Flight Delay Prediction using IBM Cloud

Team Number:	27
Team Members:	Akash Venkat (20BCB0079)
	Atharwa Namjoshi (20BCB0052)
	Ravi Kiran (20BCE2797)
	Ananya Chembai (20BCE2358)

1. INTRODUCTION

1.1 Overview

Over the last twenty years, air travel has been increasingly preferred among travellers, mainly because of its speed and in some cases comfort. This has led to phenomenal growth in air traffic and on the ground. An increase in air traffic growth has also resulted in massive levels of aircraft delays on the ground and in the air. These delays are responsible for large economic and environmental losses. The main objective of the model is to predict flight delays accurately in order to optimize flight operations and minimize delays.

1.2 Purpose

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. We then use decision tree classifier to predict if the flight arrival will be delayed or not. A flight is considered to be delayed when difference between scheduled and actual arrival times is greater than 15 minutes. Furthermore, we compare decision tree classifier with logistic regression and a simple neural network for various figures of merit.

2. LITERATURE SURVEY

2.1 Existing Problem

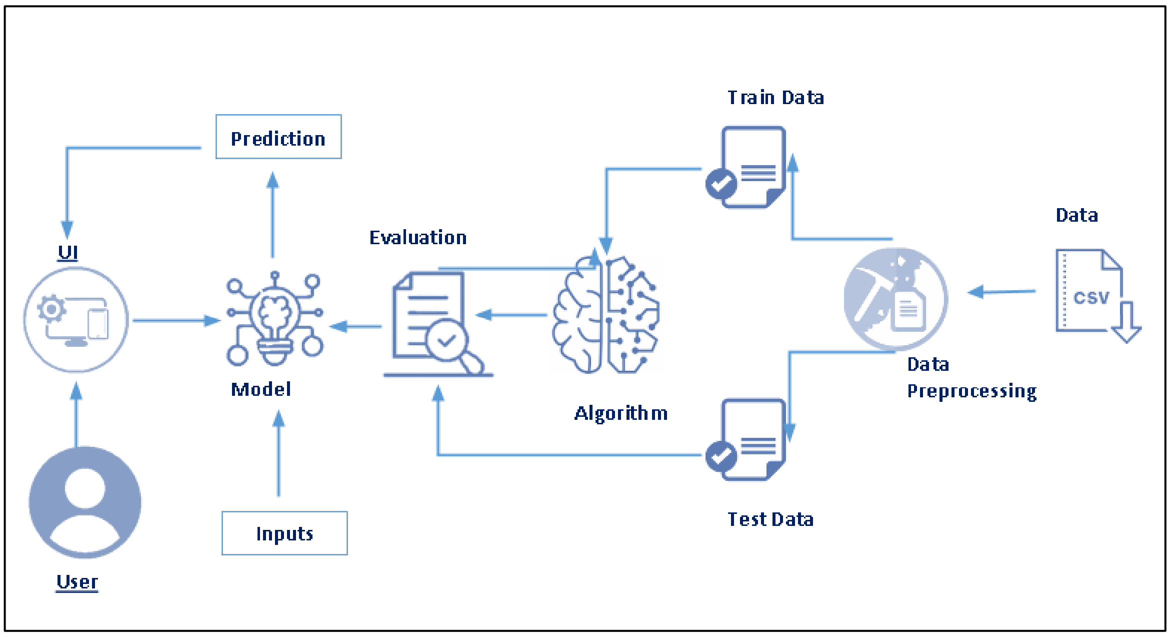
Previously, this problem is solved by using R4ML, a scalable R package, running on IBM Watson Studio to perform various Machine Learning exercises. Watson Studio is an interactive, collaborative, cloud-based environment where data scientists, developers, and others interested in data science can use tools (e.g., RStudio, Jupyter Notebooks, Spark, etc.) to collaborate, share, and gather insight from their data. R4ML provides various out-of-the-box algorithms to experiment with.

