PROJECT DOCUMENT

Date	19 November 2022
Team ID	PNT2022TMID46686
Project Name	Developing a Flight Delay Prediction Using Machine Learning

1. INTRODUCTION

1.1 Project Overview

Over the last twenty years, air travel has been increasingly preferred among travelers, 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.
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.

1.2 Purpose

The main objective of the model is to predict flight delays accurately in order to optimize flight operations and minimize delays.
Therefore, predicting flight delays can improve airline operations and passenger satisfaction, which will result in a positive impact on the economy. In this study, the main goal is to compare the performance of machine learning classification algorithms when predicting flight delays.

2. LITERATURE SURVEY

2.1 Existing problem

1.Study of flight Departure Delay and Casual Factors Using SpatialAnalysis

Assuming delay as a spatially dependent variable, finds delay distribution pattern to predict delay.

Advantages: Considers spatial factors, people, day types and time ranges of a day to contribute to the prediction.
Disadvantages: Some of the attributes considered cannot be obtained on large scale in real time.

2. Flight delay forecasting and analysis of direct and indirect factors

Network with attention mechanism to remember spatial dependencies.
Advantages: Direct and Indirect causing factors are weighted differently.

Disadvantages: Air interaction of flights not taken into account.

3. Flight delay prediction based on aviation big data and machinelearning

Comparision of LSTM and Random forest; Uses ADS-B_data for improvedaccuracy.

Advantages: Use of ADS-B can be seen promising. Showed that LSTM suffersfrom overfitting on test set.

Disadvantages: Deployment of ADS-B is hectic. More data handing takesplace.

4.Prediction of weather-induced airline delays based on machine learning algorithms

Experimented predicting delay using supervised machine learning algorithms. Uses SMOTE for weaker class sampling.

Advantages: Found weather causes to be amounting to a significant percentof delay.

Disadvantages: Flight donot taken spatial dependencies into account. Amount of delay could have been found.

5.Delay prediction from spatial and temporal perspective

ST-Random Forest for flight delay prediction using spatial features of aviation network and temporal correlation of weather condition and airportcrowdedness on flight delays.

Advantages: A real-time, highly accurate prediction system that guaranties the influence of the air traffic network in the prediction.

Disadvantages: Overfitting might occur due to LSTM.

6.Airline Flight Delay Prediction Using Machine Learning Models

Comparison among 7 classification machine learning algorithms.

Advantages: Among the considered alongs using 4 performance indicators decision tree was found to be the best in predicting flight delays.

Disadvantages: The data imbalance issue even through handled through weighted evaluation methods does had a significant effect on performance on the algorithm.

7.Predicting flight delay based on multiple linear regression

A multiple linear regression algorithm to predict delay.

Advantages: Both airline and weather features are taken into consideration. The methodology used in this gives better results compared to Naïve-Bayesand C4.5 approach.

Disadvantages: Predicts only the flights which are delayed above 30 minutes.

8.Flight Delay Prediction System

Supervised Machine Learning algorithm using Naïve Bayes.

Advantages: Considers independence among the predictors making the system scalable. Good for real time prediction.

Disadvantages: Does not take into account the impact of unprecedented reasons such as major calamities in flight delays.

9.A Deep Learning approach to flight delay prediction

A deep RNN and LSTM approach to prediction; uses limited data attributes

Advantages: Predicting two sections namely day prediction and flight prediction seems more reasonable and can give more insights for the airport managers to make necessary arrangements.

Disadvantages: Air traffic/flight interaction doesn't play great roles. Biased towards weather attributes.

2.2 References

1.Shaowu Cheng, Yaping Zhang, Siqi IIao, Ruiwei Liu, Xiao Luo, Qian Luo, "Study of Flight Departure Delay and Casual Factors Using Spatial Analysis", Journal of Advanced Transportation, vol.2019, Article ID 3525912, 11 pages, 2019.https://doi.org/10.1155/2019/3525

2.Wang, F., Bi, J., Xie, D., Zhao, X., Flight delay forecasting and analysis of direct and indirect factors. IET Intell. Transp.. Syst. 16,890-907(2022).https://doi.org/10.1049/itr2.12183

3.Gui , G., Liu, F., Sun, J., Zhou, X., Flight delay prediction based on aviation big data and machine learning. *IEEE Transactions on Vehicular Technology, 69(1), 140-150.*

4.Choi, S., Kim, Y. J., Briccno, S., & Mavris, D.(2016, September). Prediction of weather-induced airline delays based on machine learning algorithms. *In 2016 IEEE/AIAA 35th Digital Avionics System Conference (DASC) (pp. 1-6). IEEE.*

5.Li, Q., & Jing, R. (2022). *Flight Delay Prediction from Spatial and Temporal Perspective. Expert Systems with Applications, 117662.*

6.Tang, Y. (2021, October). Airline Flight Delay Prediction Using Machine Learning Models. *In 2021 5th International Conference on E- Business and Internet (pp. 151-154).*

7.Ding, Y. (2017, August). Predicting flight delay based on multiple linear regression. *In IOP Conference Series: Earth and Environmental Science (Vol. 81, No. 1, p. 012198).* IOP Publishing.

8.Borse, Y., Jain, D., Sharma, S., Vora, V.,& Zaveri, A. (2020). Flight Delay Prediction System. *Int. J. Eng. Res. Techno, 9(3), 88-92.*

9.Kim, Y . J., Choi, S., Briceno, S., & Mavris, D. (2016, September). A deep learning approach to flight delay prediction. *In 2016 IEEE/AIAA 35th Digital Avionics System Conference (DASC) (pp. 1-6). IEEE.*

2.3 Problem Statement Definition:

Customer Problem Statement :

Delay in flight makes passengers concerned and this matter causes extra expenses for the agency and the airport itself. It can affect the trade, because goods' transport is highly dependant on customer trust, which can increase or decrease the ticket sales.

Example:

Problem Statement (PS)	Iam (Customer)	I'm trying to	But	Because	Which makes me feel
PS-1	a business professional	go abroad fora business meeting	if the flights would be delayed	due to the weather condition	stress
PS-2	a tourist	go my native	if the flights would be cancelled	some technica lissues	worried

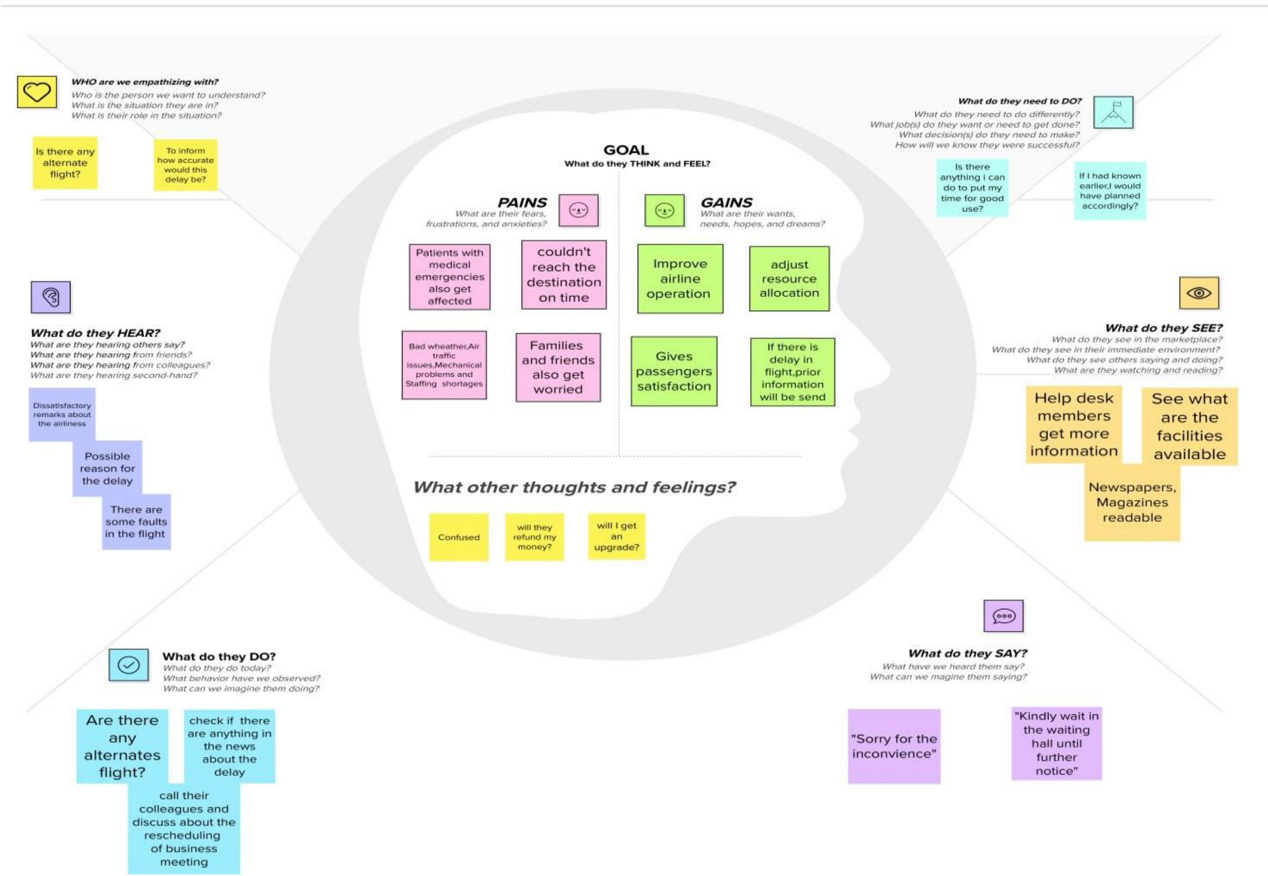
3. IDEATION & PROPOSED SOLUTION:

