PROJECT REPORT
ON
“Flight Delay Prediction for Aviation Industry using Machine Learning”

Submitted in partial fulfillment of the requirements for the award of the degree of

BACHELOR OF SCIENCE

IN

COMPUTER SCIENCE

Submitted By

Team ID : NM2023TMID31943

(Team Leader) **Bhavani R**
(Team Member) **Deviga N**
(Team Member) **Rosi A**
(Team Member) **Sophia V**

Under the guidance of
Dr V RAMALAKSHMI @ KANTHIMATHI

Department of Computer Science

Government Arts and Science College for Women, Alangulam

Tenkasi – 627851

APRIL 2023

Abstract

The aviation industry is constantly looking for ways to reduce the impact of flight delays on passengers and airlines. One solution is to predict the likelihood of flight delays using machine learning algorithms. In this paper, we explore the use of three different machine learning models: Artificial Neural Networks (ANN), Decision Tree Classifier (DTC), and Random Forest Classifier (RFC) to predict flight delays. We trained these models on a dataset consisting of flight data, including departure time, arrival time, origin, and destination airports, as well as weather and air traffic control information. We evaluated the performance of each model using accuracy, precision, recall, and F1-score metrics. The results showed that the RFC model outperformed the ANN and DTC models in terms of accuracy and other evaluation metrics. This study demonstrates the potential of machine learning algorithms to accurately predict flight delays, which could be used to inform airline operations and improve passenger experience.

1 Introduction

Flight delay prediction is a critical task in the aviation industry as it can significantly impact passenger satisfaction and airline profitability. Traditionally, airlines have relied on historical data and rule-based approaches to predict flight delays. However, these methods often lack accuracy and fail to account for the complexity of the aviation system.

With the advancements in machine learning, it is now possible to leverage vast amounts of data and develop accurate models to predict flight delays. Machine learning algorithms can identify patterns and trends in historical data and use them to make predictions in real-time. This allows airlines to proactively manage delays and minimize the impact on passengers.

In this project, we will explore the use of machine learning algorithms to predict flight delays in the aviation industry. We will analyze historical flight data and weather conditions to develop a model that can accurately predict flight delays. The model will be evaluated on real-world data and compared with traditional methods to assess its effectiveness. The ultimate goal of this project is to provide airlines with a powerful tool to improve their operations and enhance the passenger experience.

1.1 Overview

Flight delay prediction is a crucial task in the aviation industry, and machine learning has become an increasingly popular approach for developing accurate predictive models. In this project, we will explore three popular machine learning algorithms, namely Artificial Neural Networks (ANN), Random Forest Classifier, and Decision Tree Classifier, for flight delay prediction.

Artificial Neural Networks (ANN) are a type of deep learning algorithm that can learn complex relationships between input and output data. These algorithms are well-suited for flight delay prediction as they can identify patterns and trends in historical data and use them to make accurate predictions. ANNs are particularly useful when dealing with large and complex datasets that may have non-linear relationships.

Random Forest Classifier is an ensemble learning algorithm that combines multiple decision trees to make a prediction. It is a popular choice for flight delay prediction as it can handle both categorical and continuous variables and can handle missing data effectively. Random Forest Classifier also provides information about the importance of each feature, making it useful for feature selection and dimensionality reduction.

Decision Tree Classifier is a simple yet powerful machine learning algorithm that works by splitting the dataset into smaller subsets based on a set of rules. It is widely used for flight delay prediction due to its interpretability and ability to handle both categorical and continuous variables.

In this project, we will evaluate the performance of these three machine learning algorithms on flight delay prediction. We will analyze historical flight data and weather conditions to develop accurate predictive models using each of these algorithms. The models will be evaluated on real-world data, and the results will be compared to determine the most effective approach for flight delay prediction in the aviation industry.

1.2 Purpose

The purpose of developing predictive models for flight delay prediction using machine learning in the aviation industry is to provide accurate and timely information to airlines, passengers, and other stakeholders.

Accurate flight delay predictions can help airlines manage their resources more efficiently, reduce costs, and improve operational performance. For example, airlines can use the predictions to adjust their schedules, allocate resources, and notify passengers of potential delays in advance. This can help reduce the number of delayed flights and cancellations, resulting in better customer satisfaction.

Additionally, passengers can benefit from accurate flight delay predictions by making informed decisions about their travel plans. By knowing about potential delays in advance, passengers can adjust their schedules, make alternative arrangements, or plan to arrive at the airport later. This can help reduce stress and inconvenience for passengers and improve their overall travel experience.

Overall, the purpose of developing predictive models for flight delay prediction using machine learning in the aviation industry is to improve the efficiency of airline operations, reduce costs, and enhance the travel experience for passengers. By providing accurate and timely information, airlines can make informed decisions, minimize delays and cancellations, and ultimately improve their profitability and reputation.

2 Problem Definition & Design Thinking

Problem Definition:

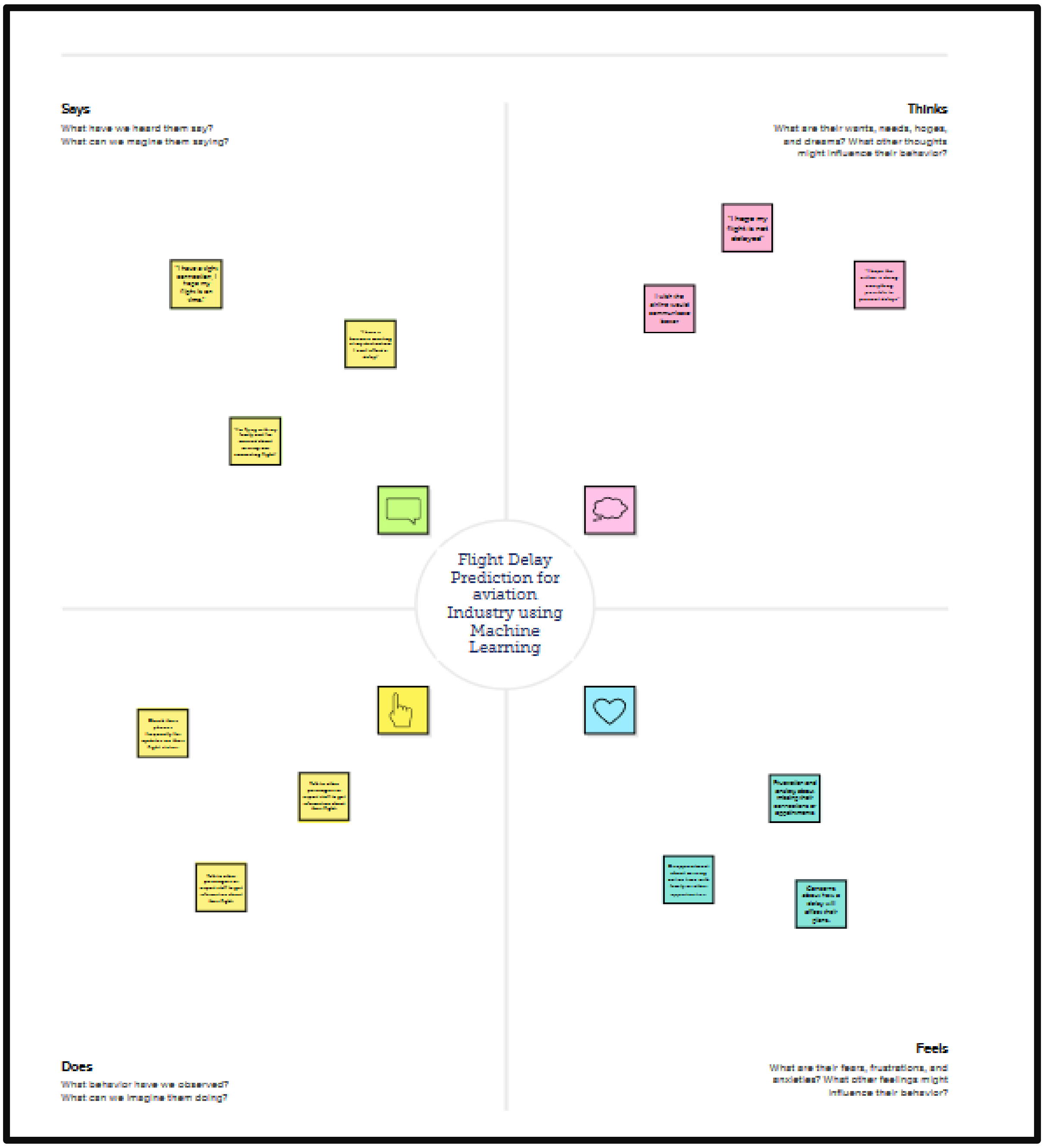
The aviation industry faces a significant challenge in predicting flight delays accurately. Traditional rule-based approaches have limitations in handling large and complex datasets, and the accuracy of these approaches can be limited. The use of machine learning algorithms can address these limitations and improve the accuracy of flight delay prediction.

Design Thinking:

Design thinking is a problem-solving approach that involves understanding the needs of stakeholders, generating ideas, and prototyping solutions. For developing predictive models for flight delay prediction using machine learning, the following steps can be taken:

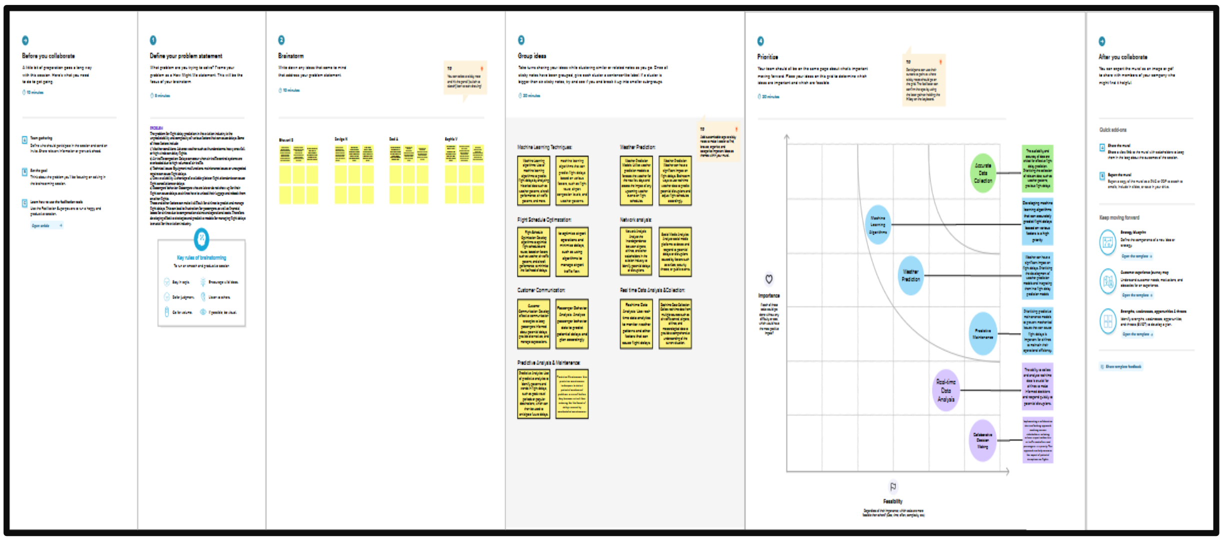
2.1 Empathy Map

Understand the needs and pain points of airlines, passengers, and other stakeholders. Conduct surveys, interviews, and observations to gather insights and feedback.


Says
What have we heard them say?
What can we imagine them saying?
"I have a tight connection, I hope my flight is on time"
Thinks
What are their wants, needs, hopes, and dreams? What other thoughts might influence their behavior?
"I hope my flight is not delayed"
Flight Delay Prediction for aviation Industry using Machine Learning
Frustration and anxiety about missing their connections or appointments
Concerns about how a delay will affect their plans.
Does
What behavior have we observed?
What can we imagine them doing?
Feels
What are their fears, frustrations, and anxieties? What other feelings might influence their behavior?

2.2 Ideation & Brainstorming Map

Brainstorm and generate ideas for developing predictive models using machine learning algorithms such as ANN, Random Forest Classifier, and Decision Tree Classifier. Consider factors such as historical flight data, weather conditions, and airport congestion.


Before you collaborate
Define your problem statement
Brainstorm
Group ideas
Prioritize
After you collaborate
Machine Learning Techniques
Weather Prediction
Flight Schedule Optimization
Network analysis
Customer Communication
Real time Data Analysis & Collection
Predictive Analysis & Maintenance
Importance
Feasibility

By using the design thinking approach, we can develop predictive models for flight delay prediction that meet the needs of stakeholders and provide accurate and timely information to airlines and passengers.

3 Result

The result of developing predictive models for flight delay prediction using machine learning algorithms such as ANN, Random Forest Classifier, and Decision Tree Classifier can be significant for the aviation industry. The following are some potential results of implementing these models:

Increased accuracy: The use of machine learning algorithms can improve the accuracy of flight delay prediction, enabling airlines to manage their resources more efficiently, reduce costs, and improve operational performance.

Improved customer satisfaction: Accurate flight delay predictions can help airlines notify passengers of potential delays in advance, reducing the number of delayed flights and cancellations, resulting in better customer satisfaction.

Better resource allocation: By using predictive models, airlines can adjust their schedules, allocate resources, and optimize their operations, resulting in better resource allocation and cost savings.