2.2 Proposed Solution

Our group has proposed a solution to solve this problem by using K-Nearest Neighbour (KNN) algorithm and Random Forest Classifier algorithm. Both methods are used separately to train models, and accuracy scores are compared for both models. It is observed that Random Forest Classifier model has a higher accuracy.

3. THEORETICAL ANALYSIS

3.1 Block Diagram



3.2 Hardware/Software Designing

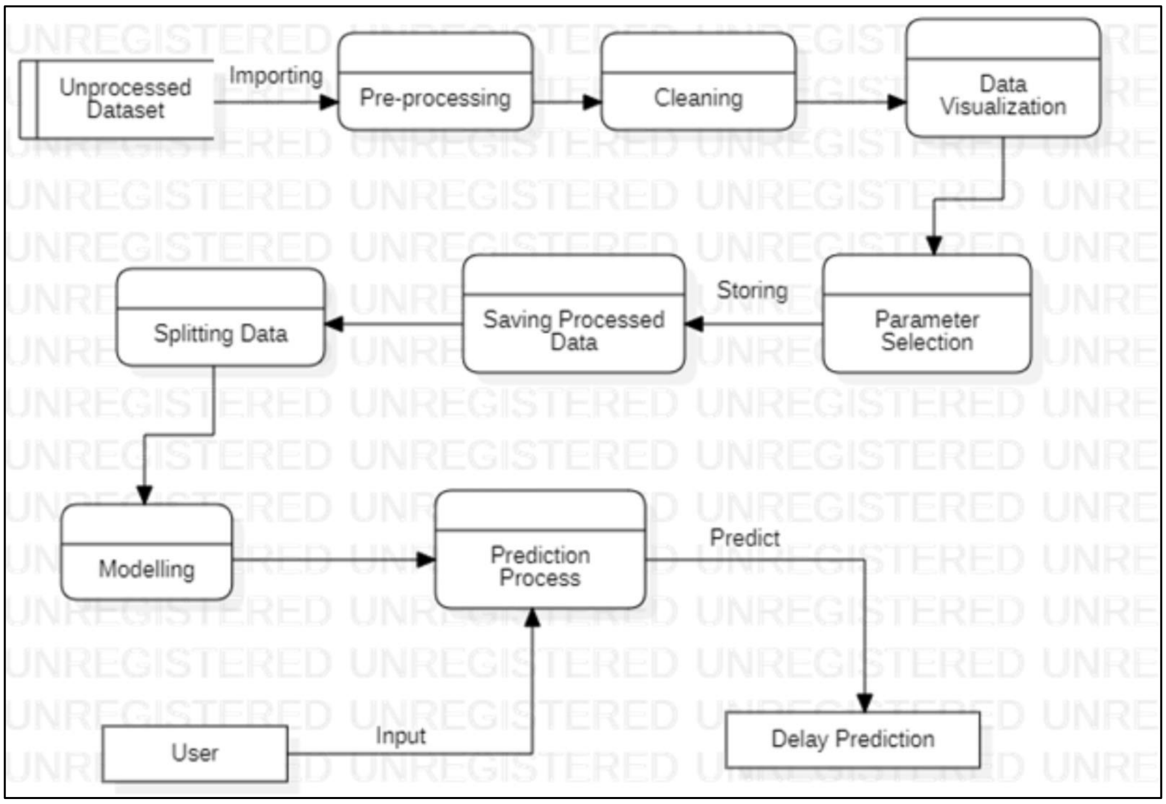
For this project, basic hardware requirements are required, such as a laptop or a PC to run the softwares for training and executing the model.

The softwares necessary include Jupyter notebooks for building the ML model, IBM Watson for training the model, and Anaconda Spyder for coding and running the HTML files.

