",
    "      <th>CRS_ARR_TIME</th>\n",
    "      <th>DEP_DEL15</th>\n",
    "      <th>ARR_DEL15</th>\n",
    "    </tr>\n",
    "  </thead>\n",
"  <tbody>\n",
"    <tr>\n",
"      <th>0</th>\n",
"      <td>1399</td>\n",
"      <td>1</td>\n",
"      <td>1</td>\n",
"      <td>5</td>\n",
"      <td>ATL</td>\n",
"      <td>SEA</td>\n",
"      <td>21</td>\n",
"      <td>0.0</td>\n",
"      <td>0.0</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>1</th>\n",
"      <td>1476</td>\n",
"      <td>1</td>\n",
"      <td>1</td>\n",
"      <td>5</td>\n",
"      <td>DTW</td>\n",
"      <td>MSP</td>\n",
"      <td>14</td>\n",
"      <td>0.0</td>\n",
"      <td>0.0</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>2</th>\n",
"      <td>1597</td>\n",
"      <td>1</td>\n",
"      <td>1</td>\n",
"      <td>5</td>\n",
"      <td>ATL</td>\n",
"      <td>SEA</td>\n",
"      <td>12</td>\n",
"      <td>0.0</td>\n",
"      <td>0.0</td>\n",
"    </tr>\n",
"    <tr>\n",
    "     <th>3</th>\n",
    "     <td>1768</td>\n",
    "     <td>1</td>\n",
    "     <td>1</td>\n",
    "     <td>5</td>\n",
    "     <td>SEA</td>\n",
    "     <td>MSP</td>\n",
    "     <td>13</td>\n",
    "     <td>0.0</td>\n",
    "     <td>0.0</td>\n",
    "    </tr>\n",
    "    <tr>\n",
    "     <th>4</th>\n",
    "     <td>1823</td>\n",
    "     <td>1</td>\n",
    "     <td>1</td>\n",
    "     <td>5</td>\n",
    "     <td>SEA</td>\n",
    "     <td>DTW</td>\n",
    "     <td>6</td>\n",
    "     <td>0.0</td>\n",
    "     <td>0.0</td>\n",
    "    </tr>\n",
    "  </tbody>\n",
    "</table>\n",
    "</div>"
   ],
   "text/plain": [
    "   FL_NUM  MONTH  DAY_OF_MONTH  DAY_OF_WEEK ORIGIN DEST  
CRS_ARR_TIME  \\\n",
    "0    1399     1            1            5    ATL  SEA           21   \n",
    "1    1476     1            1            5    DTW  MSP           14   \n",
    "2    1597     1            1            5    ATL  SEA           12   \n",
    "3    1768     1            1            5    SEA  MSP           13   \n",       "4
1823     1            1            5    SEA  DTW            6   \n",       "\n",
    "   DEP_DEL15  ARR_DEL15  \n",
    "0        0.0        0.0  \n",
    "1        0.0        0.0  \n",
      "2       0.0       0.0  \n",
      "3       0.0       0.0  \n",
      "4       0.0       0.0  "
     ]
    },
    "execution_count": 38,
    "metadata": {},
    "output_type": "execute_result"
   }
  ],
  "source": [
   "import math\n",
   "\n",
   "for index, row in data.iterrows():\n",
   "        data.loc[index,'CRS_ARR_TIME']     =     math.floor(row['CRS_ARR_TIME']     /     100)\n",
"data.head()"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 39,
  "metadata": {
   "id": "0zqvvjJvE7q8"
  },
  "outputs": [],
  "source": [
   "from sklearn.preprocessing import  LabelEncoder\n",
   "le = LabelEncoder()\n",
   "data['DEST'] = le.fit_transform(data['DEST'])\n",
   "data['ORIGIN'] = le.fit_transform(data['ORIGIN'])"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 40,
  "metadata": {
   "colab": {
    "base_uri": "https://localhost:8080/",
     "height": 206
    },
    "id": "XZOglhp8HfYI",
    "outputId": "197bbb64-a553-4f0d-c20f-a41af343c96d"
   },
   "outputs": [
    {
     "data": {
      "text/html": [
       "<div>\n",
       "<style scoped>\n",
       "    .dataframe tbody tr th:only-of-type {\n",
       "        vertical-align: middle;\n",
       "    }\n",
       "\n",
       "    .dataframe tbody tr th {\n",
       "        vertical-align: top;\n",
       "    }\n",
       "\n",
       "    .dataframe thead th {\n",
       "        text-align: right;\n",
       "    }\n",
       "</style>\n",
       "<table border=\"1\" class=\"dataframe\">\n",
       "  <thead>\n",
       "    <tr style=\"text-align: right;\">\n",
       "      <th></th>\n",
       "      <th>FL_NUM</th>\n",
       "      <th>MONTH</th>\n",
       "      <th>DAY_OF_MONTH</th>\n",
       "      <th>DAY_OF_WEEK</th>\n",
       "      <th>ORIGIN</th>\n",
       "      <th>DEST</th>\n",
       "      <th>CRS_ARR_TIME</th>\n",
       "      <th>DEP_DEL15</th>\n",
       "      <th>ARR_DEL15</th>\n",
       "    </tr>\n",
       "  </thead>\n",
"  <tbody>\n",
"    <tr>\n",
"      <th>0</th>\n",
"      <td>1399</td>\n",
"      <td>1</td>\n",
"      <td>1</td>\n",
"      <td>5</td>\n",
"      <td>0</td>\n",
"      <td>4</td>\n",
"      <td>21</td>\n",
"      <td>0.0</td>\n",
"      <td>0.0</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>1</th>\n",
"      <td>1476</td>\n",
"      <td>1</td>\n",
"      <td>1</td>\n",
"      <td>5</td>\n",
"      <td>1</td>\n",
"      <td>3</td>\n",
"      <td>14</td>\n",
"      <td>0.0</td>\n",
"      <td>0.0</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>2</th>\n",
"      <td>1597</td>\n",
"      <td>1</td>\n",
"      <td>1</td>\n",
"      <td>5</td>\n",
"      <td>0</td>\n",
"      <td>4</td>\n",
"      <td>12</td>\n",