3.1 Empathy Map Canvase:

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. Using a machine learning model, we can predict flight arrival delays.

Example:

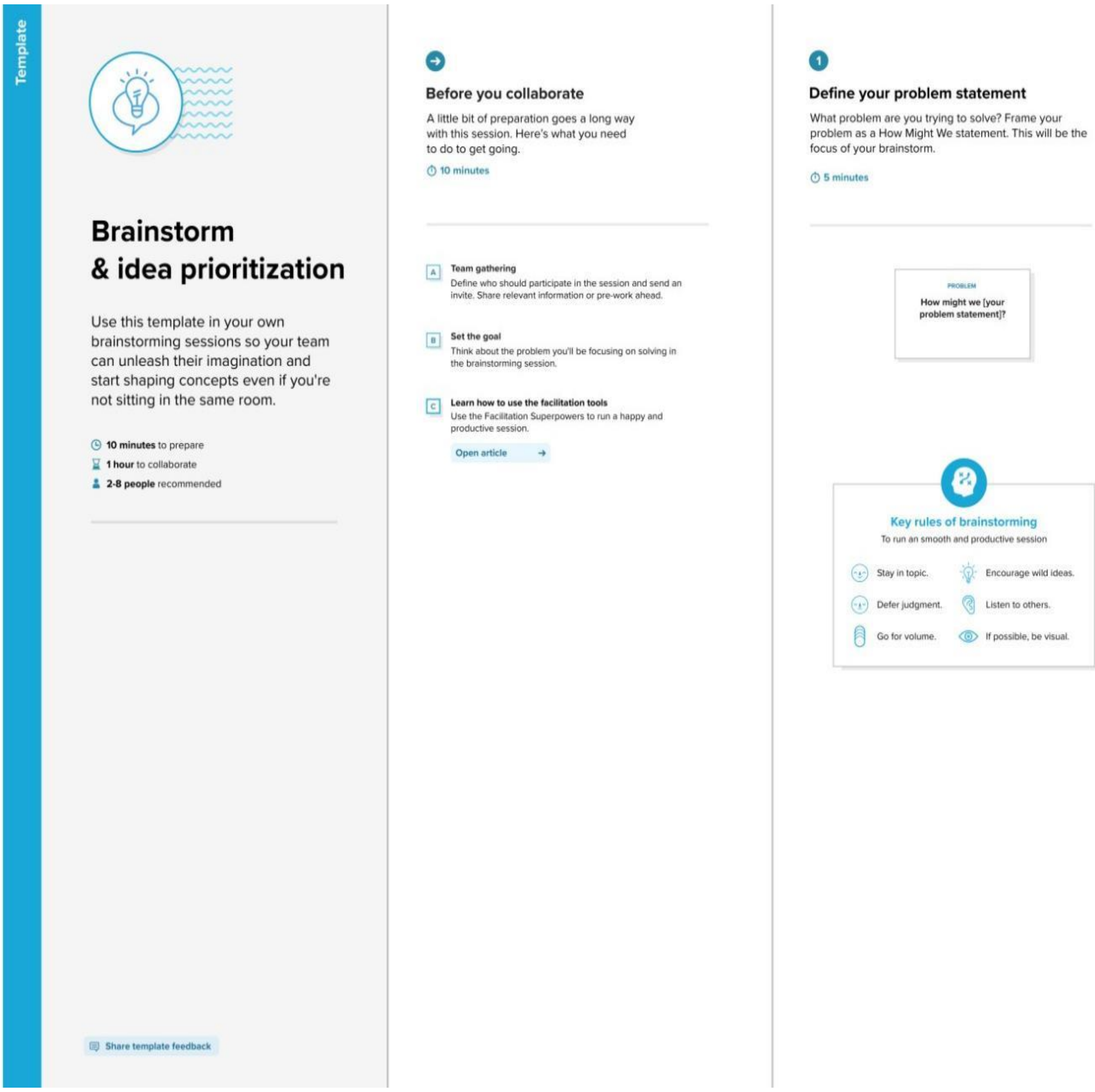
Empathy Map



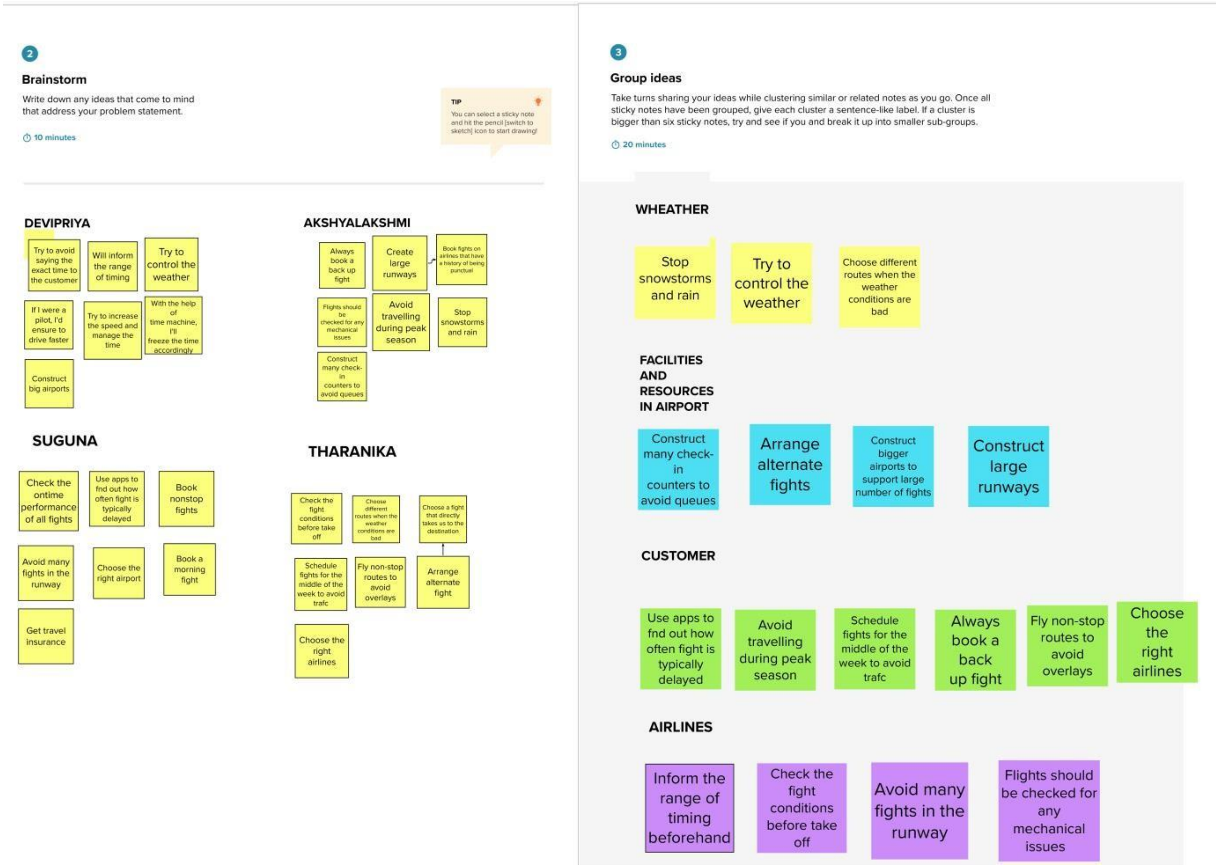
3.2 Brainstorm & Idea Prioritization :

The main objective of the model is to predict flight delays accurately inorder to optimize flight operations, to save passengers and airlines fromall the hardships caused due to flight delays or in worst case cancellations.