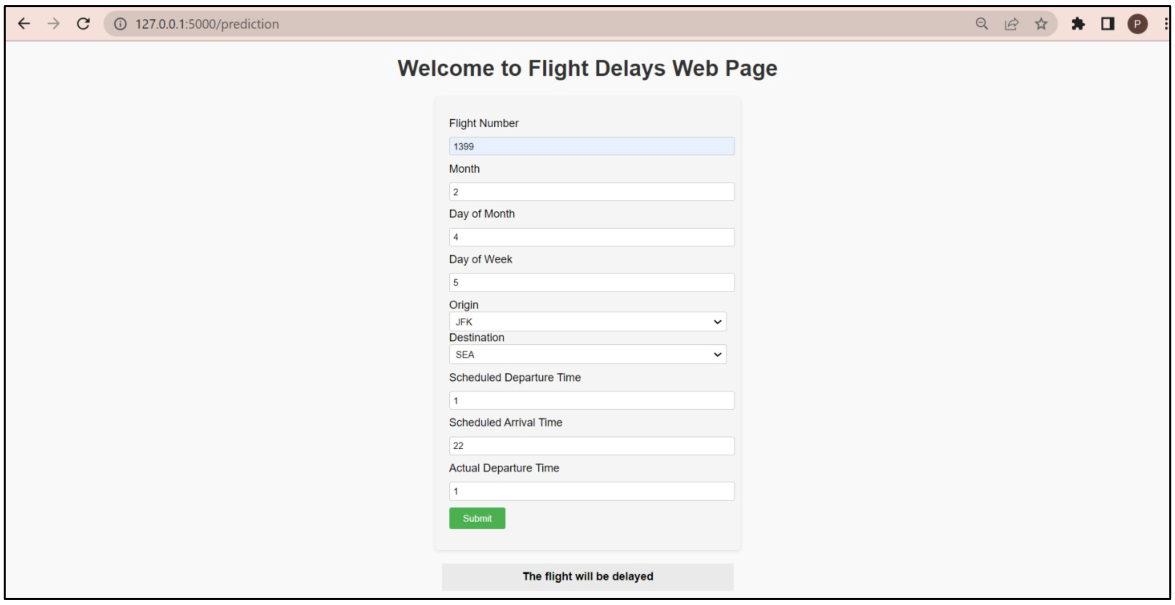
4. EXPERIMENTAL INVESTIGATIONS

a. The algorithms used for the model are the KNN algorithm and Random Forest Classifier algorithm.
b. The 'ARR_DEL15' variable is the dependent variable, while all other variables are independent.
c. One Hot Encoding is applied to the categorical independent variables.
d. The Departure Delay is closely related to Arrival Delay.
e. The final accuracy of the Random Forest Classification model comes out to be 92%.