"      <td>0.0</td>\n",
"      <td>0.0</td>\n",
"    </tr>\n",
"    <tr>\n",
   "     <th>3</th>\n",
   "     <td>1768</td>\n",
   "     <td>1</td>\n",
   "     <td>1</td>\n",
   "     <td>5</td>\n",
   "     <td>4</td>\n",
   "     <td>3</td>\n",
   "     <td>13</td>\n",
   "     <td>0.0</td>\n",
   "     <td>0.0</td>\n",
   "   </tr>\n",
   "   <tr>\n",
   "     <th>4</th>\n",
   "     <td>1823</td>\n",
   "     <td>1</td>\n",
   "     <td>1</td>\n",
   "     <td>5</td>\n",
   "     <td>4</td>\n",
   "     <td>1</td>\n",
   "     <td>6</td>\n",
   "     <td>0.0</td>\n",
   "     <td>0.0</td>\n",
   "   </tr>\n",
   "  </tbody>\n",
   "</table>\n",
   "</div>"
  ],
  "text/plain": [
   "   FL_NUM  MONTH  DAY_OF_MONTH  DAY_OF_WEEK  ORIGIN  DEST
CRS_ARR_TIME  \\\n",
   "0    1399      1             1            5       0     4            21   \n",
   "1    1476      1             1            5       1     3            14   \n",
   "2    1597      1             1            5       0     4            12   \n",
   "3    1768      1             1            5       4     3            13   \n",      "4    1823
1             1            5       4     1             6   \n",
   "\n",
   "   DEP_DEL15  ARR_DEL15  \n",
   "0        0.0        0.0  \n",
      "1       0.0       0.0  \n",
      "2       0.0       0.0  \n",
      "3       0.0       0.0  \n",
      "4       0.0       0.0  "
     ]
    },
    "execution_count": 40,
    "metadata": {},
    "output_type": "execute_result"
   }
  ],
  "source": [
   "data.head()"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 41,
  "metadata": {
   "id": "Tl4kcv9zHrML"
  },
  "outputs": [],
  "source": [
   "from sklearn.preprocessing import OneHotEncoder\n",
   "oh = OneHotEncoder()\n",
   "z=oh.fit_transform(data['ORIGIN'].values.reshape(-1,1)).toarray()\n",
   "t=oh.fit_transform(data['DEST'].values.reshape(-1,1)).toarray()"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 42,
  "metadata": {
   "id": "j4LsWPEWZ4d4"
  },
  "outputs": [],
  "source": [
   "data=pd.get_dummies(data,columns=['ORIGIN','DEST'])"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 43,
   "metadata": {
    "colab": {
     "base_uri": "https://localhost:8080/"
    },
    "id": "yKkTy08oZ4d4",
    "outputId": "fd8a7d47-4cac-4bd8-8066-89279d031d25"
   },
   "outputs": [
    {
     "data": {
      "text/plain": [
       "0.0    9668\n",
       "1.0    1375\n",
       "Name: ARR_DEL15, dtype: int64"
      ]
     },
     "execution_count": 43,
     "metadata": {},
     "output_type": "execute_result"
    }
   ],
   "source": [
    "data['ARR_DEL15'].value_counts()"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 44,
   "metadata": {
    "colab": {
     "base_uri": "https://localhost:8080/",
     "height": 250
    },
 "id": "5e-1t6beZ4d5",
 "outputId": "4a94d1f6-9e6d-48a4-85c8-5523d02354a7"
},
"outputs": [
 {
  "data": {
   "text/html": [
    "<div>\n",
    "<style scoped>\n",
    "    .dataframe tbody tr th:only-of-type {\n",
    "        vertical-align: middle;\n",
    "    }\n",
    "\n",
    "    .dataframe tbody tr th {\n",
    "        vertical-align: top;\n",
    "    }\n",
    "\n",
    "    .dataframe thead th {\n",
    "        text-align: right;\n",
    "    }\n",
    "</style>\n",
    "<table border=\"1\" class=\"dataframe\">\n",
    "  <thead>\n",
    "    <tr style=\"text-align: right;\">\n",
    "      <th></th>\n",
    "      <th>FL_NUM</th>\n",
    "      <th>MONTH</th>\n",
    "      <th>DAY_OF_MONTH</th>\n",
    "      <th>DAY_OF_WEEK</th>\n",
    "      <th>CRS_ARR_TIME</th>\n",
    "      <th>DEP_DEL15</th>\n",
    "      <th>ARR_DEL15</th>\n",
    "      <th>ORIGIN_0</th>\n",
    "      <th>ORIGIN_1</th>\n",
    "      <th>ORIGIN_2</th>\n",
    "      <th>ORIGIN_3</th>\n",
    "      <th>ORIGIN_4</th>\n",
    "      <th>DEST_0</th>\n",
"      <th>DEST_1</th>\n",
"      <th>DEST_2</th>\n",
"      <th>DEST_3</th>\n",
"      <th>DEST_4</th>\n",
"    </tr>\n",
"  </thead>\n",
"  <tbody>\n",
"    <tr>\n",
"      <th>11226</th>\n",
"      <td>1715</td>\n",
"      <td>12</td>\n",
"      <td>30</td>\n",
"      <td>5</td>\n",
"      <td>12</td>\n",
"      <td>0.0</td>\n",
"      <td>0.0</td>\n",
"      <td>0</td>\n",
"      <td>1</td>\n",
"      <td>0</td>\n",
"      <td>0</td>\n",
"      <td>0</td>\n",
"      <td>1</td>\n",
"      <td>0</td>\n",
"      <td>0</td>\n",
"      <td>0</td>\n",
"      <td>0</td>\n",
"    </tr>\n",
"    <tr>\n",
"      <th>11227</th>\n",
"      <td>1770</td>\n",
"      <td>12</td>\n",
"      <td>30</td>\n",
"      <td>5</td>\n",
"      <td>20</td>\n",
"      <td>1.0</td>\n",
"      <td>0.0</td>\n",
"      <td>0</td>\n",
"      <td>0</td>\n",
"     <td>0</td>\n",
"     <td>0</td>\n",
"     <td>1</td>\n",
"     <td>0</td>\n",
"     <td>0</td>\n",
"     <td>0</td>\n",
"     <td>1</td>\n",
"     <td>0</td>\n",
"   </tr>\n",
"   <tr>\n",