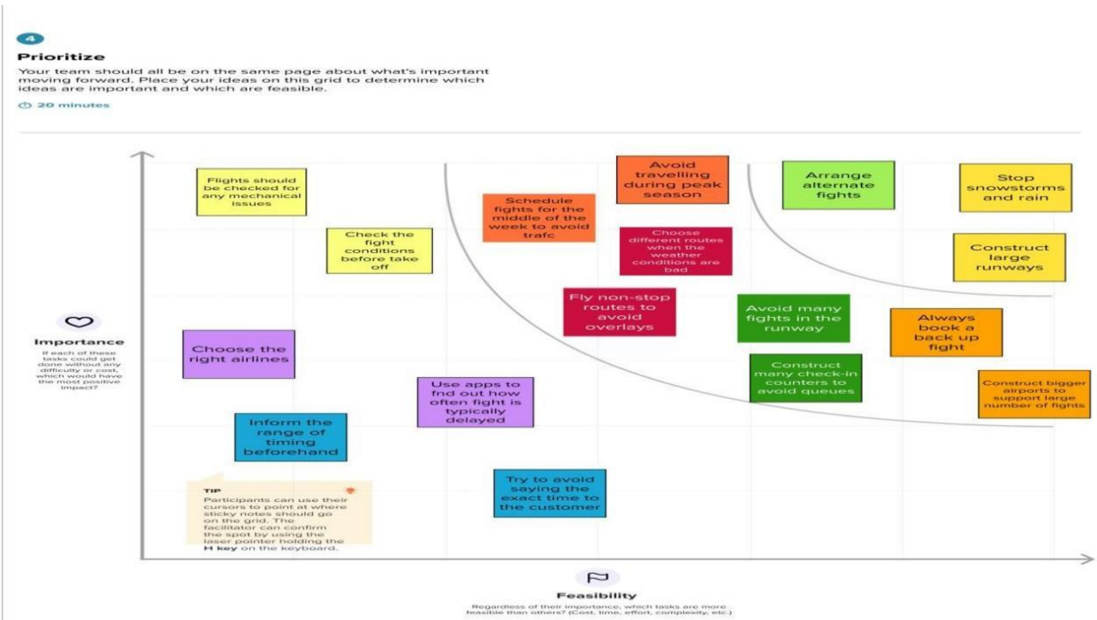
Step-1: Team Gathering, Collaboration and Select theProblem Statement



Step-2: Brainstorm, Idea Listing and Grouping



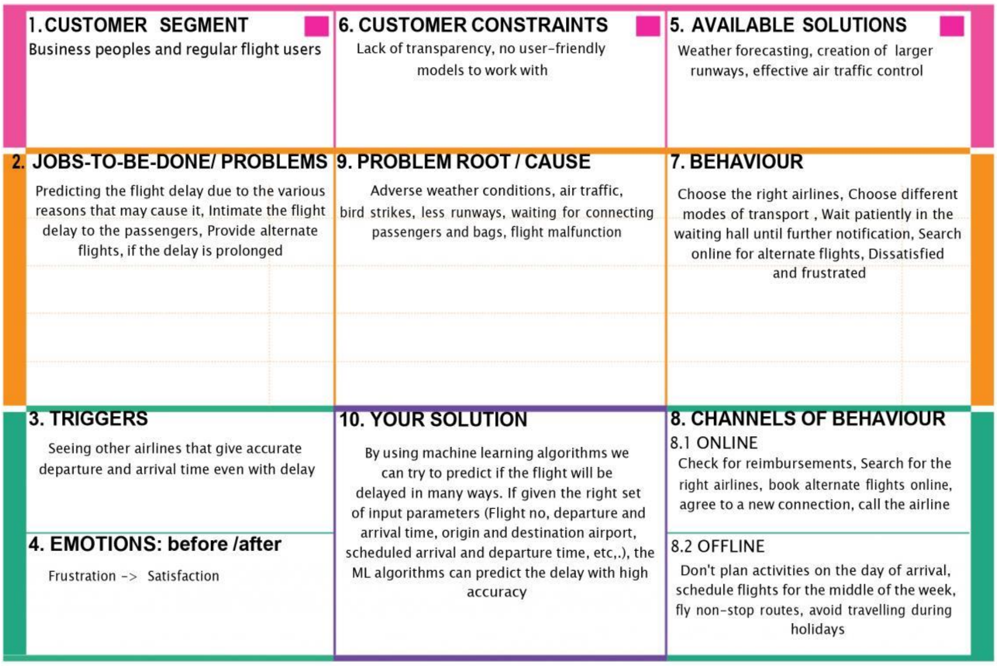
Step-3: Idea Prioritization



3.3 Proposed Solution

S.No.	Parameter	Description
1.	Problem Statement (Problemto be solved)	The main objective of the model is to predict flight delays accurately in order to optimize flight operations, to save passengers and airlines from all the hardships caused due to flight delays or in worst case cancellations.
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. 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.
3.	Novelty / Uniqueness	We compare decision tree classifier with logistic regression and a simple neural network for various figures of merit.
4.	Social Impact / Customer Satisfaction	Time management will be the social impact such that when informed earlier the passengers can planaccordingly.
5.	Business Model (Revenue Model)	• Low-cost airline business model. • B2C business Model.
6.	Scalability of the Solution	The delayed time of any type of flight can be known with maximumaccuracy

3.4 Problem Solution fit



4.REQUIREMENT ANALYSIS

4.1 Functional requirement

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	• Registration through Form • Registration through Gmail • Registration through LinkedIN
FR-2	User Confirmation	• Confirmation via Email • Confirmation via OTP
FR-3	login	• The system must allow users to log into their account by entering their email and password.
FR-4	Forgot password	• The system must allow users to reset their password by clicking on "I forgot my password" and receiving a link totheir verified email address.

FR-5	Submit	• The system must display the details users to submit all the asked information