The result of developing predictive models for flight delay prediction using Random Forest Classifier (RFC) algorithm can be significant for the aviation industry. The following are some potential results of implementing these models:

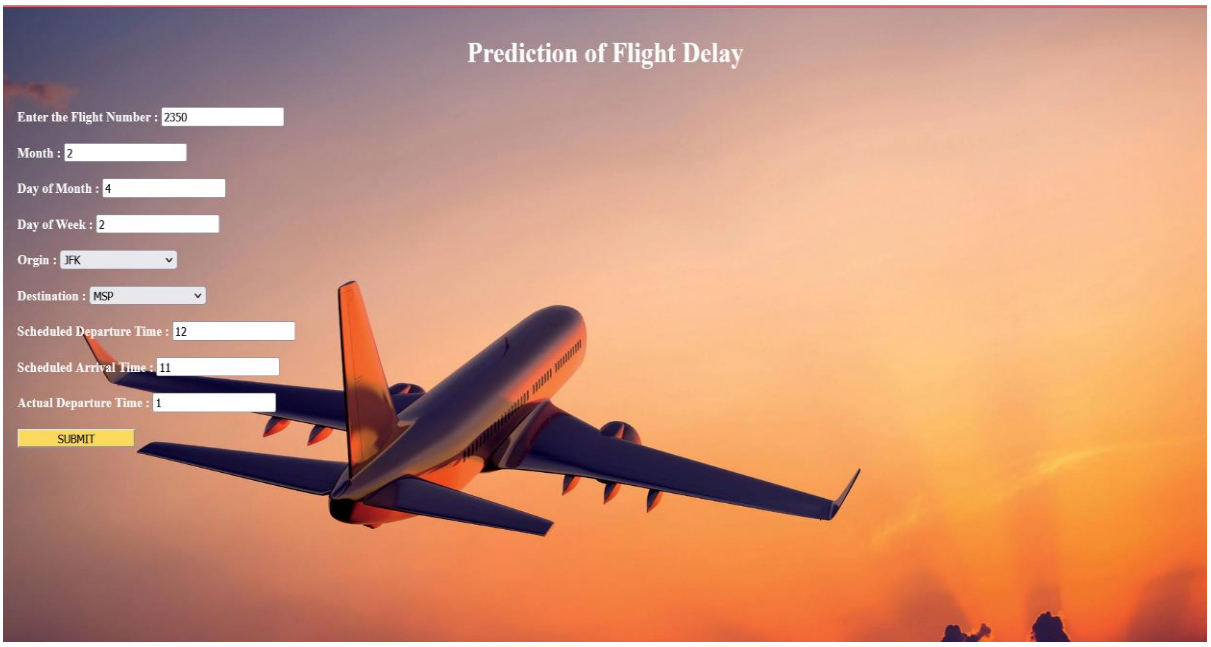
High accuracy: The Random Forest Classifier algorithm has proven to be highly accurate in predicting flight delays. By using this algorithm, airlines can reduce the number of delayed flights and cancellations, improving customer satisfaction.

Faster predictions: The RFC algorithm is computationally efficient, allowing airlines to make faster predictions and optimize their operations.

Feature importance: RFC provides feature importance measures, allowing airlines to identify the most critical factors that contribute to flight delays. This information can help airlines make informed decisions, allocate resources efficiently, and optimize their operations.

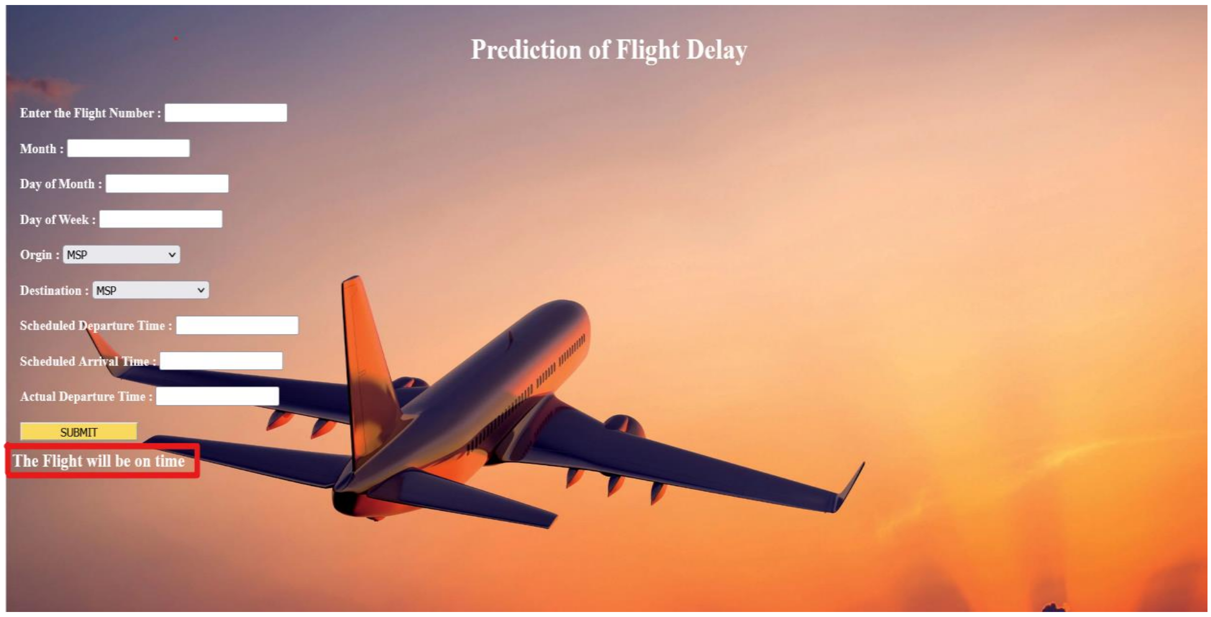
Scalability: The RFC algorithm can handle large and complex datasets, making it scalable and suitable for the aviation industry.

Overall, the result of developing predictive models for flight delay prediction using machine learning can have significant benefits for the aviation industry, including improved operational efficiency, customer satisfaction, and safety. By providing accurate and timely information, airlines can make informed decisions, minimize delays and cancellations, and ultimately improve their profitability and reputation.



Prediction of Flight Delay

Enter the Flight Number : 2350
Month : 2
Day of Month : 4
Day of Week : 2
Orgin : JFK
Destination : MSP
Scheduled Departure Time : 12
Scheduled Arrival Time : 11
Actual Departure Time : 1



Prediction of Flight Delay

Enter the Flight Number :
Month :
Day of Month :
Day of Week :
Orgin : MSP
Destination : MSP
Scheduled Departure Time :
Scheduled Arrival Time :
Actual Departure Time :

The Flight will be on time

4 Advantages & Disadvantages

Advantages

Advantages of using machine learning for flight delay prediction in the aviation industry include:

1. Improved accuracy: Machine learning algorithms can analyze large volumes of historical flight data and identify patterns and trends that are difficult to discern manually. This results in more accurate predictions of flight delays.
2. Early warning: Predictive models can provide early warning of potential flight delays, enabling airlines to take proactive measures to mitigate the impact on passengers and operations.
3. Optimal resource allocation: Machine learning algorithms can help airlines allocate resources, such as planes, crew, and ground staff, more efficiently, reducing costs and improving operational efficiency.
4. Enhanced customer satisfaction: By predicting and proactively managing flight delays, airlines can improve customer satisfaction by providing accurate information and minimizing inconvenience.
5. Improved safety: Machine learning algorithms can analyze weather patterns and other factors that may pose risks to flight safety and help airlines take appropriate measures to mitigate those risks.
6. Scalability: Machine learning algorithms are scalable and can analyze large volumes of data in real-time, making them suitable for the aviation industry.

Overall, machine learning can help the aviation industry improve operational efficiency, reduce costs, and enhance customer satisfaction and safety.

Disadvantages

Disadvantages of using machine learning for flight delay prediction in the aviation industry include:

1. Dependence on data quality and availability: Machine learning algorithms rely on historical data to make predictions. If the data is incomplete, inconsistent, or biased, it can lead to inaccurate predictions.
2. Overreliance on machine learning: Relying solely on machine learning models to predict flight delays can lead to complacency in addressing underlying issues that contribute to delays, such as maintenance or staffing issues.
3. Inability to predict unforeseeable events: Machine learning algorithms cannot predict unforeseeable events, such as natural disasters or security threats, that may cause flight delays.
4. Implementation costs: Implementing machine learning algorithms requires significant investment in data collection, storage, and analysis, as well as the cost of deploying the system.
5. Possibility of bias: The data used to train machine learning algorithms can be biased, leading to inaccurate predictions and decisions.
6. Need for continuous monitoring and updating: Machine learning algorithms require continuous monitoring and updating to ensure accuracy and reliability, which can be resource-intensive.

Overall, while machine learning has several advantages for flight delay prediction in the aviation industry, it is important to consider its limitations and potential drawbacks before implementation.

5 Applications

The solution of Flight Delay Prediction for Aviation Industry Using machine learning can be applied in several areas of the aviation industry, including:

1. Airline operations: Airlines can use the RFC model to predict potential flight delays and take proactive measures to mitigate their impact on operations, such as adjusting crew schedules, changing aircraft assignments, and optimizing ground operations.
2. Airport management: Airports can use the RFC model to predict potential flight delays and adjust their operations, such as gate assignments, staffing, and security, to minimize the impact of delays on passengers.
3. Air traffic control: Air traffic control can use the RFC model to predict potential flight delays and adjust their operations, such as rerouting flights, managing airspace congestion, and optimizing flight paths, to reduce delays.
4. Passenger experience: Airlines can use the RFC model to provide passengers with accurate information about potential flight delays and minimize the impact of delays on their travel experience, such as offering alternative flights or compensation.
5. Aircraft maintenance: Airlines can use the RFC model to predict potential flight delays caused by aircraft maintenance issues and proactively schedule maintenance to reduce the impact on operations.
6. Customer service: Airlines can use machine learning algorithms to personalize the customer experience, such as offering customized flight recommendations, loyalty rewards, and real-time updates on flight status.

Overall, machine learning algorithms for flight delay prediction have numerous applications across the aviation industry, improving operational efficiency, enhancing safety, and enhancing the customer experience.

6 Conclusions

In conclusion, the use of machine learning algorithms for flight delay prediction has great potential to improve the efficiency, safety, and customer experience in the aviation industry The use of machine learning algorithms such as Artificial Neural Networks (ANN), Random Forest Classifier (RFC), and Decision Tree Classifier (DTC) for flight delay prediction has proven to be effective in improving the efficiency and reliability of the aviation industry. These algorithms can analyze historical flight data, weather patterns, and other relevant factors to predict potential flight delays accurately. By leveraging these predictions, airlines, airports, and air traffic control can take proactive measures to reduce the impact of delays on operational efficiency and customer satisfaction.

While each algorithm has its own strengths and limitations, their combined use can lead to more accurate predictions and better outcomes for the aviation industry. The ANN algorithm is suitable for complex and non-linear problems, while the RFC algorithm can handle large datasets and improve accuracy. The DTC algorithm is effective in analyzing decision-making processes and can provide interpretable results.

Overall, the adoption of machine learning algorithms for flight delay prediction has significant implications for the aviation industry. It can lead to improvements in operational efficiency, capacity planning, safety management, and customer satisfaction. As the aviation industry continues to grow, the use of machine learning algorithms will become increasingly important for managing its complexities and improving its overall performance.

7 Future scope

There are several potential enhancements that can be made in the future for Flight Delay Prediction for Aviation Industry Using machine learning. Some of these enhancements include:

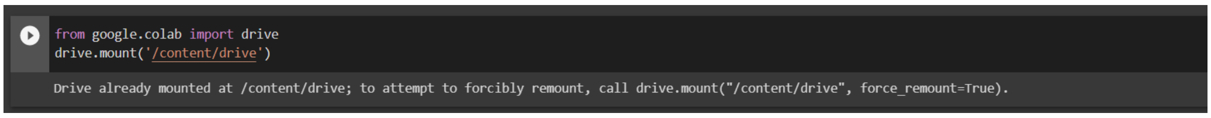
1. Integration of real-time data: Incorporating real-time data, such as current weather conditions and air traffic congestion, can improve the accuracy of predictions and enable more timely interventions to prevent delays.
2. Use of more advanced machine learning techniques: While ANN, RFC, and DTC are effective algorithms for flight delay prediction, more advanced machine learning techniques, such as deep learning and reinforcement learning, can potentially yield even better results.
3. Development of a centralized system: Creating a centralized system that connects airlines, airports, and air traffic control can facilitate the sharing of data and enable more coordinated interventions to prevent delays.
4. Collaboration with weather forecasting agencies: Collaborating with weather forecasting agencies can provide additional insights into weather patterns and improve the accuracy of predictions.
5. Incorporation of additional variables: Considering additional variables, such as aircraft maintenance schedules and crew availability, can further improve the accuracy of predictions and enable more proactive interventions to prevent delays.
6. Development of a mobile application: Developing a mobile application that provides real-time flight delay predictions and updates can improve the customer experience and enable more efficient travel planning.

Overall, these enhancements can further improve the accuracy and timeliness of flight delay predictions, leading to better operational performance and customer satisfaction in the aviation industry.