"     <th>11228</th>\n",        "      <td>1823</td>\n",
"     <td>12</td>\n",
"     <td>30</td>\n",
"     <td>5</td>\n",
"     <td>22</td>\n",
"     <td>0.0</td>\n",
"     <td>0.0</td>\n",
"     <td>0</td>\n",
"     <td>1</td>\n",
"     <td>0</td>\n",
"     <td>0</td>\n",
"     <td>0</td>\n",
"     <td>0</td>\n",
"     <td>0</td>\n",
"     <td>0</td>\n",
"     <td>0</td>\n",
"     <td>1</td>\n",
"   </tr>\n",
"   <tr>\n",
"     <th>11229</th>\n",
"     <td>1901</td>\n",
"     <td>12</td>\n",
"     <td>30</td>\n",
"     <td>5</td>\n",
"     <td>18</td>\n",
"     <td>0.0</td>\n",
"     <td>0.0</td>\n",
"     <td>1</td>\n",
      "    <td>0</td>\n",
      "    <td>0</td>\n",
      "    <td>0</td>\n",
      "    <td>0</td>\n",
      "    <td>0</td>\n",
      "    <td>0</td>\n",
      "    <td>0</td>\n",
      "    <td>0</td>\n",
      "    <td>1</td>\n",
      "  </tr>\n",
      "  <tr>\n",
      "    <th>11230</th>\n",
      "    <td>2005</td>\n",
      "    <td>12</td>\n",
      "    <td>30</td>\n",
      "    <td>5</td>\n",
      "    <td>9</td>\n",
      "    <td>0.0</td>\n",
      "    <td>0.0</td>\n",
      "    <td>1</td>\n",
      "    <td>0</td>\n",
      "    <td>0</td>\n",
      "    <td>0</td>\n",
      "    <td>0</td>\n",
      "    <td>0</td>\n",
      "    <td>1</td>\n",
      "    <td>0</td>\n",
      "    <td>0</td>\n",
      "    <td>0</td>\n",
      "  </tr>\n",
      " </tbody>\n",
      "</table>\n",
      "</div>"
     ],
     "text/plain": [
      "       FL_NUM  MONTH  DAY_OF_MONTH  DAY_OF_WEEK  CRS_ARR_TIME  
DEP_DEL15  \\\n",
      "11226    1715     12            30            5            12       0.0   \n",
     "11227   1770    12          30          5          20       1.0   \n",
     "11228   1823    12          30          5          22       0.0   \n",
     "11229   1901    12          30          5          18       0.0   \n",       "11230
2005     12          30          5          9       0.0   \n",       "\n",
     "      ARR_DEL15  ORIGIN_0  ORIGIN_1  ORIGIN_2  ORIGIN_3  ORIGIN_4
DEST_0  \\\n",
     "11226       0.0        0        1        0        0        0        1   \n",
     "11227       0.0        0        0        0        0        1        0   \n",
     "11228       0.0        0        1        0        0        0        0   \n",
     "11229       0.0        1        0        0        0        0        0   \n",
     "11230       0.0        1        0        0        0        0        0   \n",
     "\n",
     "      DEST_1  DEST_2  DEST_3  DEST_4  \n",
     "11226      0      0      0      0  \n",
     "11227      0      0      1      0  \n",
     "11228      0      0      0      1  \n",
     "11229      0      0      0      1  \n",
     "11230      1      0      0      0  "
    ]
   },
   "execution_count": 44,
   "metadata": {},
   "output_type": "execute_result"
  }
 ],
 "source": [
  "data.tail()"
 ]
},
{
 "cell_type": "markdown",
 "metadata": {
  "id": "CmmseFn9pu30"
 },
 "source": [
  "**Split the data into dependent and independent variables**\n"
 ]
},
{
 "cell_type": "code",
 "execution_count": 45,
 "metadata": {
  "id": "ejZIG0o8_V1x"
 },
 "outputs": [],
 "source": [
  "x=data[[i for i in data.columns if i!='ARR_DEL15']].values\n",
  "y=data[[i for i in data.columns if i=='ARR_DEL15']].values"
 ]
},
{
 "cell_type": "code",
 "execution_count": 46,
 "metadata": {
  "colab": {
   "base_uri": "https://localhost:8080/"
  },
  "id": "f3EN9n6BEAzu",
  "outputId": "8fb2616a-d045-48f6-84d5-39a16ef15d7d"
 },
 "outputs": [
  {
   "data": {
    "text/plain": [
     "(11043, 16)"
    ]
   },
   "execution_count": 46,
   "metadata": {},
   "output_type": "execute_result"
  }
 ],
 "source": [
  "x.shape"
 ]
},
{
 "cell_type": "code",
 "execution_count": 47,
 "metadata": {
  "colab": {
   "base_uri": "https://localhost:8080/"
  },
  "id": "CFZQx-Y2pDvy",
  "outputId": "ee97513e-10a9-45ef-80a4-ceb119cb6e03"
 },
 "outputs": [
  {
   "data": {
    "text/plain": [
     "(11043, 1)"
    ]
   },
   "execution_count": 47,
   "metadata": {},
   "output_type": "execute_result"
  }
 ],
 "source": [
  "y.shape"
 ]
},
{
 "cell_type": "markdown",
 "metadata": {
  "id": "hT9X379RZ4d9"
 },
 "source": [
  "# SPRINT-2"
 ]
},
{
 "cell_type": "markdown",
 "metadata": {
    "id": "7V0-vokFprCT"
   },
   "source": [
    "**TRAIN-TEST-SPLIT**"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 48,
   "metadata": {
    "id": "UMFjPPCjpqRo"
   },
   "outputs": [],
   "source": [
    "x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=0)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 49,
   "metadata": {
    "colab": {
     "base_uri": "https://localhost:8080/"
    },
    "id": "5PdWqhymDjuh",
    "outputId": "882dc0ff-a447-41aa-f3c0-f6d963160902"
   },
   "outputs": [
    {
     "data": {
      "text/plain": [
       "(2209, 16)"
      ]
     },
     "execution_count": 49,
     "metadata": {},
     "output_type": "execute_result"