4.2 Non-Functional requirements

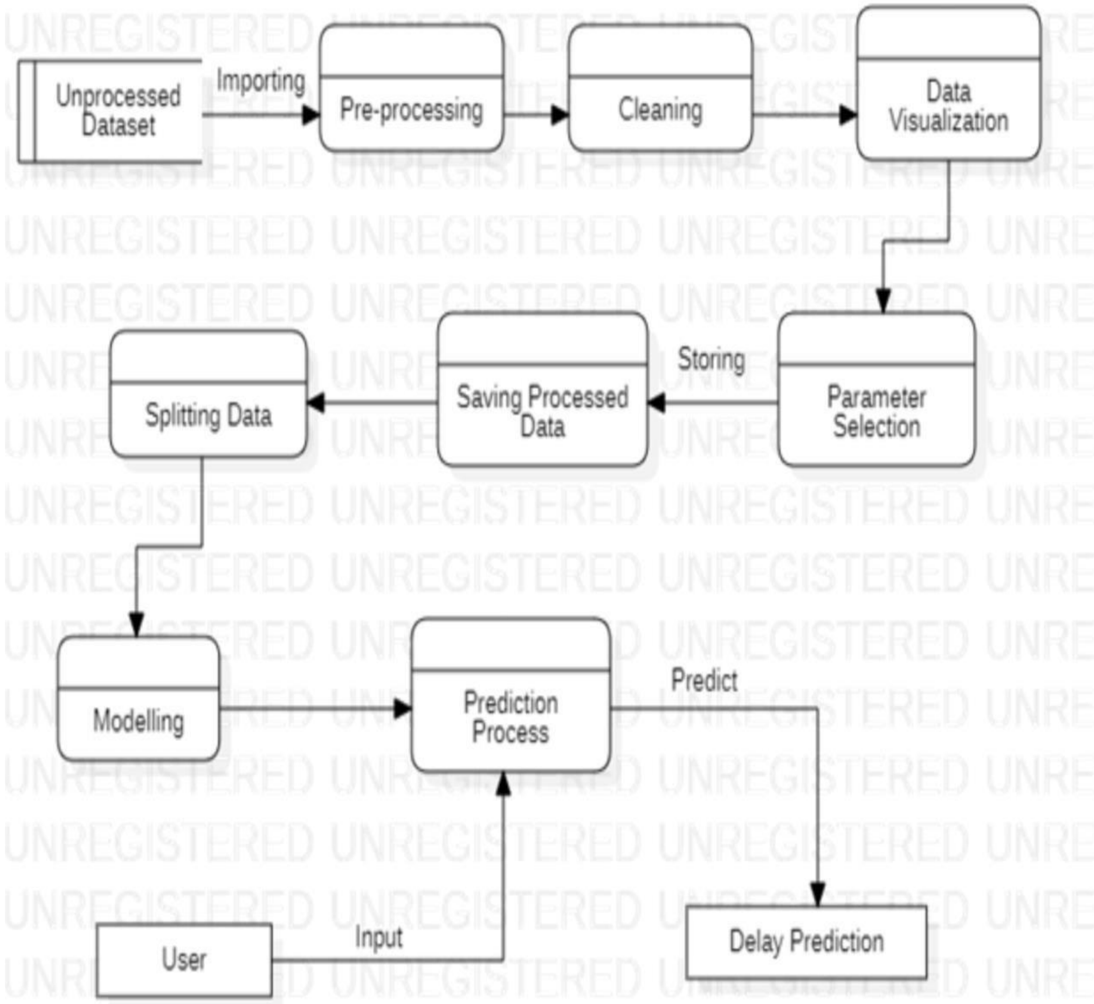
NFR No.	Non-Functional Requirement	Description
NFR-1	Usability	• Indicates how effectively and easyuser can learn and use a system.
NFR-2	Security	• Security is a non-functional requirement assuring all data inside the system or its part will be protected against malware attacks orunauthorized access. • Only direct managers can see personnel records of staff. • Customers can see their order history only during business hours.
NFR-3	Reliability	• Reliability specifies how likely the system or its element would run without a failure for a given period oftime under predefined conditions.
NFR-4	Performance	• Any interaction between the user and the system should not exceed 2seconds. • The system should receive updated inventory information every 15 minutes.
NFR-5	Availability	• Availability describes how likely the system is accessible to a user at a given point in time. • While it can be expressed as an expected percentage of successful requests, you may also define it as a percentage of time the system is accessible for operation during some time period.

5. PROJECT DESIGN

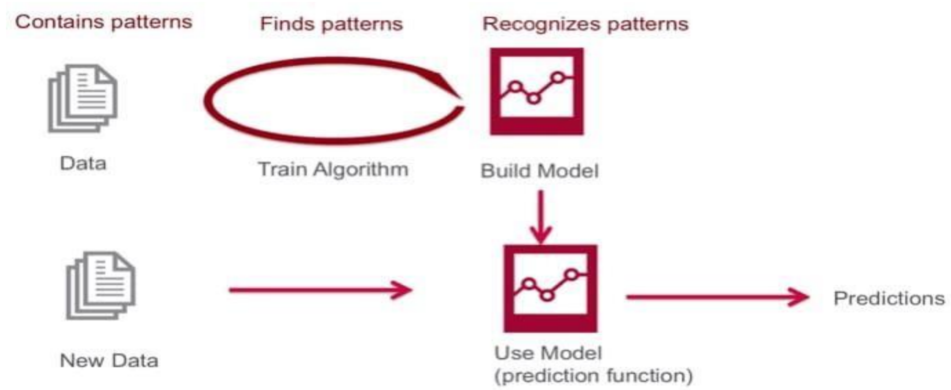
5.1 Data Flow Diagrams

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. We experimented with different combinations of classifier models and preprocessing procedures to resolve the problems we identified in previous work, such as label imbalance and encoding of categorical features. With resampling of the training data and use of a random forest classifier, we achieved higher recall, precision, and f2 scores, and thus a useful flight delay predictor.

Example: DFD Level 0 (Industry Standard)



Example: (Simplified)



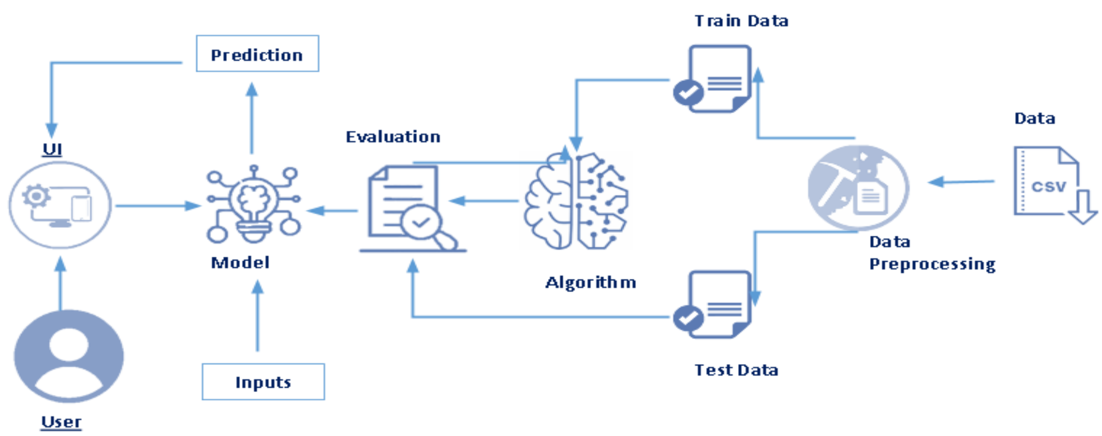
5.2 Solution & Technical Architecture

Solution Architecture

The main objective of the model is to predict flight delays accurately in order to optimize flight operations and minimize delays.

- These delays are responsible for large economic and environmental losses
- 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.
- A flight is considered to be delayed when difference between scheduled and actual arrival times.

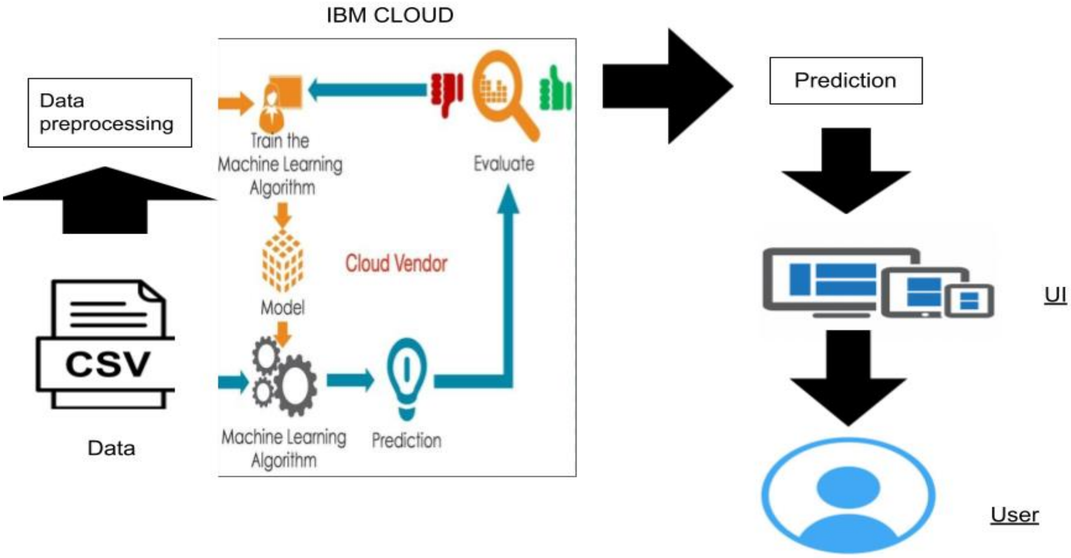
Example - Solution Architecture Diagram:



Technical Architecture:

This model predicts if there is any delay in flight time. If there is a delay in flight we predict by how much time it will get delayed using IBM cloud storage in Machine Learning.