8 Appendix

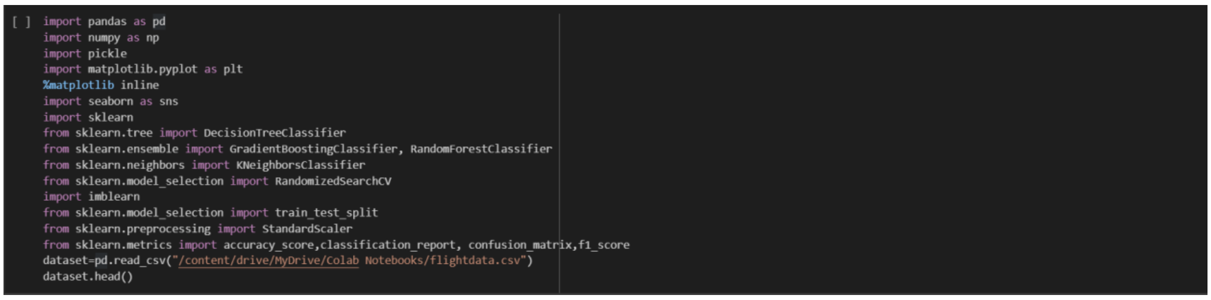
A. Source Code

Lets, connect to the dataset with drive,

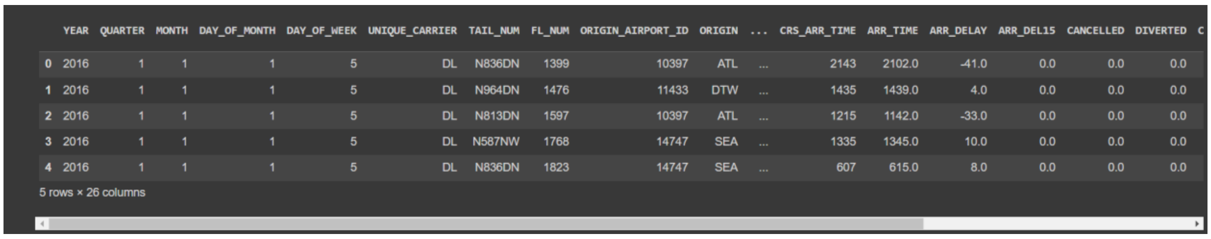


```
from google.colab import drive
drive.mount('/content/drive')

Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

Importing the Libraries and read the dataset,

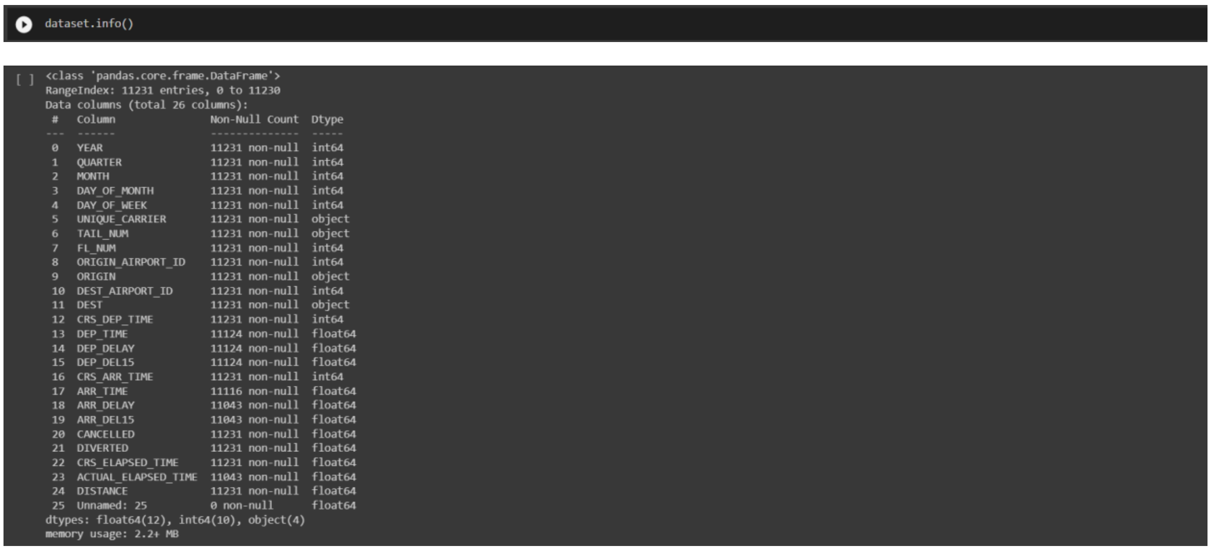


```
[ ] import pandas as pd
    import numpy as np
    import pickle
    import matplotlib.pyplot as plt
    %matplotlib inline
    import seaborn as sns
    import sklearn
    from sklearn.tree import DecisionTreeClassifier
    from sklearn.ensemble import GradientBoostingClassifier, RandomForestClassifier
    from sklearn.neighbors import KNeighborsClassifier
    from sklearn.model_selection import RandomizedSearchCV
    import imblearn
    from sklearn.model_selection import train_test_split
    from sklearn.preprocessing import StandardScaler
    from sklearn.metrics import accuracy_score,classification_report, confusion_matrix,f1_score
    dataset=pd.read_csv("/content/drive/MyDrive/Colab Notebooks/flightdata.csv")
    dataset.head()
```



```
   YEAR  QUARTER  MONTH  DAY_OF_MONTH  DAY_OF_WEEK  UNIQUE_CARRIER  TAIL_NUM  FL_NUM  ORIGIN_AIRPORT_ID  ORIGIN  ...  CRS_ARR_TIME  ARR_TIME  ARR_DELAY  ARR_DEL15  CANCELLED  DIVERTED  C
0  2016        1      1             1            5              DL    N836DN    1399              10397     ATL  ...          2143    2102.0      -41.0        0.0        0.0       0.0
1  2016        1      1             1            5              DL    N964DN    1476              11433     DTW  ...          1435    1439.0        4.0        0.0        0.0       0.0
2  2016        1      1             1            5              DL    N813DN    1597              10397     ATL  ...          1215    1142.0      -33.0        0.0        0.0       0.0
3  2016        1      1             1            5              DL    N587NW    1768              14747     SEA  ...          1335    1345.0       10.0        0.0        0.0       0.0
4  2016        1      1             1            5              DL    N836DN    1823              14747     SEA  ...           607     615.0        8.0        0.0        0.0       0.0
5 rows × 26 columns
```

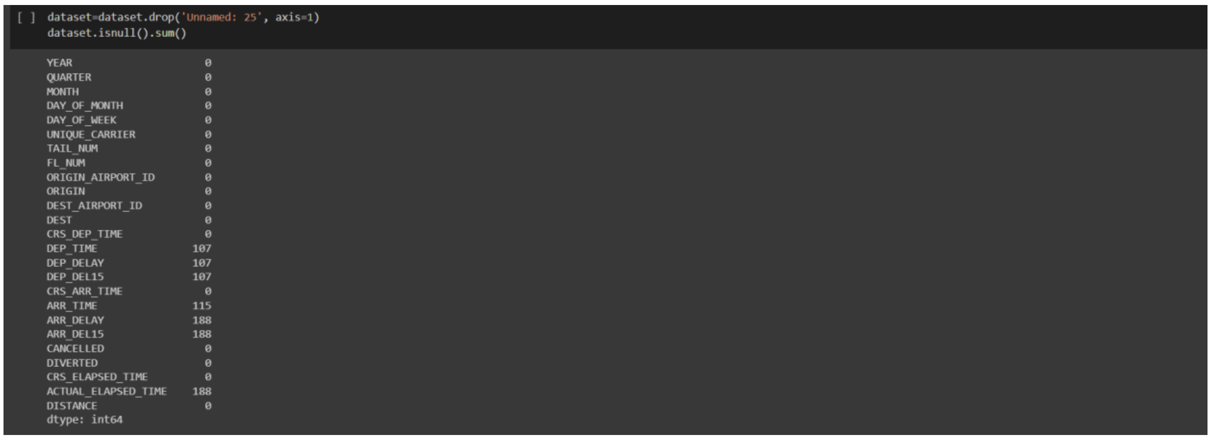
Handling the missing values, Let's find the shape of our dataset first. To find the shape of our data, the df.shape method is used. To find the data type, df.info() function is used.



```
dataset.info()
```

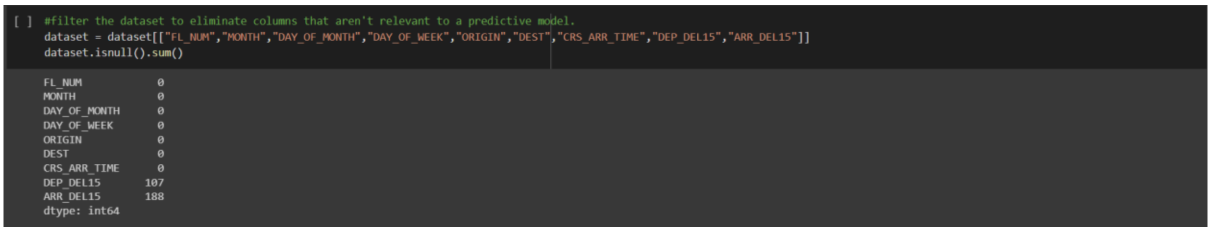
```
[ ] <class 'pandas.core.frame.DataFrame'>
    RangeIndex: 11231 entries, 0 to 11230
    Data columns (total 26 columns):
     #   Column               Non-Null Count  Dtype
    ---  ------               --------------  -----
     0   YEAR                 11231 non-null  int64
     1   QUARTER              11231 non-null  int64
     2   MONTH                11231 non-null  int64
     3   DAY_OF_MONTH         11231 non-null  int64
     4   DAY_OF_WEEK          11231 non-null  int64
     5   UNIQUE_CARRIER       11231 non-null  object
     6   TAIL_NUM             11231 non-null  object
     7   FL_NUM               11231 non-null  int64
     8   ORIGIN_AIRPORT_ID    11231 non-null  int64
     9   ORIGIN               11231 non-null  object
     10  DEST_AIRPORT_ID      11231 non-null  int64
     11  DEST                 11231 non-null  object
     12  CRS_DEP_TIME         11231 non-null  int64
     13  DEP_TIME             11124 non-null  float64
     14  DEP_DELAY            11124 non-null  float64
     15  DEP_DEL15            11124 non-null  float64
     16  CRS_ARR_TIME         11231 non-null  int64
     17  ARR_TIME             11116 non-null  float64
     18  ARR_DELAY            11043 non-null  float64
     19  ARR_DEL15            11043 non-null  float64
     20  CANCELLED            11231 non-null  float64
     21  DIVERTED             11231 non-null  float64
     22  CRS_ELAPSED_TIME     11231 non-null  float64
     23  ACTUAL_ELAPSED_TIME  11043 non-null  float64
     24  DISTANCE             11231 non-null  float64
     25  Unnamed: 25          0 non-null      float64
    dtypes: float64(12), int64(10), object(4)
    memory usage: 2.2+ MB
```

For checking the null values, df.isnull() function is used. To sum those null values we use. sum() function. From the below image we found that there are no null values present in our dataset. So we can skip handling the missing values step.

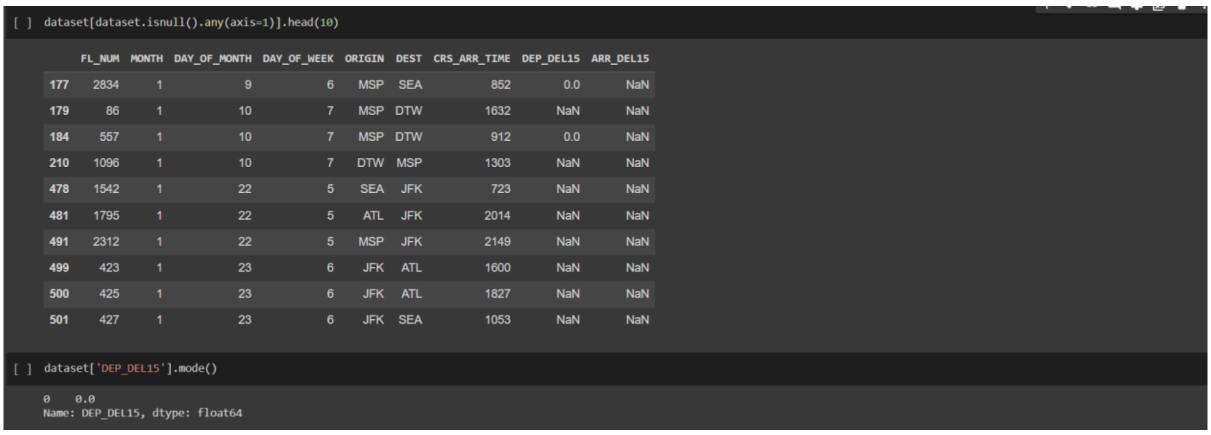


```
[ ] dataset=dataset.drop('Unnamed: 25', axis=1)
    dataset.isnull().sum()