    }
   ],
   "source": [
    "x_test.shape"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 50,
   "metadata": {
    "colab": {
     "base_uri": "https://localhost:8080/"
    },
    "id": "mKQQr8n7Dj0U",
    "outputId": "701d1d1f-1b9e-462d-9a7a-769788329397"
   },
   "outputs": [
    {
     "data": {
      "text/plain": [
       "(8834, 16)"
      ]
     },
     "execution_count": 50,
     "metadata": {},
     "output_type": "execute_result"
    }
   ],
   "source": [
    "x_train.shape"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 51,
   "metadata": {
    "colab": {
     "base_uri": "https://localhost:8080/"
    },
  "id": "SDEFR1ftDg1M",
  "outputId": "66f80b51-7571-47a1-add5-727af4cf7179"
 },
 "outputs": [
  {
   "data": {
    "text/plain": [
     "(2209, 1)"
    ]
   },
   "execution_count": 51,
   "metadata": {},
   "output_type": "execute_result"
  }
 ],
 "source": [
  "y_test.shape"
 ]
},
{
 "cell_type": "code",
 "execution_count": 52,
 "metadata": {
  "colab": {
   "base_uri": "https://localhost:8080/"
  },
  "id": "ZWxmKWgBq3SN",
  "outputId": "4d1993e9-c166-41bf-afa8-2c90c1083af5"
 },
 "outputs": [
  {
   "data": {
    "text/plain": [
     "(8834, 1)"
    ]
   },
   "execution_count": 52,
   "metadata": {},
   "output_type": "execute_result"
  }
 ],
 "source": [
  "y_train.shape"
 ]
},
{
 "cell_type": "markdown",
 "metadata": {
  "id": "_Xkyx0_TZ4eE"
 },
 "source": [
  "**STANDARDIZING INPUT VALUES**"
 ]
},
{
 "cell_type": "code",
 "execution_count": 53,
 "metadata": {
  "id": "LRHL7sa-q4l-"
 },
 "outputs": [],
 "source": [
  "sc = StandardScaler()"
 ]
},
{
 "cell_type": "code",
 "execution_count": 54,
 "metadata": {
  "id": "mcbGjZF1Z4eF"
 },
 "outputs": [],
 "source": [
  "x_train=sc.fit_transform(x_train)"
 ]
},
{
 "cell_type": "code",
 "execution_count": 55,
 "metadata": {
  "id": "q7Sd8VtXZ4eF"
 },
 "outputs": [],
 "source": [
  "x_test=sc.fit_transform(x_test)"
 ]
},
{
 "cell_type": "markdown",
 "metadata": {
  "id": "J-eHfAs5Z4eG"
 },
 "source": [
  "**MODEL BUILDING**"
 ]
},
{
 "cell_type": "code",
 "execution_count": 56,
 "metadata": {
  "id": "TP1Skb1JZ4eG"
 },
 "outputs": [],
 "source": [
  "classifier = DecisionTreeClassifier(random_state=0)"
 ]
},
{
 "cell_type": "code",
 "execution_count": 57,
 "metadata": {
  "colab": {
   "base_uri": "https://localhost:8080/"
  },
   "id": "K7geli07Z4eH",
   "outputId": "953355e9-7aed-418f-bdeb-8c8f36b403cc"
  },
  "outputs": [
   {
    "data": {
     "text/plain": [
      "DecisionTreeClassifier(random_state=0)"
     ]
    },
    "execution_count": 57,
    "metadata": {},
    "output_type": "execute_result"
   }
  ],
  "source": [
   "classifier.fit(x_train,y_train)"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 58,
  "metadata": {
   "id": "rZUOCwBXZ4eH"
  },
  "outputs": [],
  "source": [
   "predicted = classifier.predict(x_test)"
  ]
 },
 {
  "cell_type": "markdown",
  "metadata": {
   "id": "Y2i3fZ8SZ4eI"
  },
  "source": [
   "**MODEL EVALUATION**"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 59,    "metadata":
{
   "id": "hsHXutwHZ4eJ"
  },
  "outputs": [],
  "source": [
   "acc=accuracy_score(predicted,y_test)"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 60,
  "metadata": {
   "colab": {
    "base_uri": "https://localhost:8080/"
   },
   "id": "JMQ4c0BSZ4eJ",
   "outputId": "30461680-8f43-4d10-89c8-efd87602596f"
  },
  "outputs": [
   {
    "data": {
     "text/plain": [
      "0.8791308284291535"
     ]
    },
    "execution_count": 60,
    "metadata": {},
    "output_type": "execute_result"
   }
  ],
  "source": [
   "acc"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 61,
  "metadata": {
   "colab": {
    "base_uri": "https://localhost:8080/"
   },
   "id": "fkd7YjxaZ4eK",
   "outputId": "190d4603-8572-4fe4-c6b2-02a4f78e4c77"
  },
  "outputs": [
   {
    "data": {
     "text/plain": [
      "array([1.187e+03, 1.000e+00, 1.500e+01, 5.000e+00, 1.900e+01, 1.000e+00,\n",
      "           1.000e+00, 0.000e+00, 0.000e+00, 0.000e+00, 1.000e+00, 0.000e+00,\n",           "
0.000e+00, 0.000e+00, 0.000e+00, 0.000e+00, 1.000e+00])"
     ]
    },
    "execution_count": 61,
    "metadata": {},
    "output_type": "execute_result"
   }
  ],
  "source": [
   "data[data['ARR_DEL15']>0].iloc[33].values"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 62,
  "metadata": {
   "id": "EYFEFBViZ4eK"
  },
  "outputs": [],
  "source": [
   "sample=[[1.187e+03, 1.000e+00, 1.500e+01, 5.000e+00, 1.900e+01, 1.000e+00,
0.000e+00, 0.000e+00, 0.000e+00, 1.000e+00, 0.000e+00,\n",
  "       0.000e+00, 0.000e+00, 0.000e+00, 0.000e+00, 1.000e+00]]"
 ]
},
{
 "cell_type": "code",
 "execution_count": 63,
 "metadata": {
  "colab": {
   "base_uri": "https://localhost:8080/"
  },
  "id": "FG6dr3DZZ4eL",
  "outputId": "c295a5f4-77a6-486b-df75-9ab3657f2b93"
 },
 "outputs": [
  {
   "data": {
    "text/plain": [
     "array([0.])"