Example:



5.3 User Stories

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user)	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dashboard	High	Sprint-1
		USN-2	As a user, I will receive confirmation email once I have registered for the application	I can receive confirmation email & click confirm	High	Sprint-1
		USN-3	As a user, I can register for the	I can register & access the	Low	Sprint-2

			application through Facebook	dashboard with Facebook Login		
		USN-4	As a user, I can register for the application through Gmail		Medium	Sprint-1
	Login	USN-5	As a user, I can log into the application by entering email & password		High	Sprint-1
Customer (Web user)	Login	USN-6	As a web user, I can log into the application by entering my web mail id and password		Medium	Sprint-1
Customer Care Executive	Login	USN-7	As a care executive , I can log into the application by entering mail id to view the customer details		Medium	Sprint-1
Administrator	Login	USN-8	As a Administrator, I can log into the application by entering mail id to solve the customer queries		High	Sprint-1

6. PROJECT PLANNING & SCHEDULING

6.1 Sprint Planning & Estimation

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	story Points	Priority	Team Members
Sprint-1	Data Collection	USN-1	As a user, I can't interact anything. Waiting is user's task. User can listen the relationship exist between	2	high	Devipriya

			the various attributes of data bypresentation of developer			
Sprint-1	Data Pre-processing And Model Building	USN-2	As a user, I can predict flight delay by various developed ML models byconsole	1	high	Suguna Devipriya Akshayalakshmi Tharanika
Sprint-3	Application Building	USN-3	As a user, I can register for the application by entering my username, password, and confirming my password.	2	high	Devipriya Tharanika
Sprint-2	Train the Model on IBM	USN-4	As a User, I can the model by requesting the deployed model on cloud.	1	Medium	Suguna Devipriya
Sprint-3	Ideation Phase	USN-5	As a user ,I can gather the relevent information on project use case and capture the project gains and pains and analyse three ideas on the feasiblity and importance.	2	high	Suguna Devipriya Akshayalakshmi Tharanika
Sprint-3	Project Design Phase-I	USN-6	As a user, I can analyse and preparethe solution document	1	Medium	Suguna Devipriya Akshayalakshmi Tharanika
Sprint-3	Project Design Phase-II	USN-7	As a user, I can prepare the user intraction and experience of the applications	2	High	Suguna Devipriya Akshayalakshmi Tharanika
Sprint-3	Project Planning Phase	USN-7	As a user, we can prepare activity listof project	2	High	Suguna Devipriya Akshayalakshmi Tharanika
Sprint-4	Project Development Phase	USN-8	As a user, we can prepare developedcoding and testing it	2	Medium	Suguna Devipriya Akshayalakshmi Tharanika

6.2 Sprint Delivery Schedule

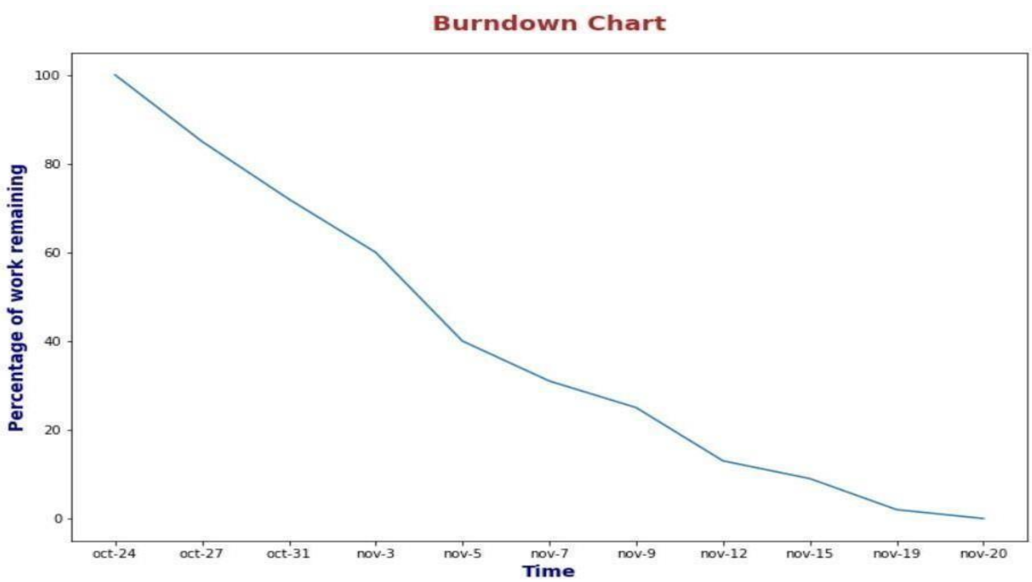
Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date(Actual)
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022	20	31 Oct 2022
Sprint-2	20	6 Days	24 Oct 2022	05 Nov 2022	20	31 Oct 2022
Sprint-3	20	6 Days	24 Oct 2022	12 Nov 2022	20	31 Oct 2022
Sprint-4	20	6 Days	24 Oct 2022	19 Nov 2022	20	31 Oct 2022

Velocity:

Imagine we have a 10-day sprint duration, and the velocity of the team is 20 (points per sprint). Let's calculate the team's average velocity (AV) per iteration unit (story points per day)

$$AV = \frac{sprint\ duration}{velocity} = \frac{20}{10} = 2$$

6.3 Reports from JIRA



7. CODING & SOLUTIONING

7.1 Feature 1

- ✓ Anoconda Prompt-Jupyter Notebook
- ✓ IBM Watson Studio
- ✓ Visual Studio-Design Page
- ✓ Spyder/Python IDE-Flask