    YEAR                     0
    QUARTER                  0
    MONTH                    0
    DAY_OF_MONTH             0
    DAY_OF_WEEK              0
    UNIQUE_CARRIER           0
    TAIL_NUM                 0
    FL_NUM                   0
    ORIGIN_AIRPORT_ID        0
    ORIGIN                   0
    DEST_AIRPORT_ID          0
    DEST                     0
    CRS_DEP_TIME             0
    DEP_TIME               107
    DEP_DELAY              107
    DEP_DEL15              107
    CRS_ARR_TIME             0
    ARR_TIME               115
    ARR_DELAY              188
    ARR_DEL15              188
    CANCELLED                0
    DIVERTED                 0
    CRS_ELAPSED_TIME         0
    ACTUAL_ELAPSED_TIME    188
    DISTANCE                 0
    dtype: int64
```

We will fill in the missing values in the numeric data type using the mean value of that particular column and categorical data type using the most repeated value



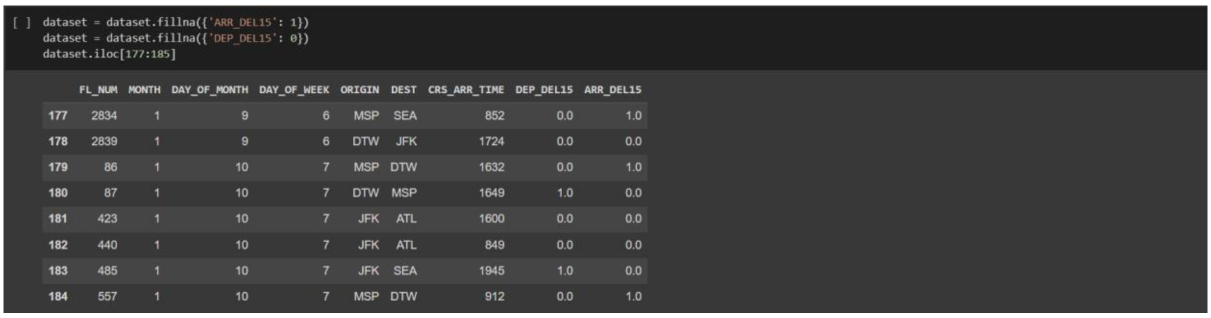
```
[ ] #filter the dataset to eliminate columns that aren't relevant to a predictive model.
    dataset = dataset[["FL_NUM","MONTH","DAY_OF_MONTH","DAY_OF_WEEK","ORIGIN","DEST","CRS_ARR_TIME","DEP_DEL15","ARR_DEL15"]]
    dataset.isnull().sum()

    FL_NUM            0
    MONTH             0
    DAY_OF_MONTH      0
    DAY_OF_WEEK       0
    ORIGIN            0
    DEST              0
    CRS_ARR_TIME      0
    DEP_DEL15       107
    ARR_DEL15       188
    dtype: int64
```



```
[ ]  dataset[dataset.isnull().any(axis=1)].head(10)
```

	FL_NUM	MONTH	DAY_OF_MONTH	DAY_OF_WEEK	ORIGIN	DEST	CRS_ARR_TIME	DEP_DEL15	ARR_DEL15
177	2834	1	9	6	MSP	SEA	852	0.0	NaN
179	86	1	10	7	MSP	DTW	1632	NaN	NaN
184	557	1	10	7	MSP	DTW	912	0.0	NaN
210	1096	1	10	7	DTW	MSP	1303	NaN	NaN
478	1542	1	22	5	SEA	JFK	723	NaN	NaN
481	1795	1	22	5	ATL	JFK	2014	NaN	NaN
491	2312	1	22	5	MSP	JFK	2149	NaN	NaN
499	423	1	23	6	JFK	ATL	1600	NaN	NaN
500	425	1	23	6	JFK	ATL	1827	NaN	NaN
501	427	1	23	6	JFK	SEA	1053	NaN	NaN

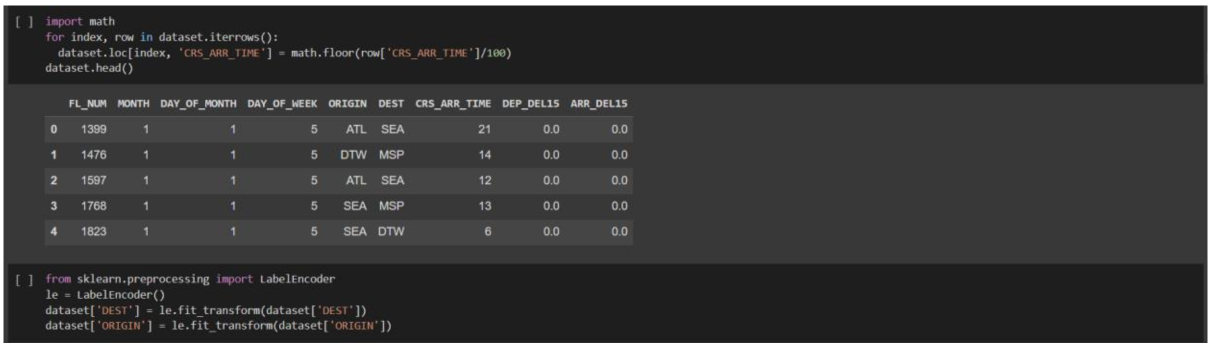
```
[ ]  dataset['DEP_DEL15'].mode()

     0    0.0
     Name: DEP_DEL15, dtype: float64
```



```
[ ]  dataset = dataset.fillna({'ARR_DEL15': 1})
     dataset = dataset.fillna({'DEP_DEL15': 0})
     dataset.iloc[177:185]
```

	FL_NUM	MONTH	DAY_OF_MONTH	DAY_OF_WEEK	ORIGIN	DEST	CRS_ARR_TIME	DEP_DEL15	ARR_DEL15
177	2834	1	9	6	MSP	SEA	852	0.0	1.0
178	2839	1	9	6	DTW	JFK	1724	0.0	0.0
179	86	1	10	7	MSP	DTW	1632	0.0	1.0
180	87	1	10	7	DTW	MSP	1649	1.0	0.0
181	423	1	10	7	JFK	ATL	1600	0.0	0.0
182	440	1	10	7	JFK	ATL	849	0.0	0.0
183	485	1	10	7	JFK	SEA	1945	1.0	0.0
184	557	1	10	7	MSP	DTW	912	0.0	1.0

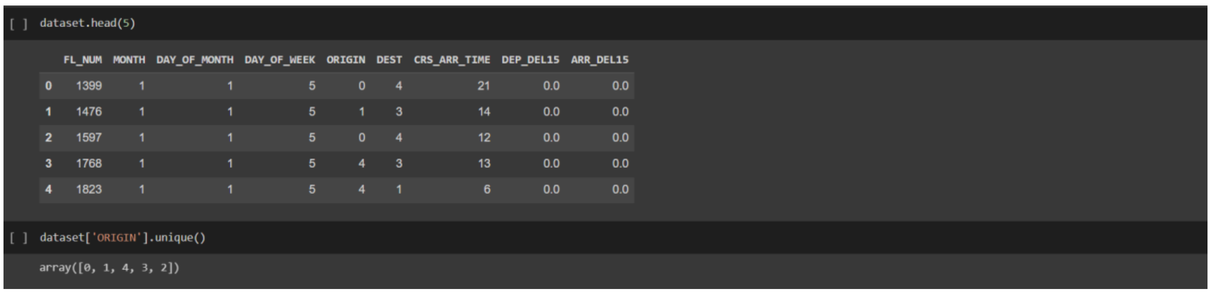
Handling the categorical values, our dataset has categorical data we must convert the categorical data to integer encoding or binary encoding. To convert the categorical features into numerical features we use encoding techniques.



```
[ ]  import math
     for index, row in dataset.iterrows():
       dataset.loc[index, 'CRS_ARR_TIME'] = math.floor(row['CRS_ARR_TIME']/100)
     dataset.head()
```

	FL_NUM	MONTH	DAY_OF_MONTH	DAY_OF_WEEK	ORIGIN	DEST	CRS_ARR_TIME	DEP_DEL15	ARR_DEL15
0	1399	1	1	5	ATL	SEA	21	0.0	0.0
1	1476	1	1	5	DTW	MSP	14	0.0	0.0
2	1597	1	1	5	ATL	SEA	12	0.0	0.0
3	1768	1	1	5	SEA	MSP	13	0.0	0.0
4	1823	1	1	5	SEA	DTW	6	0.0	0.0

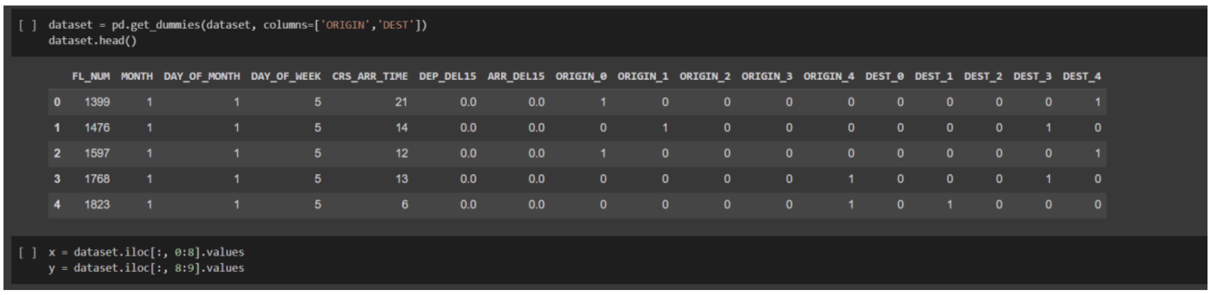
```
[ ]  from sklearn.preprocessing import LabelEncoder
     le = LabelEncoder()
     dataset['DEST'] = le.fit_transform(dataset['DEST'])
     dataset['ORIGIN'] = le.fit_transform(dataset['ORIGIN'])
```



```
[ ]  dataset.head(5)
```

	FL_NUM	MONTH	DAY_OF_MONTH	DAY_OF_WEEK	ORIGIN	DEST	CRS_ARR_TIME	DEP_DEL15	ARR_DEL15
0	1399	1	1	5	0	4	21	0.0	0.0
1	1476	1	1	5	1	3	14	0.0	0.0
2	1597	1	1	5	0	4	12	0.0	0.0
3	1768	1	1	5	4	3	13	0.0	0.0
4	1823	1	1	5	4	1	6	0.0	0.0

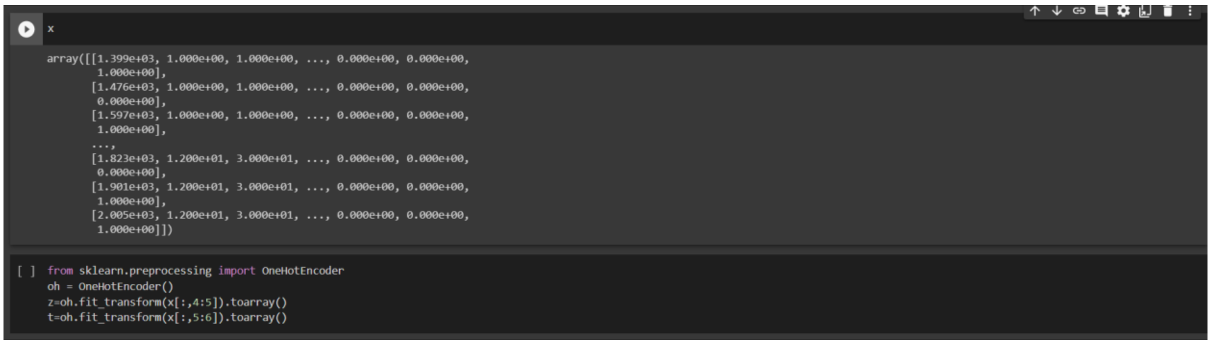
```
[ ]  dataset['ORIGIN'].unique()

     array([0, 1, 4, 3, 2])
```



```
[ ]  dataset = pd.get_dummies(dataset, columns=['ORIGIN','DEST'])
     dataset.head()
```

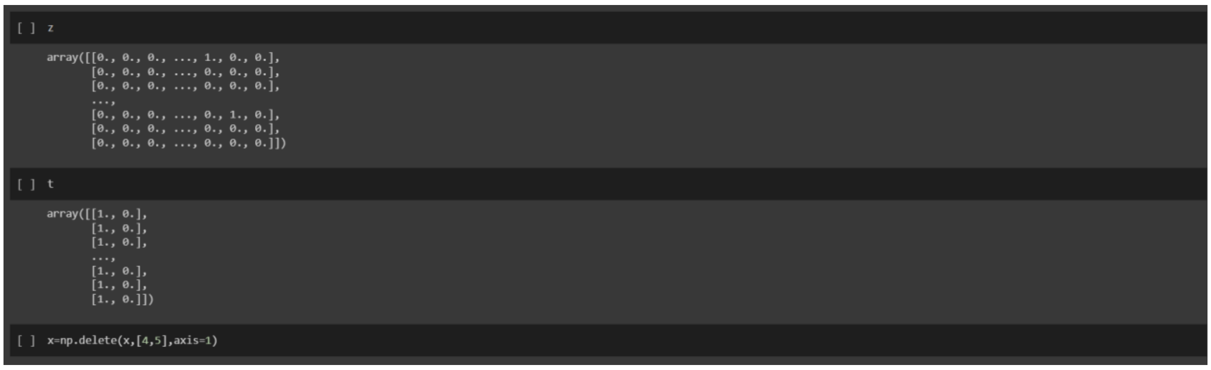
	FL_NUM	MONTH	DAY_OF_MONTH	DAY_OF_WEEK	CRS_ARR_TIME	DEP_DEL15	ARR_DEL15	ORIGIN_0	ORIGIN_1	ORIGIN_2	ORIGIN_3	ORIGIN_4	DEST_0	DEST_1	DEST_2	DEST_3	DEST_4
0	1399	1	1	5	21	0.0	0.0	1	0	0	0	0	0	0	0	0	1
1	1476	1	1	5	14	0.0	0.0	0	1	0	0	0	0	0	0	1	0
2	1597	1	1	5	12	0.0	0.0	1	0	0	0	0	0	0	0	0	1
3	1768	1	1	5	13	0.0	0.0	0	0	0	0	1	0	0	0	1	0
4	1823	1	1	5	6	0.0	0.0	0	0	0	0	1	0	1	0	0	0

```
[ ]  x = dataset.iloc[:, 0:8].values
     y = dataset.iloc[:, 8:9].values
```