    ]
   },
   "execution_count": 63,
   "metadata": {},
   "output_type": "execute_result"
  }
 ],
 "source": [
  "classifier.predict(sample)"
 ]
},
{
 "cell_type": "markdown",
 "metadata": {
  "id": "jn4MugWoZ4eL"
 },
 "source": []
},
{
 "cell_type": "code",
   "execution_count": 64,
   "metadata": {
    "id": "5cdbEErbZ4eM"
   },
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "Requirement already satisfied: ibm-watson-machine-learning in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (1.0.257)\n",
      "Requirement already satisfied: ibm-cos-sdk==2.11.* in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm-watson-machine-learning) (2.11.0)\n",
      "Requirement already satisfied: importlib-metadata in /opt/conda/envs/Python3.9/lib/python3.9/site-packages (from ibm-watson-machine-learning) (4.8.2)\n",
      "Requirement already satisfied: lomond in /opt/conda/envs/Python3.9/lib/python3.9/site-packages (from ibm-watson-machine-learning) (0.3.3)\n",
      "Requirement already satisfied: pandas<1.5.0,>=0.24.2 in /opt/conda/envs/Python3.9/lib/python3.9/site-packages (from ibm-watson-machine-learning) (1.3.4)\n",
      "Requirement already satisfied: packaging in /opt/conda/envs/Python3.9/lib/python3.9/site-packages (from ibm-watson-machine-learning) (21.3)\n",
      "Requirement already satisfied: requests in /opt/conda/envs/Python3.9/lib/python3.9/site-packages (from ibm-watson-machine-learning) (2.26.0)\n",
      "Requirement already satisfied: urllib3 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm-watson-machine-learning) (1.26.7)\n",
      "Requirement already satisfied: tabulate in /opt/conda/envs/Python3.9/lib/python3.9/site-packages (from ibm-watson-machine-learning) (0.8.9)\n",
      "Requirement already satisfied: certifi in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm-watson-machine-learning) (2022.9.24)\n",
      "Requirement already satisfied: ibm-cos-sdk-core==2.11.0 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm-cos-sdk==2.11.*->ibm-watson-machine-learning) (2.11.0)\n",
      "Requirement already satisfied: ibm-cos-sdk-s3transfer==2.11.0 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm-cos-sdk==2.11.*->ibm-watson-machine-learning) (2.11.0)\n",
     "Requirement already satisfied: jmespath<1.0.0,>=0.7.1 in
/opt/conda/envs/Python3.9/lib/python3.9/site-packages (from ibm-cos-sdk==2.11.*->ibm-
watson-machinelearning) (0.10.0)\n",
     "Requirement already satisfied: python-dateutil<3.0.0,>=2.1 in /opt/conda/envs/Python-
3.9/lib/python3.9/site-packages (from ibm-cos-sdkcore==2.11.0->ibm-cos-sdk==2.11.*->ibm-
watson-machine-learning) (2.8.2)\n",
     "Requirement already satisfied: pytz>=2017.3 in /opt/conda/envs/Python-
3.9/lib/python3.9/site-packages (from pandas<1.5.0,>=0.24.2->ibm-watson-machinelearning)
(2021.3)\n",
     "Requirement already satisfied: numpy>=1.17.3 in
/opt/conda/envs/Python3.9/lib/python3.9/site-packages (from pandas<1.5.0,>=0.24.2->ibm-
watson-machinelearning) (1.20.3)\n",
     "Requirement already satisfied: six>=1.5 in /opt/conda/envs/Python-
3.9/lib/python3.9/site-packages (from python-dateutil<3.0.0,>=2.1->ibm-cos-sdkcore==2.11.0-
>ibm-cos-sdk==2.11.*->ibm-watson-machine-learning) (1.15.0)\n",
     "Requirement already satisfied: charset-normalizer~=2.0.0 in
/opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from requests->ibm-
watsonmachine-learning) (2.0.4)\n",
     "Requirement already satisfied: idna<4,>=2.5 in /opt/conda/envs/Python-
3.9/lib/python3.9/site-packages (from requests->ibm-watson-machine-learning) (3.3)\n",
     "Requirement already satisfied: zipp>=0.5 in /opt/conda/envs/Python-
3.9/lib/python3.9/site-packages (from importlib-metadata->ibm-watson-machinelearning)
(3.6.0)\n",
     "Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in
/opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from packaging->ibm-
watsonmachine-learning) (3.0.4)\n"
    ]
   }
  ],
  "source": [
   "!pip install -U ibm-watson-machine-learning"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 74,
  "metadata": {},
  "outputs": [],
   "source": [
    "from ibm_watson_machine_learning import APIClient\n",
    "import json"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 75,
   "metadata": {},
   "outputs": [],
   "source": [
    "wml_credentials = {\n",
    "    \"apikey\":\"zCU3gbntxqL8kInfTM2Q95jPfkfkVI9Mt8sLNC8NRipq\",\n",
    "    \"url\":\"https://eu-de.ml.cloud.ibm.com\"\n",
    "}"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 76,
   "metadata": {},
   "outputs": [],
   "source": [
    "wml_client = APIClient(wml_credentials)\n"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 77,
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "Note: 'limit' is not provided. Only first 50 records will be displayed if the number of records exceed 50\n",
      "------------------------------------  -----------  ------------------------\n",
     "ID                                    NAME        CREATED\n",
     "d0a11148-f6ea-4361-aec2-6f11167aec40  flightdelay  2022-11-
18T02:34:41.070Z\n",
     "------------------------------------  -----------  ------------------------\n"     ]