7.2 Feature 2

- ✓ Registration Page
- ✓ Login Page
- ✓ Prediction Page

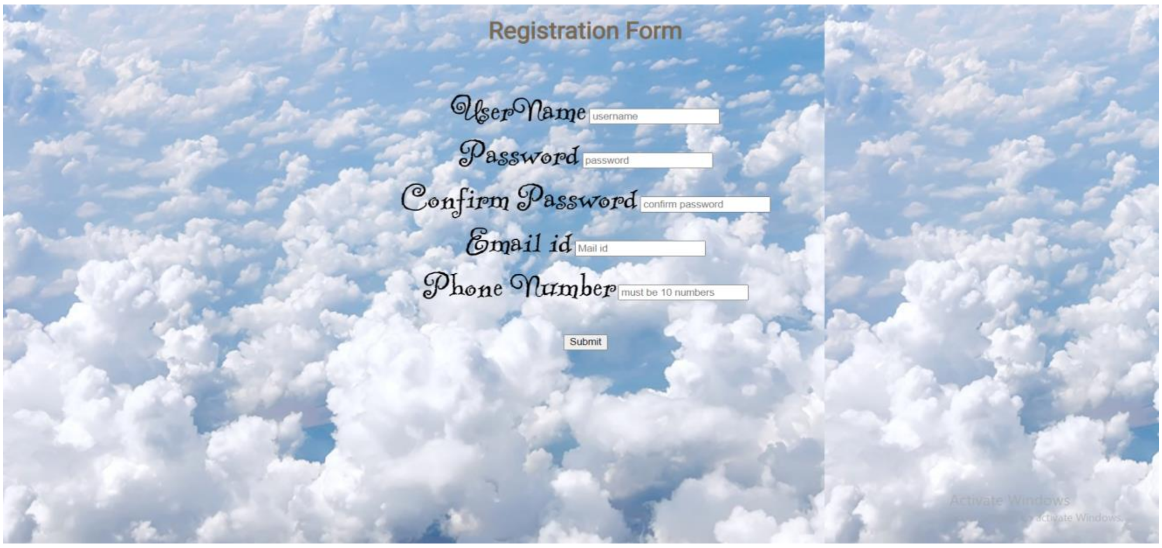
7.3 Database Schema

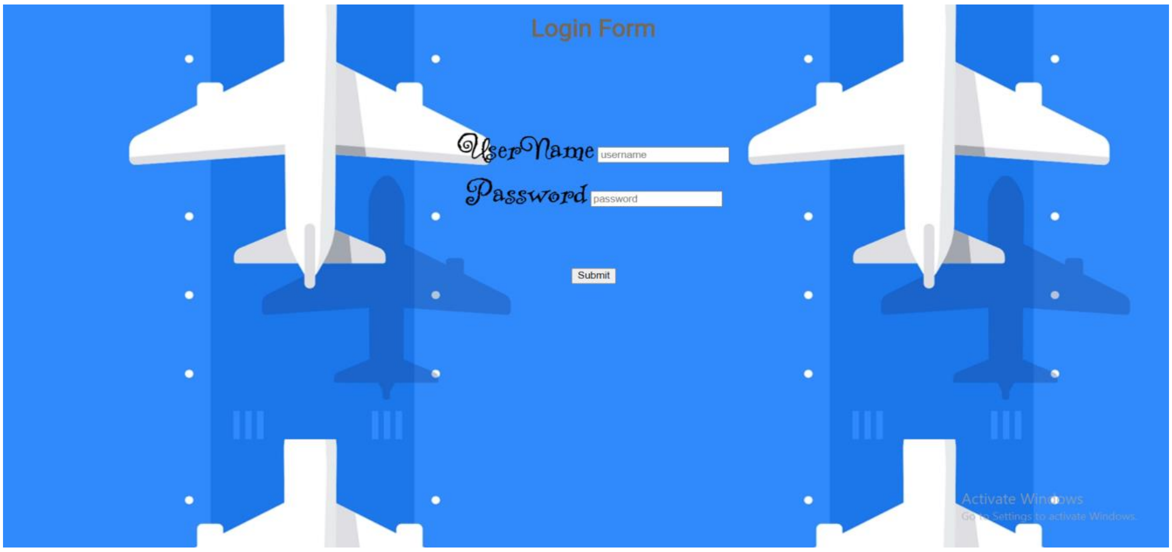
- ✓ Flask
- ✓ IBM Watson Cloud

8. TESTING

8.1 Test Cases

8.2 User Acceptance Testing





9. RESULTS

9.1 Performance Metrics

You'r Flight is delayed

For the bad weather condition

10. ADVANTAGES & DISADVANTAGES

ADVANTAGES

- Predicting flight delays can improve airline operations and passenger satisfaction, which will result in a positive impact on the economy.

- The main goal is to compare the performance of machine learning classification algorithms when predicting flight delays.
- The results can be applied to increase customer satisfaction and incomes of airline agencies.

DISADVANTAGES

- Delay in flight is inevitable , which has too much negative economic effects on passengers.
- Increase in capital costs, reallocation of flight crews and aircraft, and additional crew expenses.

11. CONCLUSION

Predicting flight delays is on interesting research topic and required many attentions these years. Majority of research have tried to develop and expand their models in order to increase the precision and accuracy of predicting flight delays. Since the issue of flights being on-time is very important, flight delay prediction models must have high precision and accuracy.

We can see that it is possible to predict flight delay patterns from just the volume of concurrently published tweets, and their sentiment and objectivity. This is not unreasonable; people tend to post about airport delays on Twitter; it stands to reason that these posts would become more frequent, and more profoundly emotional, as the delays get worse.

Without more data, we cannot make a robust model and find out the role of related factors and chance on these results.

However, as a proof of concept, there is potential for these results. It may be possible to routinely use tweets to ascertain an understanding of concurrent airline delays and traffic patterns, which could be useful in a variety of circumstances

12. FUTURE SCOPE

The future work of this project includes incorporating a larger dataset. There are many different ways to pre-process a larger dataset like running a Spark cluster over a server or using a cloud-based services like AWS and Azure to process the data. With the new advancement in the field of Machine learning, we can use Neural Networks algorithm on the flight and weather data. Neural Network works on the pattern matching methodology.
It is divided into three basic parts for data modelling that includes feed forward networks, feedback networks, and self organization network. Feed-forward and feedback networks are generally used in the areas of prediction, pattern recognition, associative memory, and optimization calculation, whereas self-organization networks are generally used in cluster analysis. Neural Network

offers distributed computer architecture with important learning abilities to represent nonlinear relationships.

13.APPENDIX

Source code

Notebook code:

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [3]:
```
flights=pd.read_csv("C:\\Users\\Devi\\Downloads\\flights.csv\\flights.csv")
flights=flights.sample(n=100000)
flights.head()
```

```
C:\Users\Devi\AppData\Local\Temp\ipykernel_2956\1966170758.py:1: DtypeWarning: Columns (7,8) have mixed types. Specify dtype op
tion on import or set low_memory=False.
  flights=pd.read_csv("C:\\Users\\Devi\\Downloads\\flights.csv\\flights.csv")
```

Out[3]:

	YEAR	MONTH	DAY	DAY_OF_WEEK	AIRLINE	FLIGHT_NUMBER	TAIL_NUMBER	ORIGIN_AIRPORT	DESTINATION_AIRPORT	SCHEDULED_DEPARTUR
5126156	2015	11	17	2	MQ	3125	N646MQ	BMI	ORD	6:
301985	2015	1	20	2	AA	2482	N505AA	DFW	AUS	18
4886973	2015	11	1	7	EV	2828	N629AE	DFW	LAW	21!
1589374	2015	4	12	7	WN	1371	N368SW	BNA	CLE	14!
4545902	2015	10	11	7	DL	1370	N917DN	10397	14576	9

5 rows × 31 columns

In [7]:
```
flights.shape
```

Out[7]: (100000, 31)

In [8]:
```
flights.isnull().values.any()
```

Out[8]: True

In [9]:
```
#Checking how many null values are there in each columns
flights.isnull().sum()
```

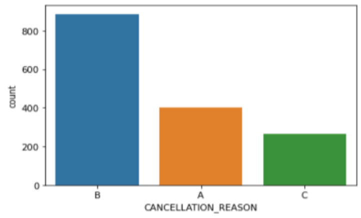
Out[8]: True

In [9]: #Checking how many null values are there in each columns
flights.isnull().sum()

Out[9]:
```
YEAR                       0
MONTH                      0
DAY                        0
DAY_OF_WEEK                0
AIRLINE                    0
FLIGHT_NUMBER              0
TAIL_NUMBER              277
ORIGIN_AIRPORT             0
DESTINATION_AIRPORT        0
SCHEDULED_DEPARTURE        0
DEPARTURE_TIME          1477
DEPARTURE_DELAY         1477
TAXI_OUT                1538
WHEELS_OFF              1538
SCHEDULED_TIME             1
ELAPSED_TIME            1803
AIR_TIME                1803
DISTANCE                   0
WHEELS_ON               1596
TAXI_IN                 1596
SCHEDULED_ARRIVAL          0
ARRIVAL_TIME            1596
ARRIVAL_DELAY           1803
DIVERTED                   0
CANCELLED                  0
CANCELLATION_REASON    98449
AIR_SYSTEM_DELAY       81864
SECURITY_DELAY         81864
AIRLINE_DELAY          81864
LATE_AIRCRAFT_DELAY    81864
WEATHER_DELAY          81864
dtype: int64
```


In [10]:
```
sns.countplot(x='CANCELLATION_REASON',data=flights)
```

```
In [10]: sns.countplot(x='CANCELLATION_REASON',data=flights)
Out[10]: <AxesSubplot:xlabel='CANCELLATION_REASON', ylabel='count'>
```

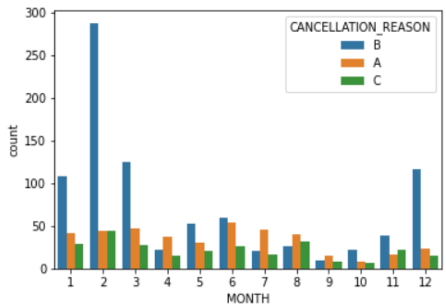


Reason for Cancellation of flight: A - Airline/Carrier; B - Weather; C - National Air System; D - Security

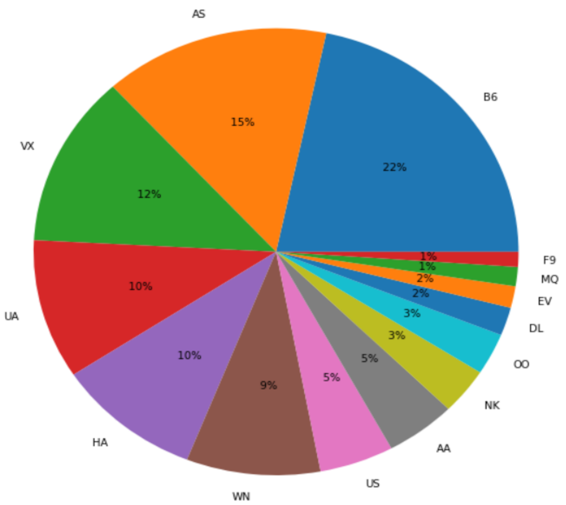
We can observe from graph easily that mostly weather is responsible for delays of flight.