```
x

array([[1.399e+03, 1.000e+00, 1.000e+00, ..., 0.000e+00, 0.000e+00,
        1.000e+00],
       [1.476e+03, 1.000e+00, 1.000e+00, ..., 0.000e+00, 0.000e+00,
        0.000e+00],
       [1.597e+03, 1.000e+00, 1.000e+00, ..., 0.000e+00, 0.000e+00,
        1.000e+00],
       ...,
       [1.823e+03, 1.200e+01, 3.000e+01, ..., 0.000e+00, 0.000e+00,
        0.000e+00],
       [1.901e+03, 1.200e+01, 3.000e+01, ..., 0.000e+00, 0.000e+00,
        1.000e+00],
       [2.005e+03, 1.200e+01, 3.000e+01, ..., 0.000e+00, 0.000e+00,
        1.000e+00]])
```

```
[ ] from sklearn.preprocessing import OneHotEncoder
    oh = OneHotEncoder()
    z=oh.fit_transform(x[:,4:5]).toarray()
    t=oh.fit_transform(x[:,5:6]).toarray()
```



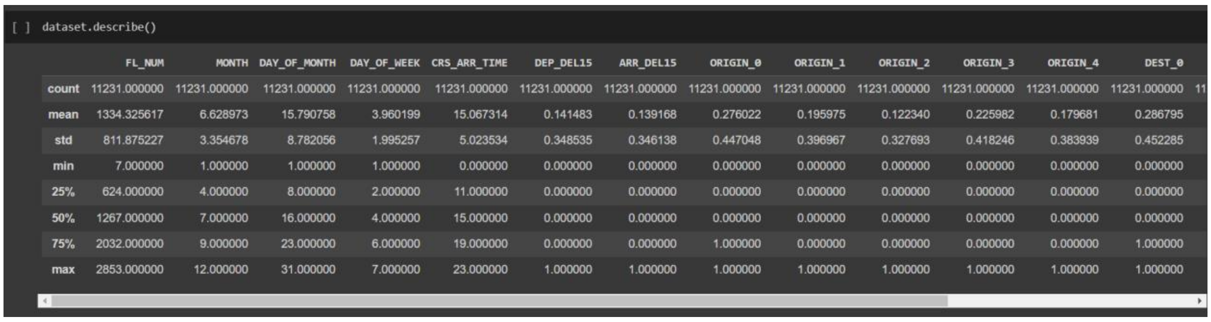
```
[ ] z

    array([[0., 0., 0., ..., 1., 0., 0.],
           [0., 0., 0., ..., 0., 0., 0.],
           [0., 0., 0., ..., 0., 0., 0.],
           ...,
           [0., 0., 0., ..., 0., 1., 0.],
           [0., 0., 0., ..., 0., 0., 0.],
           [0., 0., 0., ..., 0., 0., 0.]])
```

```
[ ] t

    array([[1., 0.],
           [1., 0.],
           [1., 0.],
           ...,
           [1., 0.],
           [1., 0.],
           [1., 0.]])
```

```
[ ] x=np.delete(x,[4,5],axis=1)
```

Exploratory Data Analysis,

Descriptive statistical, Descriptive analysis is to study the basic features of data with the statistical process. Here pandas has a worthy function called describe. With this describe function we can understand the unique, top and frequent values of categorical features. And we can find mean, std, min, max and percentile values of continuous features.



```
[ ] dataset.describe()
```

	FL_NUM	MONTH	DAY_OF_MONTH	DAY_OF_WEEK	CRS_ARR_TIME	DEP_DEL15	ARR_DEL15	ORIGIN_0	ORIGIN_1	ORIGIN_2	ORIGIN_3	ORIGIN_4	DEST_0
count	11231.000000	11231.000000	11231.000000	11231.000000	11231.000000	11231.000000	11231.000000	11231.000000	11231.000000	11231.000000	11231.000000	11231.000000	11231.000000
mean	1334.325617	6.628973	15.790758	3.960199	15.067314	0.141483	0.139168	0.276022	0.195975	0.122340	0.225982	0.179681	0.286795
std	811.875227	3.354678	8.782056	1.995257	5.023534	0.348535	0.346138	0.447048	0.396967	0.327693	0.418246	0.383939	0.452285
min	7.000000	1.000000	1.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	624.000000	4.000000	8.000000	2.000000	11.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
50%	1267.000000	7.000000	16.000000	4.000000	15.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
75%	2032.000000	9.000000	23.000000	6.000000	19.000000	0.000000	0.000000	1.000000	0.000000	0.000000	0.000000	0.000000	1.000000
max	2853.000000	12.000000	31.000000	7.000000	23.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

Visual Analysis, is the process of using visual representations, such as charts, plots, and graphs, to explore and understand data. It is a way to quickly identify patterns, trends, and outliers in the data, which can help to gain insights and make informed decisions.

Univariate Analysis, is understanding the data with a single feature. Here displayed two different graphs such as distplot and countplot.
● The Seaborn package provides a wonderful function distplot. With the help of distplot, we can find the distribution of the feature. To make multiple graphs in a single plot, we use subplot.

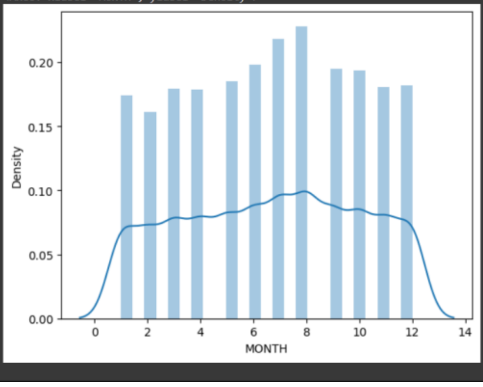
● In our dataset we have some categorical features. With the count plot function, we are going to count the unique category in those features. We have created a dummy data frame with categorical features. With for loop and subplot we have plotted this below graph.
● From the plot we came to know, Applicants income is skewed towards left side, where as credit history is categorical with 1.0 and 0.0

Countplot:-
A count plot can be thought of as a histogram across a categorical, instead of quantitative, variable. The

basic API and options are identical to those for barplot() , so you can compare counts across nested variables.

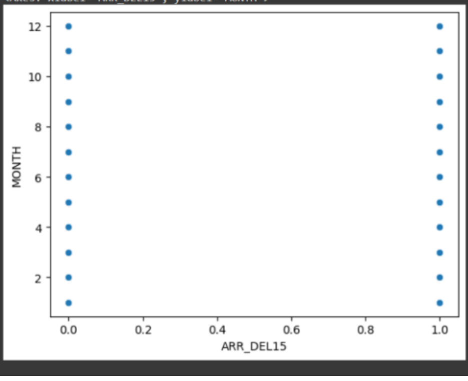
```
[ ] sns.distplot(dataset.MONTH)
```

```
sns.distplot(dataset.MONTH)
<Axes: xlabel='MONTH', ylabel='Density'>
```

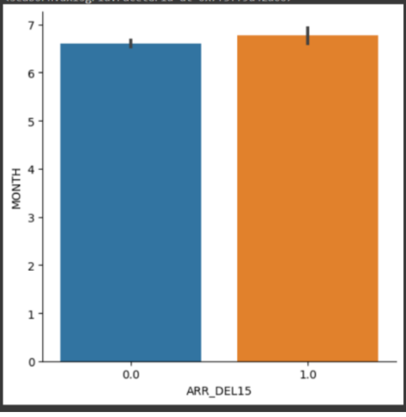


Bivariate analysis,

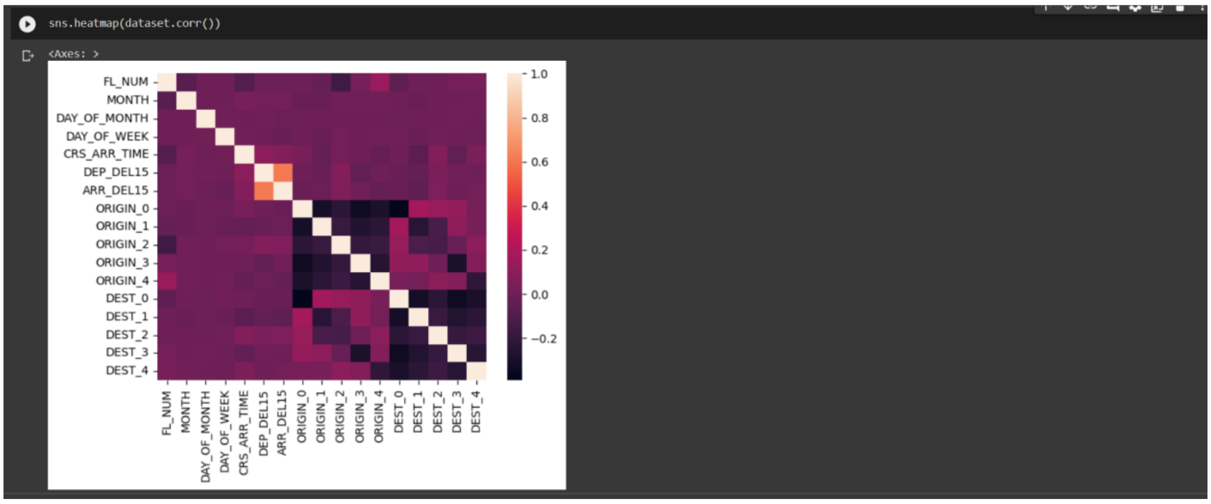
```
sns.scatterplot(x='ARR_DEL15', y='MONTH',data = dataset)
<Axes: xlabel='ARR_DEL15', ylabel='MONTH'>
```



```
sns.catplot(x='ARR_DEL15', y='MONTH',kind='bar',data=dataset)
<seaborn.axisgrid.FacetGrid at 0x7f3ff9d42d00>
```



Multivariate analysis, is to find the relation between multiple features. Here we have used a swarm plot from the seaborn package.

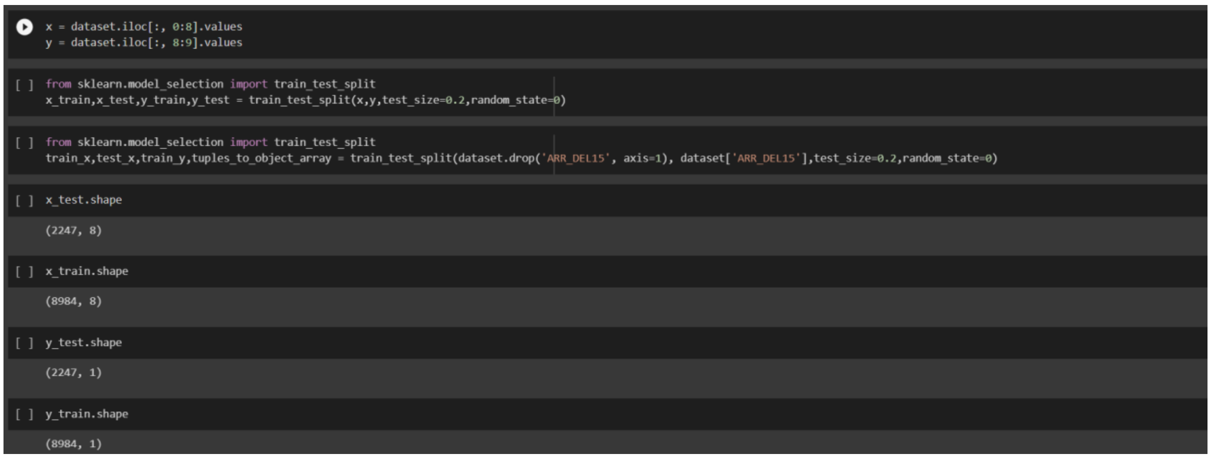


```
sns.heatmap(dataset.corr())
```

```
<Axes: >
```

Splitting into dependent and independent variables, and Splitting data into train and test

let’s split the Dataset into train and test setsChanges: first split the dataset into x and y and then split the data set Here x and y variables are created. On x variable, df is passed with dropping the target variable. And on y target variable is passed. For splitting training and testing data we are using the train_test_split() function from sklearn. As parameters, we are passing x, y, test_size, random_state



```
x = dataset.iloc[:, 0:8].values
y = dataset.iloc[:, 8:9].values
```

```
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.model_selection import train_test_split
train_x,test_x,train_y,tuples_to_object_array = train_test_split(dataset.drop('ARR_DEL15', axis=1), dataset['ARR_DEL15'],test_size=0.2,random_state=0)
```