    }
   ],
   "source": [
    "wml_client.spaces.list()"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 78,
   "metadata": {},
   "outputs": [],
   "source": [
    "SPACE_ID= \"d0a11148-f6ea-4361-aec2-6f11167aec40\""
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 79,
   "metadata": {},
   "outputs": [
    {
     "data": {
      "text/plain": [
       "'SUCCESS'"
      ]
     },
     "execution_count": 79,
     "metadata": {},
     "output_type": "execute_result"
    }
   ],
   "source": [
    "wml_client.set.default_space(SPACE_ID)"
   ]
```

```
 },
 {
  "cell_type": "code",
  "execution_count": 80,
  "metadata": {},
  "outputs": [
   {
    "name": "stdout",
    "output_type": "stream",
    "text": [
     "------------------------------  ------------------------------------  ----\n",
     "NAME                            ASSET_ID                              TYPE\n",
     "default_py3.6                   0062b8c9-8b7d-44a0-a9b9-46c416adcbd9  base\n",
     "kernel-spark3.2-scala2.12                   020d69ce-7ac1-5e68-ac1a-31189867356a     base\n",
"pytorch-onnx_1.3-py3.7-edt      069ea134-3346-5748-b513-49120e15d288  base\n",
     "scikit-learn_0.20-py3.6        09c5a1d0-9c1e-4473-a344-eb7b665ff687  base\n",
     "spark-mllib_3.0-scala_2.12      09f4cff0-90a7-5899-b9ed-1ef348aebdee  base\n",
     "pytorch-onnx_rt22.1-py3.9       0b848dd4-e681-5599-be41-b5f6fccc6471  base\n",
     "ai-function_0.1-py3.6          0cdb0f1e-5376-4f4d-92dd-da3b69aa9bda  base\n",
     "shiny-r3.6                     0e6e79df-875e-4f24-8ae9-62dcc2148306  base\n",
"tensorflow_2.4-py3.7-horovod    1092590a-307d-563d-9b62-4eb7d64b3f22  base\n",
     "pytorch_1.1-py3.6              10ac12d6-6b30-4ccd-8392-3e922c096a92  base\n",
     "tensorflow_1.15-py3.6-ddl       111e41b3-de2d-5422-a4d6-bf776828c4b7  base\n",
     "autoai-kb_rt22.2-py3.10        125b6d9a-5b1f-5e8d-972a-b251688ccf40  base\n",
     "runtime-22.1-py3.9             12b83a17-24d8-5082-900f-0ab31fbfd3cb  base\n",
     "scikit-learn_0.22-py3.6        154010fa-5b3b-4ac1-82af-4d5ee5abbc85  base\n",
     "default_r3.6                   1b70aec3-ab34-4b87-8aa0-a4a3c8296a36  base\n",
     "pytorch-onnx_1.3-py3.6       1bc6029a-cc97-56da-b8e0-39c3880dbbe7 base\n",     "kernel-
spark3.3-r3.6         1c9e5454-f216-59dd-a20e-474a5cdf5988 base\n",     "pytorch-onnx_rt22.1-
py3.9-edt    1d362186-7ad5-5b59-8b6c-9d0880bde37f  base\n",
     "tensorflow_2.1-py3.6           1eb25b84-d6ed-5dde-b6a5-3fbdf1665666  base\n",
     "spark-mllib_3.2               20047f72-0a98-58c7-9ff5-a77b012eb8f5  base\n",
     "tensorflow_2.4-py3.8-horovod    217c16f6-178f-56bf-824a-b19f20564c49  base\n",
     "runtime-22.1-py3.9-cuda        26215f05-08c3-5a41-a1b0-da66306ce658  base\n",
     "do_py3.8                       295addb5-9ef9-547e-9bf4-92ae3563e720  base\n",
     "autoai-ts_3.8-py3.8            2aa0c932-798f-5ae9-abd6-15e0c2402fb5  base\n",
     "tensorflow_1.15-py3.6          2b73a275-7cbf-420b-a912-eae7f436e0bc  base\n",
     "kernel-spark3.3-py3.9          2b7961e2-e3b1-5a8c-a491-482c8368839a  base\n",
    "pytorch_1.2-py3.6             2c8ef57d-2687-4b7d-acce-01f94976dac1  base\n",
    "spark-mllib_2.3                          2e51f700-bca0-4b0d-88dc-5c6791338875   base\n",
"pytorch-onnx_1.1-py3.6-edt      32983cea-3f32-4400-8965-dde874a8d67e  base\n",
    "spark-mllib_3.0-py37           36507ebe-8770-55ba-ab2a-eafe787600e9  base\n",
    "spark-mllib_2.4              390d21f8-e58b-4fac-9c55-d7ceda621326  base\n",
    "autoai-ts_rt22.2-py3.10        396b2e83-0953-5b86-9a55-7ce1628a406f  base\n",
    "xgboost_0.82-py3.6            39e31acd-5f30-41dc-ae44-60233c80306e  base\n",
    "pytorch-onnx_1.2-py3.6-edt      40589d0e-7019-4e28-8daa-fb03b6f4fe12  base\n",
    "pytorch-onnx_rt22.2-py3.10      40e73f55-783a-5535-b3fa-0c8b94291431  base\n",
    "default_r36py38              41c247d3-45f8-5a71-b065-8580229facf0  base\n",
    "autoai-ts_rt22.1-py3.9         4269d26e-07ba-5d40-8f66-2d495b0c71f7  base\n",
    "autoai-obm_3.0               42b92e18-d9ab-567f-988a-4240ba1ed5f7  base\n",
    "pmml-3.0_4.3                493bcb95-16f1-5bc5-bee8-81b8af80e9c7  base\n",
    "spark-mllib_2.4-r_3.6          49403dff-92e9-4c87-a3d7-a42d0021c095  base\n",
    "xgboost_0.90-py3.6            4ff8d6c2-1343-4c18-85e1-689c965304d3  base\n",
    "pytorch-onnx_1.1-py3.6         50f95b2a-bc16-43bb-bc94-b0bed208c60b  base\n",
    "autoai-ts_3.9-py3.8           52c57136-80fa-572e-8728-a5e7cbb42cde  base\n",
    "spark-mllib_2.4-scala_2.11      55a70f99-7320-4be5-9fb9-9edb5a443af5  base\n",
    "spark-mllib_3.0              5c1b0ca2-4977-5c2e-9439-ffd44ea8ffe9  base\n",
    "autoai-obm_2.0               5c2e37fa-80b8-5e77-840f-d912469614ee  base\n",
    "spss-modeler_18.1             5c3cad7e-507f-4b2a-a9a3-ab53a21dee8b  base\n",
    "cuda-py3.8                  5d3232bf-c86b-5df4-a2cd-7bb870a1cd4e  base\n",
    "runtime-22.2-py3.10-xc         5e8cddff-db4a-5a6a-b8aa-2d4af9864dab  base\n",
    "autoai-kb_3.1-py3.7           632d4b22-10aa-5180-88f0-f52dfb6444d7  base\n",
    "pytorch-onnx_1.7-py3.8        634d3cdc-b562-5bf9-a2d4-ea90a478456b  base\n",     "spark-