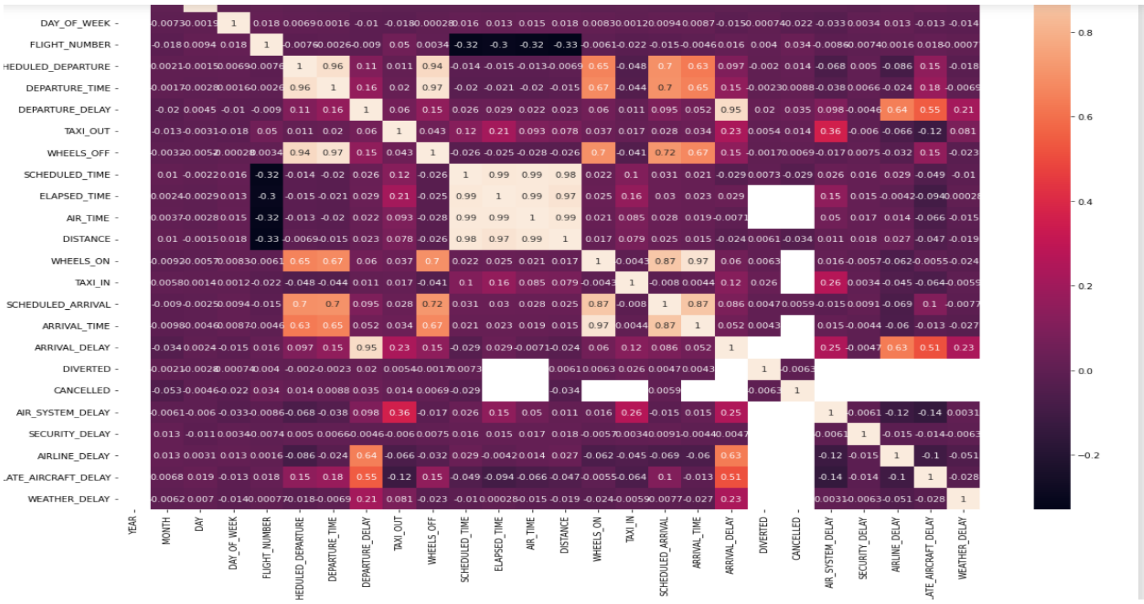
```
In [11]: sns.countplot(x="MONTH",hue="CANCELLATION_REASON",data=flights)
Out[11]: <AxesSubplot:xlabel='MONTH', ylabel='count'>
```


```
In [11]: sns.countplot(x="MONTH",hue="CANCELLATION_REASON",data=flights)
Out[11]: <AxesSubplot:xlabel='MONTH', ylabel='count'>
```


```
In [13]: axis = plt.subplots(figsize=(10,14))
         Name = flights["AIRLINE"].unique()
         size = flights["AIRLINE"].value_counts()
         plt.pie(size,labels=Name,autopct='%5.0f%%')
         plt.show()
```






```
In [15]: corr=flights.corr()
         corr
```

Out[15]:

	YEAR	MONTH	DAY	DAY_OF_WEEK	FLIGHT_NUMBER	SCHEDULED_DEPARTURE	DEPARTURE_TIME	DEPARTURE_DELAY
YEAR	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
MONTH	NaN	1.000000	0.007081	-0.007332	-0.017579	0.002145	-0.001736	-0.020367
DAY	NaN	0.007081	1.000000	-0.001879	0.009444	-0.001522	-0.002795	0.004475
DAY_OF_WEEK	NaN	-0.007332	-0.001879	1.000000	0.017603	0.006920	0.001583	-0.010172
FLIGHT_NUMBER	NaN	-0.017579	0.009444	0.017603	1.000000	-0.007647	-0.002607	-0.009030
SCHEDULED_DEPARTURE	NaN	0.002145	-0.001522	0.006920	-0.007647	1.000000	0.962323	0.106157
DEPARTURE_TIME	NaN	-0.001736	-0.002795	0.001583	-0.002607	0.962323	1.000000	0.162673
DEPARTURE_DELAY	NaN	-0.020367	0.004475	-0.010172	-0.009030	0.106157	0.162673	1.000000
TAXI_OUT	NaN	-0.013407	-0.003136	-0.017501	0.050336	0.011400	0.019587	0.060080
WHEELS_OFF	NaN	-0.003186	-0.005191	-0.000279	0.003353	0.935585	0.970613	0.152682
SCHEDULED_TIME	NaN	0.010021	-0.002182	0.016069	-0.315325	-0.014080	-0.019955	0.025738
ELAPSED_TIME	NaN	0.002400	-0.002916	0.012793	-0.304805	-0.014835	-0.020563	0.029420
AIR_TIME	NaN	0.003713	-0.002754	0.015260	-0.317845	-0.012937	-0.020114	0.022075
DISTANCE	NaN	0.010276	-0.001468	0.017848	-0.329578	-0.006852	-0.015497	0.022768
WHEELS_ON	NaN	-0.009227	-0.005665	0.008346	-0.006089	0.653357	0.674076	0.059729
TAXI_IN	NaN	0.005813	0.001404	0.001187	-0.022263	-0.047533	-0.043566	0.011152
SCHEDULED_ARRIVAL	NaN	-0.009036	-0.002505	0.009368	-0.015010	0.697462	0.703035	0.095033
ARRIVAL_TIME	NaN	-0.009809	-0.004639	0.008686	-0.004624	0.628821	0.648776	0.051903
ARRIVAL_DELAY	NaN	-0.034249	0.002439	-0.015251	0.016388	0.096605	0.152964	0.945491
DIVERTED	NaN	-0.002083	-0.002818	-0.000745	0.004030	-0.002010	-0.002266	0.020249
CANCELLED	NaN	-0.053077	-0.004623	-0.022414	0.033925	0.014256	0.008834	0.035417
AIR_SYSTEM_DELAY	NaN	-0.006099	-0.005956	-0.032959	-0.008633	-0.067550	-0.038027	0.097931
SECURITY_DELAY	NaN	0.013406	-0.011116	0.003352	-0.007379	0.005027	0.006567	-0.004632

```
In [16]: variables_to_remove=["YEAR","FLIGHT_NUMBER","TAIL_NUMBER","DEPARTURE_TIME","TAXI_OUT","WHEELS_OFF","ELAPSED_TIME","AIR_TIME","WHE
         flights.drop(variables_to_remove,axis=1,inplace= True)
         flights.columns
```

```
Out[16]: Index(['MONTH', 'DAY', 'DAY_OF_WEEK', 'AIRLINE', 'ORIGIN_AIRPORT',
                'DESTINATION_AIRPORT', 'SCHEDULED_DEPARTURE', 'DEPARTURE_DELAY',
                'DISTANCE', 'ARRIVAL_DELAY'],
               dtype='object')
```

```
In [17]: airport = pd.read_csv("C:\\Users\\Devi\\Downloads\\airports.csv")
         airport
```

Out[17]:

	IATA_CODE	AIRPORT	CITY	STATE	COUNTRY	LATITUDE	LONGITUDE
0	ABE	Lehigh Valley International Airport	Allentown	PA	USA	40.65236	-75.44040
1	ABI	Abilene Regional Airport	Abilene	TX	USA	32.41132	-99.68190
2	ABQ	Albuquerque International Sunport	Albuquerque	NM	USA	35.04022	-106.60919
3	ABR	Aberdeen Regional Airport	Aberdeen	SD	USA	45.44906	-98.42183
4	ABY	Southwest Georgia Regional Airport	Albany	GA	USA	31.53552	-84.19447
...	...	...	...	...	...	...	...

```
flights.loc[~flights.ORIGIN_AIRPORT.isin(airport.IATA_CODE.values),'ORIGIN_AIRPORT']='OTHER'
flights.loc[~flights.DESTINATION_AIRPORT.isin(airport.IATA_CODE.values),'DESTINATION_AIRPORT']='OTHER'
flights
```

	MONTH	DAY	DAY_OF_WEEK	AIRLINE	ORIGIN_AIRPORT	DESTINATION_AIRPORT	SCHEDULED_DEPARTURE	DEPARTURE_DELAY	DISTANCE	ARRI
4330572	9	27	7	B6	BOS	PIT	1515	7.0	496	
2153991	5	17	7	B6	LAX	FLL	1430	4.0	2343	
2268611	5	24	7	AS	SEA	SNA	1655	-9.0	978	
5344954	12	1	2	VX	LAX	FLL	1025	53.0	2343	
1728777	4	21	2	UA	MCO	EWR	800	-8.0	937	
...	...	...	...	...	...	...	...	...	...	
3542391	8	8	6	AS	LAX	SEA	1955	82.0	954	
3777973	8	23	7	OO	SLC	BUR	838	-1.0	574	
4002231	9	6	7	WN	LAS	PIT	1010	1.0	1910	
1143520	3	16	1	DL	SFO	ATL	730	-2.0	2139	
5414693	12	5	6	AA	CLT	DFW	1855	-7.0	936	