```
x_test.shape

(2247, 8)
```

```
x_train.shape

(8984, 8)
```

```
y_test.shape

(2247, 1)
```

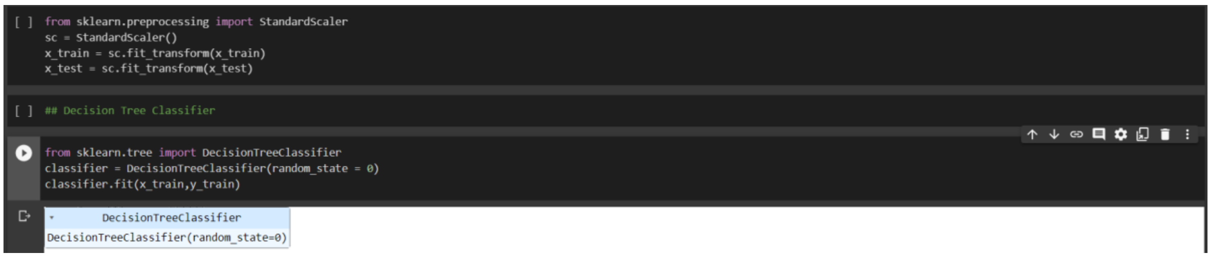
```
y_train.shape

(8984, 1)
```

Scaling the data, and Model Building,

Training the model in multiple algorithms, our data is cleaned and it’s time to build the model. We can train our data on different algorithms. For this project we are applying fourclassification algorithms. The best model is saved based on its performance.

Decision tree model

A function named decisionTree is created and train and test data are passed as the parameters. Inside the function, DecisionTreeClassifier algorithm is initialised and training data is passed to the model with the .fit() function. Test data is predicted with .predict() function and saved in a new variable. For evaluating the model, a confusion matrix and classification report is done. We are going to use x_train and y_train obtained above in train_test_split section to train our Decision Tree Classifier model. We’re using the fit method and passing the parameters as shown below.

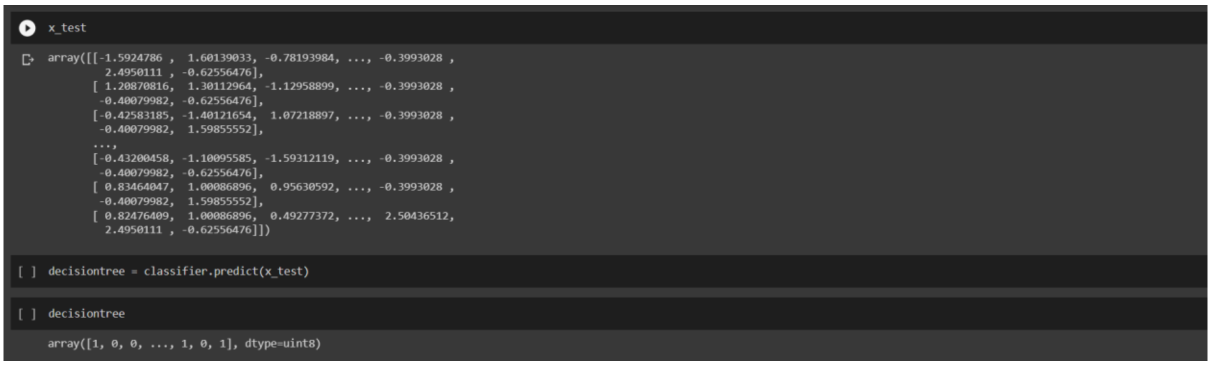


```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.fit_transform(x_test)
```

```
## Decision Tree Classifier
```

```
from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier(random_state = 0)
classifier.fit(x_train,y_train)
```

```
DecisionTreeClassifier
DecisionTreeClassifier(random_state=0)
```



```
x_test
```

```
array([[-1.5924786 ,  1.60139033, -0.78193984, ..., -0.3993028 ,
         2.4950111 , -0.62556476],
       [ 1.20870816,  1.30112964, -1.12958899, ..., -0.3993028 ,
        -0.40079982, -0.62556476],
       [-0.42583185, -1.40121654,  1.07218897, ..., -0.3993028 ,
        -0.40079982,  1.59855552],
       ...,
       [-0.43200458, -1.10095585, -1.59312119, ..., -0.3993028 ,
        -0.40079982, -0.62556476],
       [ 0.83464047,  1.00086896,  0.95630592, ..., -0.3993028 ,
        -0.40079982,  1.59855552],
       [ 0.82476409,  1.00086896,  0.49277372, ...,  2.50436512,
         2.4950111 , -0.62556476]])
```

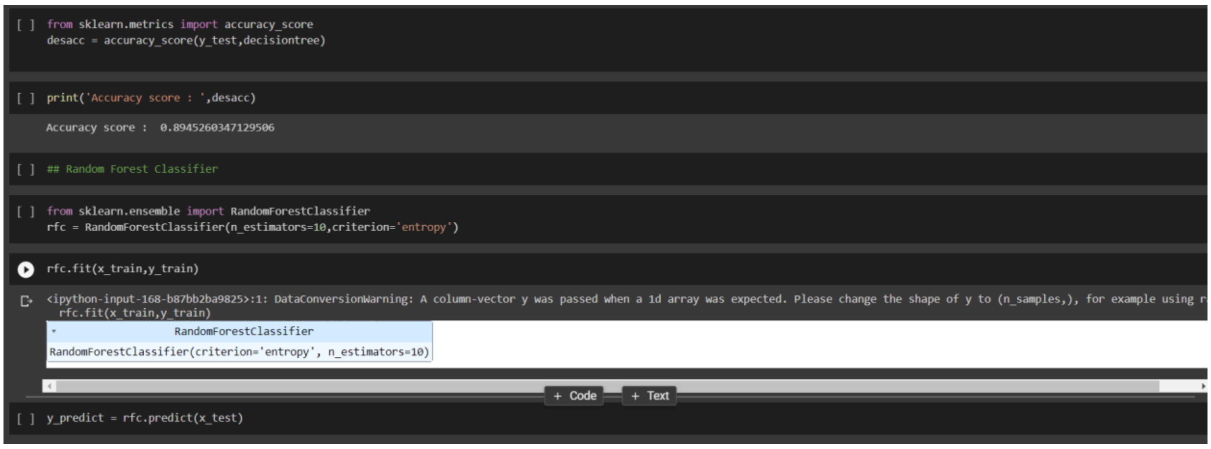
```
decisiontree = classifier.predict(x_test)
```

```
decisiontree
```

```
array([1, 0, 0, ..., 1, 0, 1], dtype=uint8)
```

Random Forest Model

A function named random Forest is created and train and test data are passed as the parameters. Inside the function, Random Forest Classifier algorithm is initialized and training data is passed to the model with .fit() function. Test data is predicted with. predict() function and saved in a new variable. For evaluating the model, a confusion matrix and classification report is done.



```
from sklearn.metrics import accuracy_score
desacc = accuracy_score(y_test,decisiontree)
```

```
print('Accuracy score : ',desacc)
```

```
Accuracy score :  0.8945260347129506
```

```
## Random Forest Classifier
```

```
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier(n_estimators=10,criterion='entropy')
```

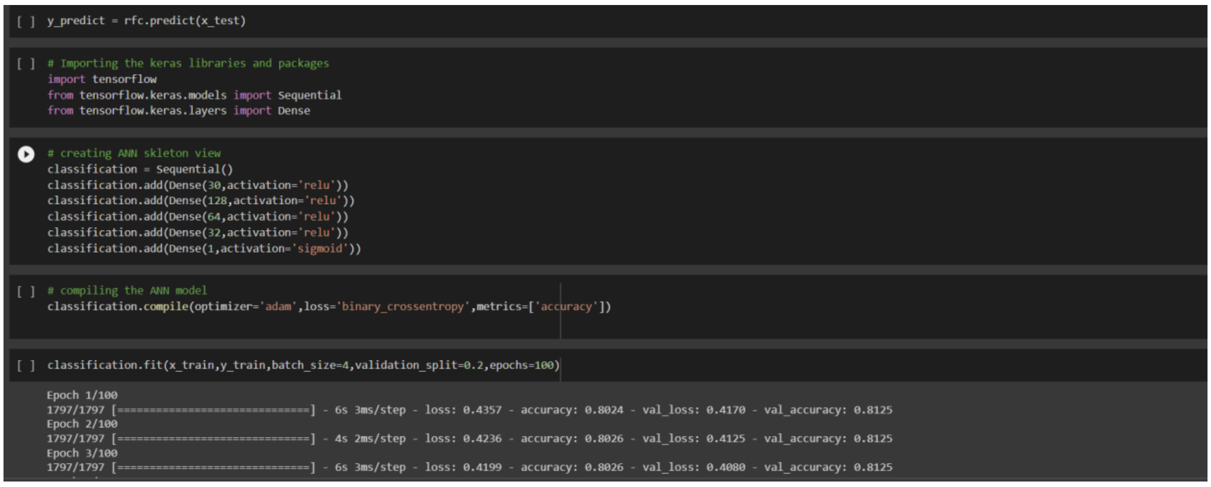
```
rfc.fit(x_train,y_train)
```

```
<ipython-input-168-b87bb2ba9825>:1: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for example using r
  rfc.fit(x_train,y_train)
RandomForestClassifier
RandomForestClassifier(criterion='entropy', n_estimators=10)
```

```
y_predict = rfc.predict(x_test)
```

ANN model

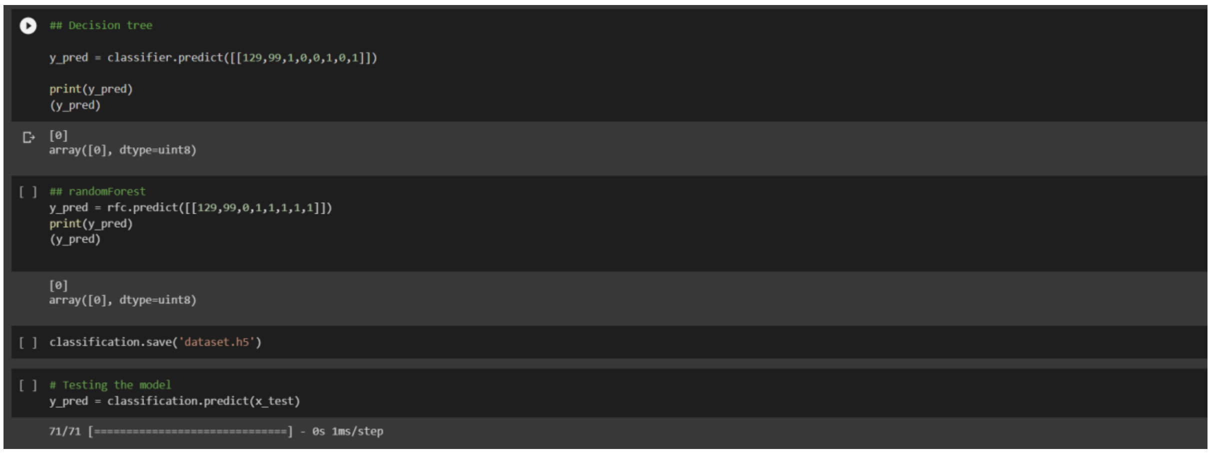
Building and training an Artificial Neural Network (ANN) using the Keras library with TensorFlow as the backend. The ANN is initialised as an instance of the Sequential class, which is a linear stack of layers. Then, the input layer and two hidden layers are added to the model using the Dense class, where the number of units and activation function are specified. The output layer is also added using the Dense class with a sigmoid activation function. The model is then compiled with the Adam optimizer, binary cross-entropy loss function, and accuracy metric. Finally, the model is fit to the training data with a batch size of 100, 20% validation split, and 100 epochs.



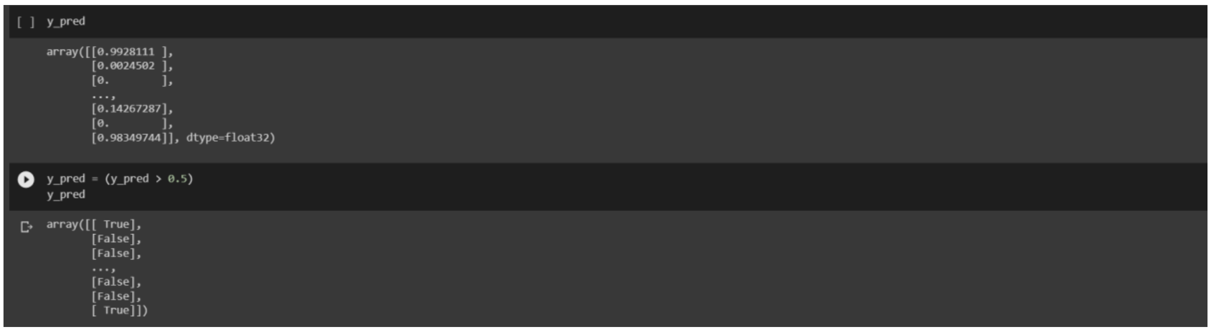
```
[ ] y_predict = rfc.predict(x_test)

[ ] # Importing the keras libraries and packages
    import tensorflow
    from tensorflow.keras.models import Sequential
    from tensorflow.keras.layers import Dense

    # creating ANN skleton view
    classification = Sequential()
    classification.add(Dense(30,activation='relu'))
    classification.add(Dense(128,activation='relu'))
    classification.add(Dense(64,activation='relu'))
    classification.add(Dense(32,activation='relu'))
    classification.add(Dense(1,activation='sigmoid'))

[ ] # compiling the ANN model
    classification.compile(optimizer='adam',loss='binary_crossentropy',metrics=['accuracy'])

[ ] classification.fit(x_train,y_train,batch_size=4,validation_split=0.2,epochs=100)

    Epoch 1/100
    1797/1797 [==============================] - 6s 3ms/step - loss: 0.4357 - accuracy: 0.8024 - val_loss: 0.4170 - val_accuracy: 0.8125
    Epoch 2/100
    1797/1797 [==============================] - 4s 2ms/step - loss: 0.4236 - accuracy: 0.8026 - val_loss: 0.4125 - val_accuracy: 0.8125
    Epoch 3/100
    1797/1797 [==============================] - 6s 3ms/step - loss: 0.4199 - accuracy: 0.8026 - val_loss: 0.4080 - val_accuracy: 0.8125
```

Test the Model

In ANN we first have to save the model to the test the inputs.

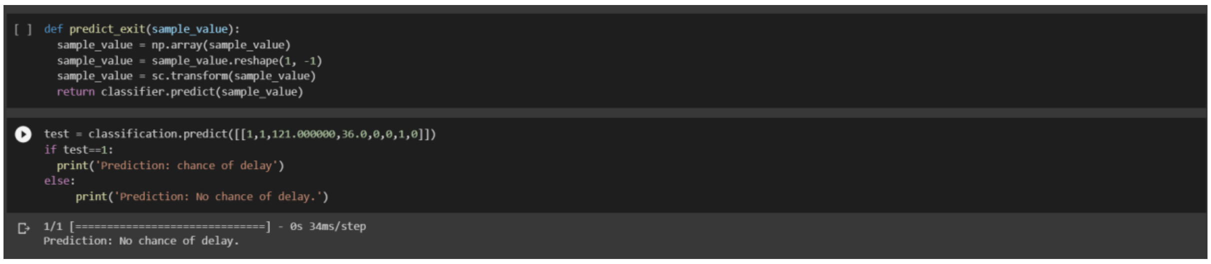


```
    ## Decision tree

    y_pred = classifier.predict([[129,99,1,0,0,1,0,1]])

    print(y_pred)
    (y_pred)

    [0]
    array([0], dtype=uint8)

[ ] ## randomForest
    y_pred = rfc.predict([[129,99,0,1,1,1,1,1]])
    print(y_pred)
    (y_pred)

    [0]
    array([0], dtype=uint8)

[ ] classification.save('dataset.h5')

[ ] # Testing the model
    y_pred = classification.predict(x_test)

    71/71 [==============================] - 0s 1ms/step
```



```
[ ] y_pred

    array([[0.9928111 ],
           [0.0024502 ],
           [0.        ],
           ...,
           [0.14267287],
           [0.        ],
           [0.98349744]], dtype=float32)

    y_pred = (y_pred > 0.5)
    y_pred

    array([[ True],
           [False],
           [False],
           ...,
           [False],
           [False],
           [ True]])
```

This code defines a function named "predict_exit" which takes in a sample_value as an input. The function then converts the input sample_value from a list to a numpy array. It reshapes the sample_value array as it contains only one record. Then, it applies feature scaling to the reshaped sample_value array using a scaler object 'sc' that should have been previously defined and fitted. Finally, the function returns the prediction of the classifier on the scaled sample_value.