mllib_2.3-r_3.6          6586b9e3-ccd6-4f92-900f-0f8cb2bd6f0c  base\n",
    "tensorflow_2.4-py3.7           65e171d7-72d1-55d9-8ebb-f813d620c9bb  base\n",
    "spss-modeler_18.2             687eddc9-028a-4117-b9dd-e57b36f1efa5  base\n",
    "pytorch-onnx_1.2-py3.6         692a6a4d-2c4d-45ff-a1ed-b167ee55469a  base\n",
    "spark-mllib_2.3-scala_2.11      7963efe5-bbec-417e-92cf-0574e21b4e8d  base\n",
    "spark-mllib_2.4-py37           7abc992b-b685-532b-a122-a396a3cdbaab  base\n",
    "caffe_1.0-py3.6              7bb3dbe2-da6e-4145-918d-b6d84aa93b6b  base\n",
    "pytorch-onnx_1.7-py3.7         812c6631-42b7-5613-982b-02098e6c909c  base\n",
    "cuda-py3.6                           82c79ece-4d12-40e6-8787-a7b9e0f62770   base\n",
"tensorflow_1.15-py3.6-horovod   8964680e-d5e4-5bb8-919b-8342c6c0dfd8  base\n",
    "hybrid_0.1                  8c1a58c6-62b5-4dc4-987a-df751c2756b6  base\n",
    "pytorch-onnx_1.3-py3.7         8d5d8a87-a912-54cf-81ec-3914adaa988d  base\n",
    "caffe-ibm_1.0-py3.6           8d863266-7927-4d1e-97d7-56a7f4c0a19b  base\n",
    "spss-modeler_17.1              902d0051-84bd-4af6-ab6b-8f6aa6fdeabb  base\n",
    "do_12.10                       9100fd72-8159-4eb9-8a0b-a87e12eefa36  base\n",
    "do_py3.7                       9447fa8b-2051-4d24-9eef-5acb0e3c59f8  base\n",
    "spark-mllib_3.0-r_3.6          94bb6052-c837-589d-83f1-f4142f219e32  base\n",
    "cuda-py3.7-opence              94e9652b-7f2d-59d5-ba5a-23a414ea488f  base\n",
    "nlp-py3.8                      96e60351-99d4-5a1c-9cc0-473ac1b5a864  base\n",
    "cuda-py3.7                     9a44990c-1aa1-4c7d-baf8-c4099011741c  base\n",
    "hybrid_0.2                     9b3f9040-9cee-4ead-8d7a-780600f542f7  base\n",
    "spark-mllib_3.0-py38           9f7a8fc1-4d3c-5e65-ab90-41fa8de2d418  base\n",      "autoai-
kb_3.3-py3.7            a545cca3-02df-5c61-9e88-998b09dc79af  base\n",
    "spark-mllib_3.0-py39           a6082a27-5acc-5163-b02c-6b96916eb5e0  base\n",
    "runtime-22.1-py3.9-do          a7e7dbf1-1d03-5544-994d-e5ec845ce99a  base\n",
    "default_py3.8                  ab9e1b80-f2ce-592c-a7d2-4f2344f77194  base\n",
    "tensorflow_rt22.1-py3.9        acd9c798-6974-5d2f-a657-ce06e986df4d  base\n",
    "kernel-spark3.2-py3.9          ad7033ee-794e-58cf-812e-a95f4b64b207  base\n",
    "autoai-obm_2.0 with Spark 3.0    af10f35f-69fa-5d66-9bf5-acb58434263a  base\n",
    "runtime-22.2-py3.10            b56101f1-309d-549b-a849-eaa63f77b2fb  base\n",
    "default_py3.7_opence           c2057dd4-f42c-5f77-a02f-72bdbd3282c9  base\n",
    "tensorflow_2.1-py3.7           c4032338-2a40-500a-beef-b01ab2667e27  base\n",
    "do_py3.7_opence                cc8f8976-b74a-551a-bb66-6377f8d865b4  base\n",
    "spark-mllib_3.3                d11f2434-4fc7-58b7-8a62-755da64fdaf8  base\n",
    "autoai-kb_3.0-py3.6            d139f196-e04b-5d8b-9140-9a10ca1fa91a  base\n",
    "spark-mllib_3.0-py36           d82546d5-dd78-5fbb-9131-2ec309bc56ed  base\n",      "autoai-
kb_3.4-py3.8            da9b39c3-758c-5a4f-9cfd-457dd4d8c395  base\n",
    "kernel-spark3.2-r3.6           db2fe4d6-d641-5d05-9972-73c654c60e0a  base\n",
    "autoai-kb_rt22.1-py3.9         db6afe93-665f-5910-b117-d879897404d9  base\n",
"tensorflow_rt22.1-py3.9-horovod  dda170cc-ca67-5da7-9b7a-cf84c6987fae  base\n",
    "autoai-ts_1.0-py3.7            deef04f0-0c42-5147-9711-89f9904299db  base\n",
    "tensorflow_2.1-py3.7-horovod    e384fce5-fdd1-53f8-bc71-11326c9c635f  base\n",
    "default_py3.7                  e4429883-c883-42b6-87a8-f419d64088cd  base\n",
    "do_22.1                        e51999ba-6452-5f1f-8287-17228b88b652  base\n",
    "autoai-obm_3.2                 eae86aab-da30-5229-a6a6-1d0d4e368983  base\n",
    "runtime-22.2-r4.2              ec0a3d28-08f7-556c-9674-ca7c2dba30bd  base\n",
    "tensorflow_rt22.2-py3.10       f65bd165-f057-55de-b5cb-f97cf2c0f393  base\n",
    "do_20.1                        f686cdd9-7904-5f9d-a732-01b0d6b10dc5  base\n",      "pytorch-
onnx_rt22.2-py3.10-edt   f8a05d07-e7cd-57bb-a10b-23f1d4b837ac  base\n",
    "scikit-learn_0.19-py3.6        f963fa9d-4bb7-5652-9c5d-8d9289ef6ad9  base\n",
    "tensorflow_2.4-py3.8           fe185c44-9a99-5425-986b-59bd1d2eda46  base\n",
    "------------------------------  -----------------------------------  ----\n"      ]
   }
  ],
  "source": [
   "wml_client.software_specifications.list(500)"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 81,
  "metadata": {},
  "outputs": [
   {
    "data": {
     "text/plain": [
      "'1.0.2'"
     ]
    },
    "execution_count": 81,
    "metadata": {},
    "output_type": "execute_result"
   }
  ],
  "source": [
   "import sklearn \n",
   "sklearn.__version__"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 86,
  "metadata": {},
  "outputs": [],
  "source": [
   "MODEL_NAME = 'flightdelay'\n",
   "DEPLOYMENT_NAME = \"flightdelay\"\n",
   "DEMO_MODEL = classifier"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 83,
  "metadata": {},
  "outputs": [],
  "source": [
   "# Set Python Version\n",
   "software_spec_uid = wml_client.software_specifications.get_id_by_name('runtime-
22.1-py3.9')"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 84,