5. FLOWCHART



6. RESULT

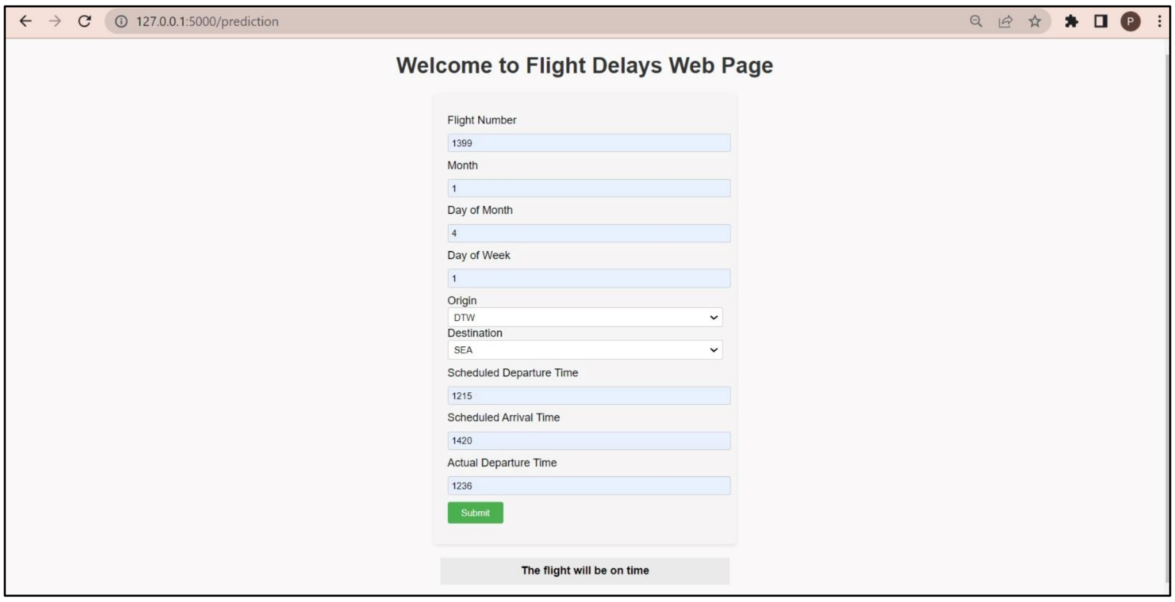

127.0.0.1:5000/prediction

Welcome to Flight Delays Web Page

Flight Number: 1399
Month: 2
Day of Month: 4
Day of Week: 5
Origin: JFK
Destination: SEA
Scheduled Departure Time: 1
Scheduled Arrival Time: 22
Actual Departure Time: 1

Submit

The flight will be delayed


127.0.0.1:5000/prediction

Welcome to Flight Delays Web Page

Flight Number: 1399
Month: 1
Day of Month: 4
Day of Week: 1
Origin: DTW
Destination: SEA
Scheduled Departure Time: 1215
Scheduled Arrival Time: 1420
Actual Departure Time: 1236

Submit

The flight will be on time

7. ADVANTAGES AND DISADVANTAGES

Advantages of Random Forest Classification:

a. High accuracy: Random Forest tends to provide high accuracy in classification tasks. It combines predictions from multiple decision trees, reducing the risk of overfitting and improving generalization.
b. Robust to overfitting: Random Forest is designed to mitigate overfitting by aggregating predictions from multiple decision trees. Each tree is trained on a random subset of features and data, leading to diverse predictions that balance bias and variance.

c. Handles high-dimensional data: Random Forest performs well even with datasets that have a large number of features. It can effectively handle high-dimensional data and identify important features for classification.
d. Non-parametric: Similar to KNN, Random Forest is a non-parametric algorithm, which means it doesn't assume a specific data distribution. It can handle complex relationships and interactions between features without requiring strong assumptions.
e. Handles missing values and outliers: Random Forest can handle missing values and outliers by using surrogate splits and averaging predictions from multiple trees. This helps in reducing the impact of missing data and outliers on the overall model performance.

Disadvantages of Random Forest Classification:

a. Black box model: Random Forest is considered a black box model because it lacks interpretability compared to simpler models like decision trees. It can be challenging to understand the underlying reasoning behind the predictions made by the ensemble of trees.
b. Computational complexity: Random Forest can be computationally expensive, especially when dealing with large datasets and a large number of trees. Building multiple decision trees and combining their predictions can require significant computational resources and time.
c. Memory requirements: Random Forest stores multiple decision trees in memory, which can consume substantial memory resources, particularly when the dataset is large or the number of trees is high.
d. Longer training time: Compared to simpler models, training a Random Forest model can take longer due to the construction of multiple decision trees and feature sampling.
e. Difficulty in handling imbalanced data: Random Forest can struggle with imbalanced datasets, where the classes have significantly different numbers of instances. It tends to favor the majority class and may require additional techniques such as class weighting or resampling to address this issue effectively.