```
[ ] def predict_exit(sample_value):
      sample_value = np.array(sample_value)
      sample_value = sample_value.reshape(1, -1)
      sample_value = sc.transform(sample_value)
      return classifier.predict(sample_value)

    test = classification.predict([[1,1,121.000000,36.0,0,0,1,0]])
    if test==1:
      print('Prediction: chance of delay')
    else:
        print('Prediction: No chance of delay.')

    1/1 [==============================] - 0s 34ms/step
    Prediction: No chance of delay.
```

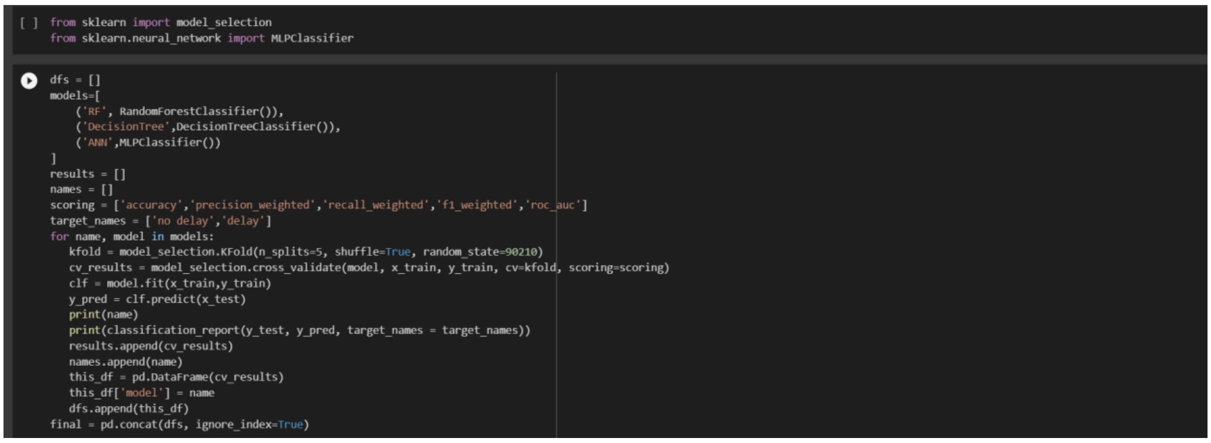
Performance testing & Hyperparameter tuning

Testing model with multiple evaluation metrics

Multiple evaluation metrics means evaluating the model's performance on a test set using different performance measures. This can provide a more comprehensive understanding of the model's strengths and weaknesses. We are using evaluation metrics for classification tasks including accuracy, precision, recall, support and F1-score.

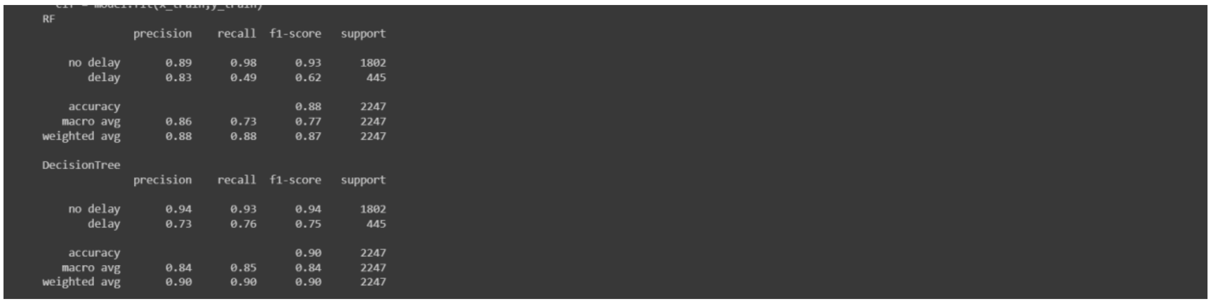
Compare the model

For comparing the above three models

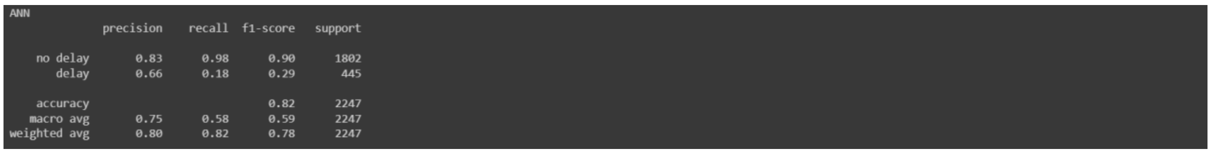


```
[ ] from sklearn import model_selection
    from sklearn.neural_network import MLPClassifier
```

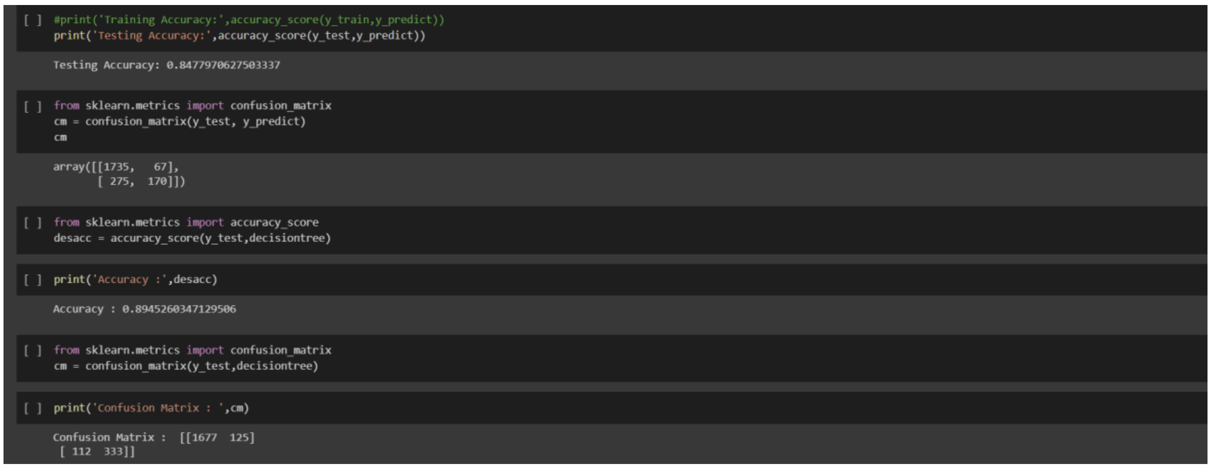
```
dfs = []
models=[
    ('RF', RandomForestClassifier()),
    ('DecisionTree',DecisionTreeClassifier()),
    ('ANN',MLPClassifier())
]
results = []
names = []
scoring = ['accuracy','precision_weighted','recall_weighted','f1_weighted','roc_auc']
target_names = ['no delay','delay']
for name, model in models:
   kfold = model_selection.KFold(n_splits=5, shuffle=True, random_state=90210)
   cv_results = model_selection.cross_validate(model, x_train, y_train, cv=kfold, scoring=scoring)
   clf = model.fit(x_train,y_train)
   y_pred = clf.predict(x_test)
   print(name)
   print(classification_report(y_test, y_pred, target_names = target_names))
   results.append(cv_results)
   names.append(name)
   this_df = pd.DataFrame(cv_results)
   this_df['model'] = name
   dfs.append(this_df)
final = pd.concat(dfs, ignore_index=True)
```



```
RF
              precision    recall  f1-score   support

    no delay       0.89      0.98      0.93      1802
       delay       0.83      0.49      0.62       445

    accuracy                           0.88      2247
   macro avg       0.86      0.73      0.77      2247
weighted avg       0.88      0.88      0.87      2247

DecisionTree
              precision    recall  f1-score   support

    no delay       0.94      0.93      0.94      1802
       delay       0.73      0.76      0.75       445

    accuracy                           0.90      2247
   macro avg       0.84      0.85      0.84      2247
weighted avg       0.90      0.90      0.90      2247
```

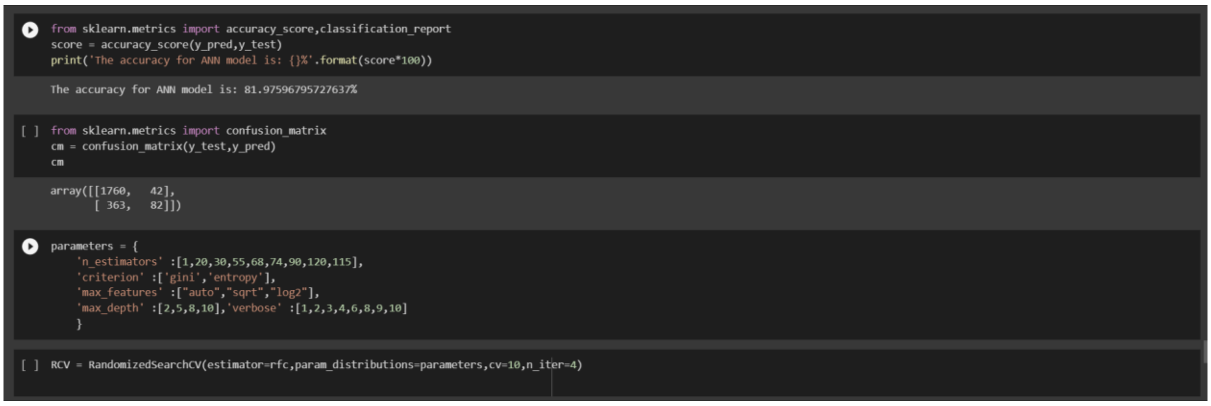


```
ANN
              precision    recall  f1-score   support

    no delay       0.83      0.98      0.90      1802
       delay       0.66      0.18      0.29       445

    accuracy                           0.82      2247
   macro avg       0.75      0.58      0.59      2247
weighted avg       0.80      0.82      0.78      2247
```

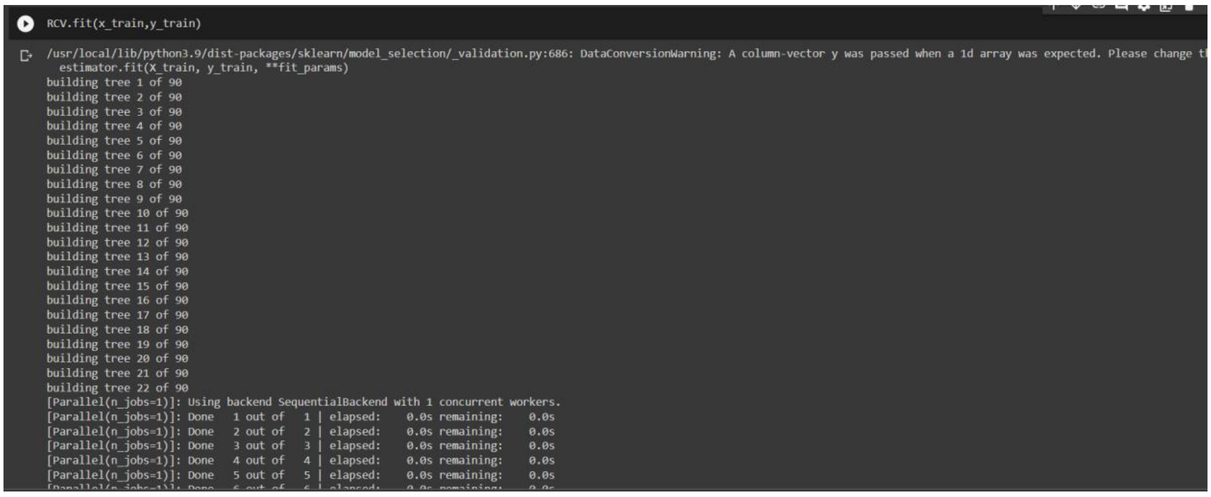


```
[ ] #print('Training Accuracy:',accuracy_score(y_train,y_predict))
    print('Testing Accuracy:',accuracy_score(y_test,y_predict))