  "metadata": {},
  "outputs": [],
  "source": [
   "# Setup model meta\n",
   "model_props = {\n",
   "    wml_client.repository.ModelMetaNames.NAME: MODEL_NAME, \n",
   "    wml_client.repository.ModelMetaNames.TYPE: 'scikit-learn_1.0', \n",     "
wml_client.repository.ModelMetaNames.SOFTWARE_SPEC_UID:
software_spec_uid \n",
   "}"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 87,
  "metadata": {},
  "outputs": [],
  "source": [
   "#Save model\n",
   "model_details = wml_client.repository.store_model(\n",
   "    model=DEMO_MODEL, \n",
   "    meta_props=model_props, \n",
   "    training_data=x_train, \n",
   "    training_target=y_train\n",
   ")"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 88,
  "metadata": {},
  "outputs": [
   {
    "data": {
     "text/plain": [
      "{'entity': {'hybrid_pipeline_software_specs': [],\n",
      "  'label_column': 'l0',\n",
      "  'schemas': {'input': [{'fields': [{'name': 'f0', 'type': 'float'},\n",
      "      {'name': 'f1', 'type': 'float'},\n",
      "      {'name': 'f2', 'type': 'float'},\n",
      "      {'name': 'f3', 'type': 'float'},\n",
      "      {'name': 'f4', 'type': 'float'},\n",
      "      {'name': 'f5', 'type': 'float'},\n",
      "      {'name': 'f6', 'type': 'float'},\n",
      "      {'name': 'f7', 'type': 'float'},\n",
      "      {'name': 'f8', 'type': 'float'},\n",
      "      {'name': 'f9', 'type': 'float'},\n",
      "     {'name': 'f10', 'type': 'float'},\n",      "
{'name': 'f11', 'type': 'float'},\n",
      "      {'name': 'f12', 'type': 'float'},\n",
      "      {'name': 'f13', 'type': 'float'},\n",
      "      {'name': 'f14', 'type': 'float'},\n",
      "      {'name': 'f15', 'type': 'float'}],\n",
      "     'id': '1',\n",
      "     'type': 'struct'}],\n",
      "   'output': []},\n",
      "  'software_spec': {'id': '12b83a17-24d8-5082-900f-0ab31fbfd3cb',\n",
      "   'name': 'runtime-22.1-py3.9'},\n",
      "  'type': 'scikit-learn_1.0'},\n",
      " 'metadata': {'created_at': '2022-11-18T02:48:27.322Z',\n",
      "  'id': '1dfde4a2-7300-4e47-89a1-977b7135d2ed',\n",
     "  'modified_at': '2022-11-18T02:48:30.866Z',\n",
     "  'name': 'flightdelay',\n",
     "  'owner': 'IBMid-66300415QB',\n",
     "  'resource_key': 'd7c9b9ed-9974-4243-90a9-999467ee2ccd',\n",
     "  'space_id': 'd0a11148-f6ea-4361-aec2-6f11167aec40'},\n",
     " 'system': {'warnings': []}}"
    ]
   },
   "execution_count": 88,
   "metadata": {},
   "output_type": "execute_result"
  }
 ],
 "source": [
  "model_details"
 ]
},
{
 "cell_type": "code",
 "execution_count": 89,
 "metadata": {},
 "outputs": [
  {
   "data": {
    "text/plain": [
     "'1dfde4a2-7300-4e47-89a1-977b7135d2ed'"
    ]
   },
   "execution_count": 89,
   "metadata": {},
   "output_type": "execute_result"
  }
 ],
 "source": [
  "model_id = wml_client.repository.get_model_id(model_details)\n",     "model_id"
 ]
},
{
   "cell_type": "code",
   "execution_count": 90,
   "metadata": {},
   "outputs": [],
   "source": [
    "# Set meta\n",
    "deployment_props = {\n",
    "    wml_client.deployments.ConfigurationMetaNames.NAME:DEPLOYMENT_NAME,
\n",
    "    wml_client.deployments.ConfigurationMetaNames.ONLINE: {}\n",
    "}"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 91,
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "\n",
      "\n",

"#####################################################################
##################\n",
      "\n",
      "Synchronous deployment creation for uid: '1dfde4a2-7300-4e47-89a1-
977b7135d2ed' started\n",
      "\n",

"#####################################################################
##################\n",
      "\n",
      "\n",
      "initializing\n",
      "Note: online_url is deprecated and will be removed in a future release. Use serving_urls
instead.\n",
      "\n",
      "ready\n",
      "\n",
      "\n",
      "-------------------------------------------------------------------------------------------------\n",
      "Successfully finished deployment creation, deployment_uid='abf3959e-b7bd-4fde-
9f34-1295348fea93'\n",
      "-------------------------------------------------------------------------------------------------\n",      "\n",
      "\n"
     ]
    }
   ],
   "source": [
    "# Deploy\n",
    "deployment = wml_client.deployments.create(\n",
    "    artifact_uid=model_id, \n",
    "    meta_props=deployment_props \n",
    ")"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": []
  }
 ],
 "metadata": {
  "colab": {
   "collapsed_sections": [],
   "provenance": []
  },
  "kernelspec": {
   "display_name": "Python 3.9",
   "language": "python",
   "name": "python3"
  },
  "language_info": {
   "codemirror_mode": {
    "name": "ipython",
    "version": 3
   },
   "file_extension": ".py",
   "mimetype": "text/x-python",
   "name": "python",
   "nbconvert_exporter": "python",
   "pygments_lexer": "ipython3",
   "version": "3.9.13"
  }
 },
 "nbformat": 4,
 "nbformat_minor": 1
}
```

GITHUB LINK

https://github.com/IBM-EPBL/IBM-Project-38319-1660377953

DEMO LINK

https://drive.google.com/file/d/118dL30CuuwKDxkTKR_zTKDppbsVfsn7C/view?usp=drivesdk