8. APPLICATIONS

a. Airline Operations: Airlines can utilize the ML model to predict flight delays, enabling them to optimize their operations. By anticipating potential delays, airlines can efficiently allocate resources, adjust schedules, and minimize disruptions to passengers.
b. Passenger Assistance: The ML model can provide valuable information to passengers regarding the likelihood of flight delays. Passengers can be notified in advance,

allowing them to make informed decisions about their travel plans, such as adjusting their arrival time at the airport or rescheduling connecting flights.

c. Crew Scheduling: Airlines can leverage the ML model to improve crew scheduling and allocation. By predicting flight delays, they can adjust the crew schedules accordingly, ensuring the availability of appropriately trained personnel at the right time.
d. Customer Service Enhancement: The ML model can aid in enhancing customer service by enabling proactive communication with passengers affected by potential flight delays. Airlines can provide timely updates and alternative arrangements, mitigating the inconvenience caused by delays and improving customer satisfaction.
e. Airport Resource Management: Airports can utilize the ML model to optimize the utilization of airport resources, such as gates, ground staff, and baggage handling. By predicting flight delays, they can allocate resources more efficiently, streamline operations, and reduce congestion.

9. CONCLUSION

In conclusion, the development of a flight delay prediction model using IBM Watson has demonstrated its potential for significant benefits in the aviation industry. By leveraging machine learning techniques, we have created a predictive model that can accurately forecast flight delays based on historical and real-time data.

10. FUTURE SCOPE

The project can be expanded to incorporate real-time data sources such as weather conditions, air traffic data, and aircraft status. By integrating these additional factors into the model, more accurate and timely predictions can be made, improving the overall reliability of the system. Also, further exploration of feature engineering techniques can be undertaken to identify and incorporate additional relevant features that may impact flight delays. This can involve analyzing historical patterns, airline-specific factors, or external events that could affect flight operations.

11. BIBLIOGRAPHY

1. “Predicting flight delay and building an ML pipeline using R4ML”
   https://github.com/IBM/predict-flight-delay-using-r4ml/blob/master/README.md
2. “FLIGHT DELAY PREDICTION a project”
   https://scholarworks.calstate.edu/downloads/qr46r081g
3. “IBM Video Developing A Flight Delay Prediction Model Using Machine Learning”
   https://www.youtube.com/watch?v=4PqCK89hJ9Q

APPENDIX

A. Source Code

Flight_Delay_RFC.ipynb

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
data=pd.read_csv('flightdata.csv')

data.head()
data.tail()

#shape of the dataset
data.shape

# Descriptive Analysis
data.describe()
data.dtypes

#missing values
data.isna().sum()

#Dropping null values
data['DEP_TIME'].fillna(data['DEP_TIME'].mean(),inplace=True)
data['DEP_DELAY'].fillna(data['DEP_DELAY'].mean(),inplace=True)
data['DEP_DEL15'].fillna(data['DEP_DEL15'].mean(),inplace=True)
data['ARR_TIME'].fillna(data['ARR_TIME'].mean(),inplace=True)
data['ARR_DELAY'].fillna(data['ARR_DELAY'].mean(),inplace=True)
data['ARR_DEL15'].fillna(data['ARR_DEL15'].mean(),inplace=True)
data['ACTUAL_ELAPSED_TIME'].fillna(data['ACTUAL_ELAPSED_TIME'].mean(),inplace=True)
data.head()

plt.figure(figsize=(10, 6))
sns.countplot(x='ORIGIN', data=data)
plt.xlabel('Origin Airport')
plt.ylabel('Count of Flights')
plt.title('Count of Flights by Origin Airport')
plt.show()
```