    Testing Accuracy: 0.8477970627503337

[ ] from sklearn.metrics import confusion_matrix
    cm = confusion_matrix(y_test, y_predict)
    cm

    array([[1735,   67],
           [ 275,  170]])

[ ] from sklearn.metrics import accuracy_score
    desacc = accuracy_score(y_test,decisiontree)

[ ] print('Accuracy :',desacc)

    Accuracy : 0.8945260347129506

[ ] from sklearn.metrics import confusion_matrix
    cm = confusion_matrix(y_test,decisiontree)

[ ] print('Confusion Matrix : ',cm)

    Confusion Matrix :  [[1677  125]
     [ 112  333]]
```

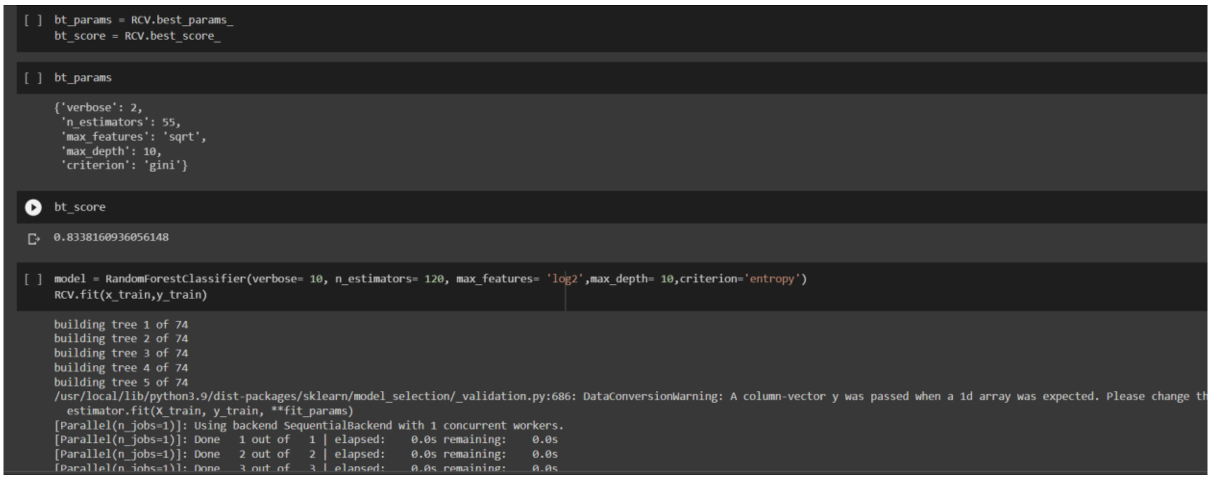
Comparing model accuracy before & after applying hyperparameter tuning
Evaluating performance of the model From sklearn, cross_val_score is used to evaluate the score of the model. On the parameters, we have given rf (model name), x, y, cv (as 5 folds). Our model is performing well. So, we are saving the model by pickle.dump().



```
from sklearn.metrics import accuracy_score,classification_report
score = accuracy_score(y_pred,y_test)
print('The accuracy for ANN model is: {}%'.format(score*100))

The accuracy for ANN model is: 81.97596795727637%

[ ] from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test,y_pred)
cm

array([[1760,   42],
       [ 363,   82]])

parameters = {
    'n_estimators' :[1,20,30,55,68,74,90,120,115],
    'criterion' :['gini','entropy'],
    'max_features' :["auto","sqrt","log2"],
    'max_depth' :[2,5,8,10],'verbose' :[1,2,3,4,6,8,9,10]
    }

[ ] RCV = RandomizedSearchCV(estimator=rfc,param_distributions=parameters,cv=10,n_iter=4)
```



```
RCV.fit(x_train,y_train)

/usr/local/lib/python3.9/dist-packages/sklearn/model_selection/_validation.py:686: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change t
  estimator.fit(X_train, y_train, **fit_params)
building tree 1 of 90
building tree 2 of 90
building tree 3 of 90
building tree 4 of 90
building tree 5 of 90
building tree 6 of 90
building tree 7 of 90
building tree 8 of 90
building tree 9 of 90
building tree 10 of 90
building tree 11 of 90
building tree 12 of 90
building tree 13 of 90
building tree 14 of 90
building tree 15 of 90
building tree 16 of 90
building tree 17 of 90
building tree 18 of 90
building tree 19 of 90
building tree 20 of 90
building tree 21 of 90
building tree 22 of 90
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   2 out of   2 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   3 out of   3 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   4 out of   4 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   5 out of   5 | elapsed:    0.0s remaining:    0.0s
```

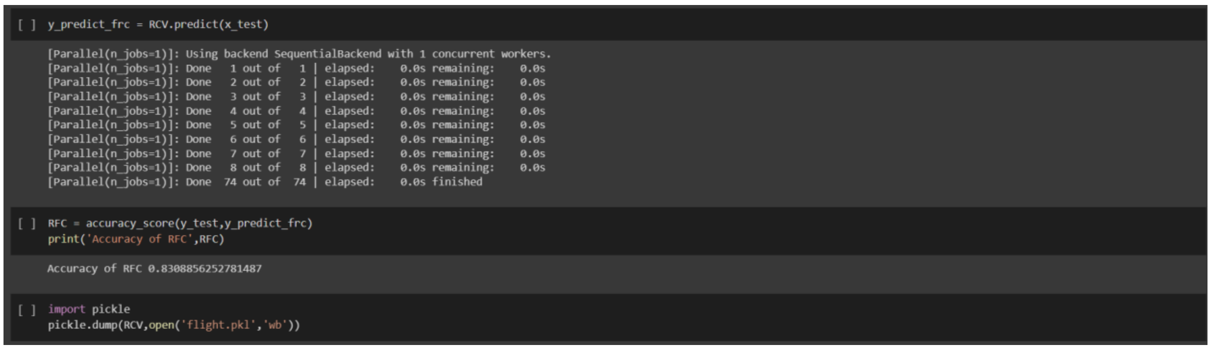


```
[ ] bt_params = RCV.best_params_
    bt_score = RCV.best_score_

[ ] bt_params

    {'verbose': 2,
     'n_estimators': 55,
     'max_features': 'sqrt',
     'max_depth': 10,
     'criterion': 'gini'}

bt_score

0.8338160936056148

[ ] model = RandomForestClassifier(verbose= 10, n_estimators= 120, max_features= 'log2',max_depth= 10,criterion='entropy')
    RCV.fit(x_train,y_train)

    building tree 1 of 74
    building tree 2 of 74
    building tree 3 of 74
    building tree 4 of 74
    building tree 5 of 74
    /usr/local/lib/python3.9/dist-packages/sklearn/model_selection/_validation.py:686: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change th
      estimator.fit(X_train, y_train, **fit_params)
    [Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
    [Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
    [Parallel(n_jobs=1)]: Done   2 out of   2 | elapsed:    0.0s remaining:    0.0s
```

Model Deployment

Save the best model

Saving the best model after comparing its performance using different evaluation metrics means selecting the model with the highest performance and saving its weights and configuration. This can be useful in avoiding the need to retrain the model every time it is needed and also to be able to use it in the future.



```
[ ] y_predict_frc = RCV.predict(x_test)

    [Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
    [Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
    [Parallel(n_jobs=1)]: Done   2 out of   2 | elapsed:    0.0s remaining:    0.0s
    [Parallel(n_jobs=1)]: Done   3 out of   3 | elapsed:    0.0s remaining:    0.0s
    [Parallel(n_jobs=1)]: Done   4 out of   4 | elapsed:    0.0s remaining:    0.0s
    [Parallel(n_jobs=1)]: Done   5 out of   5 | elapsed:    0.0s remaining:    0.0s
    [Parallel(n_jobs=1)]: Done   6 out of   6 | elapsed:    0.0s remaining:    0.0s
    [Parallel(n_jobs=1)]: Done   7 out of   7 | elapsed:    0.0s remaining:    0.0s
    [Parallel(n_jobs=1)]: Done   8 out of   8 | elapsed:    0.0s remaining:    0.0s
    [Parallel(n_jobs=1)]: Done  74 out of  74 | elapsed:    0.0s finished

[ ] RFC = accuracy_score(y_test,y_predict_frc)
    print('Accuracy of RFC',RFC)

    Accuracy of RFC 0.8308856252781487

[ ] import pickle
    pickle.dump(RCV,open('flight.pkl','wb'))
```

Integrate with Framework

Building a web application that is integrated to the model we built. A UI is provided for the uses where he has to enter the values for predictions. The enter values are given to the saved model and prediction is showcased on the UI.

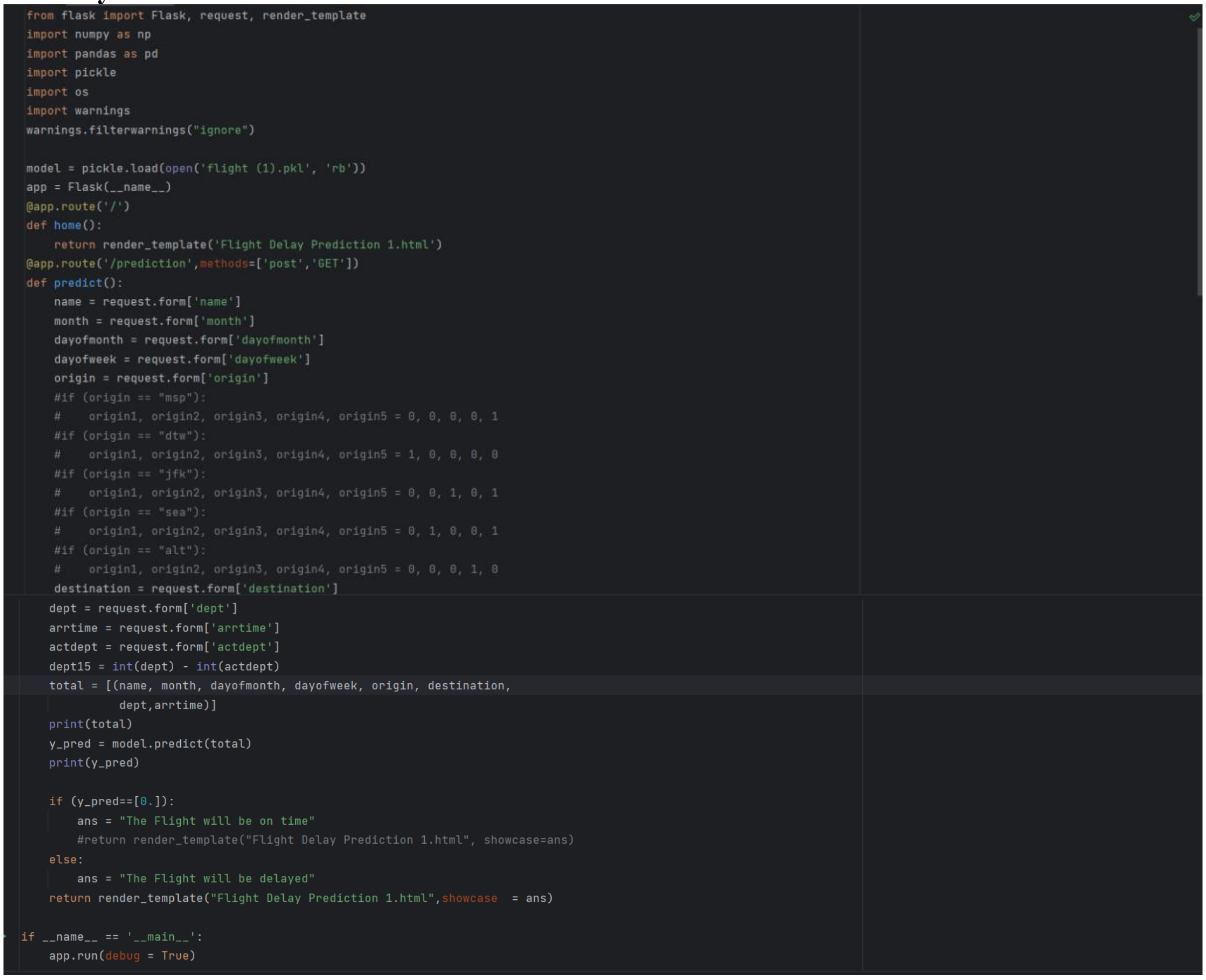
Building up HTML pages

For this project we create one HTML files namely

● Flight Delay Prediction 1.html

and save them in the templates folder.

Build Python code



```python
from flask import Flask, request, render_template
import numpy as np
import pandas as pd
import pickle
import os
import warnings
warnings.filterwarnings("ignore")

model = pickle.load(open('flight (1).pkl', 'rb'))
app = Flask(__name__)
@app.route('/')
def home():
    return render_template('Flight Delay Prediction 1.html')
@app.route('/prediction',methods=['post','GET'])
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, 1
    #if (origin == "sea"):
    #    origin1, origin2, origin3, origin4, origin5 = 0, 1, 0, 0, 1
    #if (origin == "alt"):
    #    origin1, origin2, origin3, origin4, origin5 = 0, 0, 0, 1, 0
    destination = request.form['destination']
    dept = request.form['dept']
    arrtime = request.form['arrtime']
    actdept = request.form['actdept']
    dept15 = int(dept) - int(actdept)
    total = [(name, month, dayofmonth, dayofweek, origin, destination,
              dept,arrtime)]
    print(total)
    y_pred = model.predict(total)
    print(y_pred)

    if (y_pred==[0.]):
        ans = "The Flight will be on time"
        #return render_template("Flight Delay Prediction 1.html", showcase=ans)
    else:
        ans = "The Flight will be delayed"
    return render_template("Flight Delay Prediction 1.html",showcase  = ans)

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