```
# Bar Chart - Count of Flights by Destination Airport
plt.figure(figsize=(10, 6))
sns.countplot(x='DEST', data=data)
plt.xlabel('Destination Airport')
plt.ylabel('Count of Flights')
plt.title('Count of Flights by Destination Airport')
plt.show()
sns.scatterplot(x='ARR_DELAY',y='ARR_DEL15',data=data)
sns.catplot(x="ARR_DEL15",y="ARR_DELAY",kind='bar',data=data)

data.drop(columns=['Unnamed:
25','CANCELLED','QUARTER','TAIL_NUM','DISTANCE','DIVERTED','UNIQUE_CARRIER','YEAR','D
EST_AIRPORT_ID','ORIGIN_AIRPORT_ID','CRS_ELAPSED_TIME','ACTUAL_ELAPSED_TIME','DE
P_DELAY','ARR_TIME','ARR_DELAY'],inplace=True)

data.head(35)
print(data[['ORIGIN', 'DEST']].dtypes)
print(data['ORIGIN'].unique())
print(data['DEST'].unique())

data.isnull().any()
data.duplicated()
data=pd.get_dummies(data,columns=['ORIGIN'])
data=pd.get_dummies(data,columns=['DEST'])

x=data.drop('ARR_DEL15',axis=1)
y=data['ARR_DEL15']

from sklearn.preprocessing import StandardScaler
# Create an instance of the StandardScaler
sc = StandardScaler(with_mean=False)
x=sc.fit_transform(x)
x
y=le.fit_transform(y)

data.corr()
plt.figure(figsize=(10,8))
sns.heatmap(data.corr(),annot=True)

from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=0)

from sklearn.ensemble import RandomForestClassifier
```

```
#training the model
rf.fit(x_train,y_train)

#test the model
pred=rf.predict(x_test)
pred

x_train.shape
x_test.shape
y_train.shape
y_test.shape

# Evaluate the model
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
accuracy=accuracy_score(y_test,pred)
conmat=confusion_matrix(y_test,pred)
print(accuracy)
print(conmat)
print(classification_report(y_test,pred))

import pickle
pickle.dump(rf,open("flight.pkl","wb"))
```

flight.py

```
from flask import Flask, render_template, request
import pickle

app = Flask(__name__)
model = pickle.load(open(r'D:\Python\Flask\flight.pkl', 'rb'))

@app.route('/')
def home():
   return render_template("flight.html")

@app.route('/prediction', methods=['POST'])
def 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 == "alt":
    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 == "alt":
    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, int(origin1), int(origin2), int(origin3),
     int(origin4), int(origin5), int(destination1), int(destination2),
     int(destination3), int(destination4), int(destination5), dept, arrtime,
     actdept, dept15]
]

y_pred = model.predict(total)


print(y_pred)



if y_pred[0] == 0:
    ans = "The flight will be on time"
else:
    ans = "The flight will be delayed"
    return render_template("flight.html",showcase=ans)

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

flight.html

```
<html>
<body>
    <p>Welcome to Flight Delays Web Page</p>
    <form action = "/prediction" method= "post">
    <p>Flight Number</p>
    <p><input type="text" name = "name" /></p>

    <p>Month</p>
    <p><input type="text" name = "month" /></p>

    <p>Day of Month</p>
    <p><input type="text" name = "dayofmonth" /></p>

    <p>Day of Week</p>
    <p><input type="text" name = "dayofweek" /></p>

    <label for = "origins">Origin</label>
    <select name ="origin">
    <option Value = "msp">MSP</option>
    <option Value = "dtw">DTW</option>
    <option Value = "jfk">JFK</option>
    <option Value = "sea">SEA</option>
    <option Value = "alt">ALT</option>
    </select>

     <label for = "destinations">Destination</label>
     <select name ="destination">
     <option Value = "msp">MSP</option>
     <option Value = "dtw">DTW</option>
     <option Value = "jfk">JFK</option>
     <option Value = "sea">SEA</option>
     <option Value = "alt">ALT</option>
     </select>

     <p>Scheduled Departure Time</p>
     <p><input type="text" name = "dept" /></p>

     <p>Scheduled Arrival Time</p>
     <p><input type="text" name = "arrtime" /></p>

     <p>Actual Departure Time</p>
     <p><input type="text" name = "actdept" /></p>

    <p><input type="submit" value = "submit" /></p>
    </form>
    <b>{{showcase}}</b>
</body>
</html>
```