100000 rows × 10 columns

```
print(flights.ORIGIN_AIRPORT.nunique())
print(flights.DESTINATION_AIRPORT.nunique())
print(flights.AIRLINE.nunique())
```



```
321
320
14
```

In [20]:
```
flights=flights.dropna()
flights
```

Out[20]:

	MONTH	DAY	DAY_OF_WEEK	AIRLINE	ORIGIN_AIRPORT	DESTINATION_AIRPORT	SCHEDULED_DEPARTURE	DEPARTURE_DELAY	DISTANCE	ARRI
4330572	9	27	7	B6	BOS	PIT	1515	7.0	496	
2153991	5	17	7	B6	LAX	FLL	1430	4.0	2343	
2268611	5	24	7	AS	SEA	SNA	1655	-9.0	978	
5344954	12	1	2	VX	LAX	FLL	1025	53.0	2343	
1728777	4	21	2	UA	MCO	EWR	800	-8.0	937	
...	...	...	...	...	...	...	...	...	...	
3542391	8	8	6	AS	LAX	SEA	1955	82.0	954	
3777973	8	23	7	OO	SLC	BUR	838	-1.0	574	
4002231	9	6	7	WN	LAS	PIT	1010	1.0	1910	
1143520	3	16	1	DL	SFO	ATL	730	-2.0	2139	
5414693	12	5	6	AA	CLT	DFW	1855	-7.0	936	

98197 rows × 10 columns

In [21]:
```
flights.shape
```

Out[21]: (98197, 10)

In [22]:
```
df=pd.DataFrame(flights)
df['DAY_OF_WEEK']= df['DAY_OF_WEEK'].apply(str)
df["DAY_OF_WEEK"].replace({"1":"SUNDAY", "2": "MONDAY", "3": "TUESDAY", "4":"WEDNESDAY", "5":"THURSDAY", "6":"FRIDAY", "7":"SATUR
flights
```

Out[22]:

	MONTH	DAY	DAY_OF_WEEK	AIRLINE	ORIGIN_AIRPORT	DESTINATION_AIRPORT	SCHEDULED_DEPARTURE	DEPARTURE_DELAY	DISTANCE	ARRI
4330572	9	27	SATURDAY	B6	BOS	PIT	1515	7.0	496	
2153991	5	17	SATURDAY	B6	LAX	FLL	1430	4.0	2343	

In [23]:
```
dums = ['AIRLINE','ORIGIN_AIRPORT','DESTINATION_AIRPORT','DAY_OF_WEEK']
df_cat=pd.get_dummies(df[dums],drop_first=True)
df_cat
```

Out[23]:

	AIRLINE_AS	AIRLINE_B6	AIRLINE_DL	AIRLINE_EV	AIRLINE_F9	AIRLINE_HA	AIRLINE_MQ	AIRLINE_NK	AIRLINE_OO	AIRLINE_UA	...	DESTINATIC
4330572	0	1	0	0	0	0	0	0	0	0	...	
2153991	0	1	0	0	0	0	0	0	0	0	...	
2268611	1	0	0	0	0	0	0	0	0	0	...	
5344954	0	0	0	0	0	0	0	0	0	0	...	
1728777	0	0	0	0	0	0	0	0	0	1	...	
...	...	...	...	...	...	...	...	...	...	...	...	...
3542391	1	0	0	0	0	0	0	0	0	0	...	
3777973	0	0	0	0	0	0	0	0	1	0	...	
4002231	0	0	0	0	0	0	0	0	0	0	...	
1143520	0	0	1	0	0	0	0	0	0	0	...	
5414693	0	0	0	0	0	0	0	0	0	0	...	

98197 rows × 658 columns

In [24]:
```
df_cat.columns
```

Out[24]:
```
Index(['AIRLINE_AS', 'AIRLINE_B6', 'AIRLINE_DL', 'AIRLINE_EV', 'AIRLINE_F9',
       'AIRLINE_HA', 'AIRLINE_MQ', 'AIRLINE_NK', 'AIRLINE_OO', 'AIRLINE_UA',
       ...
       'DESTINATION_AIRPORT_WYS', 'DESTINATION_AIRPORT_XNA',
       'DESTINATION_AIRPORT_YAK', 'DESTINATION_AIRPORT_YUM',
       'DAY_OF_WEEK_MONDAY', 'DAY_OF_WEEK_SATURDAY', 'DAY_OF_WEEK_SUNDAY',
       'DAY_OF_WEEK_THURSDAY', 'DAY_OF_WEEK_TUESDAY', 'DAY_OF_WEEK_WEDNESDAY'],
      dtype='object', length=658)
```


In [23]:
```
dums = ['AIRLINE','ORIGIN_AIRPORT','DESTINATION_AIRPORT','DAY_OF_WEEK']
df_cat=pd.get_dummies(df[dums],drop_first=True)
df_cat
```

Out[23]:

	AIRLINE_AS	AIRLINE_B6	AIRLINE_DL	AIRLINE_EV	AIRLINE_F9	AIRLINE_HA	AIRLINE_MQ	AIRLINE_NK	AIRLINE_OO	AIRLINE_UA	...	DESTINATIC
4330572	0	1	0	0	0	0	0	0	0	0	...	
2153991	0	1	0	0	0	0	0	0	0	0	...	
2268611	1	0	0	0	0	0	0	0	0	0	...	
5344954	0	0	0	0	0	0	0	0	0	0	...	
1728777	0	0	0	0	0	0	0	0	0	1	...	
...	...	...	...	...	...	...	...	...	...	...	...	...
3542391	1	0	0	0	0	0	0	0	0	0	...	
3777973	0	0	0	0	0	0	0	0	1	0	...	
4002231	0	0	0	0	0	0	0	0	0	0	...	
1143520	0	0	1	0	0	0	0	0	0	0	...	
5414693	0	0	0	0	0	0	0	0	0	0	...	

98197 rows × 658 columns

In [24]:
```
df_cat.columns
```

Out[24]:
```
Index(['AIRLINE_AS', 'AIRLINE_B6', 'AIRLINE_DL', 'AIRLINE_EV', 'AIRLINE_F9',
       'AIRLINE_HA', 'AIRLINE_MQ', 'AIRLINE_NK', 'AIRLINE_OO', 'AIRLINE_UA',
       ...
       'DESTINATION_AIRPORT_WYS', 'DESTINATION_AIRPORT_XNA',
       'DESTINATION_AIRPORT_YAK', 'DESTINATION_AIRPORT_YUM',
       'DAY_OF_WEEK_MONDAY', 'DAY_OF_WEEK_SATURDAY', 'DAY_OF_WEEK_SUNDAY',
       'DAY_OF_WEEK_THURSDAY', 'DAY_OF_WEEK_TUESDAY', 'DAY_OF_WEEK_WEDNESDAY'],
      dtype='object', length=658)
```


In [25]:
```
df.columns
```

Out[25]:
```
Index(['MONTH', 'DAY', 'DAY_OF_WEEK', 'AIRLINE', 'ORIGIN_AIRPORT',
       'DESTINATION_AIRPORT', 'SCHEDULED_DEPARTURE', 'DEPARTURE_DELAY',
       'DISTANCE', 'ARRIVAL_DELAY'],
      dtype='object')
```

In [26]:
```
flights.columns
```

Out[26]:
```
Index(['MONTH', 'DAY', 'DAY_OF_WEEK', 'AIRLINE', 'ORIGIN_AIRPORT',
       'DESTINATION_AIRPORT', 'SCHEDULED_DEPARTURE', 'DEPARTURE_DELAY',
       'DISTANCE', 'ARRIVAL_DELAY'],
      dtype='object')
```

In [27]:
```
var_to_remove=["DAY_OF_WEEK","AIRLINE","ORIGIN_AIRPORT","DESTINATION_AIRPORT"]
df.drop(var_to_remove,axis=1,inplace=True)
df
```

Out[27]:

	MONTH	DAY	SCHEDULED_DEPARTURE	DEPARTURE_DELAY	DISTANCE	ARRIVAL_DELAY
4330572	9	27	1515	7.0	496	4.0
2153991	5	17	1430	4.0	2343	3.0
2268611	5	24	1655	-9.0	978	-9.0
5344954	12	1	1025	53.0	2343	44.0
1728777	4	21	800	-8.0	937	-19.0
...	...	...	...	...	...	...
3542391	8	8	1955	82.0	954	91.0
3777973	8	23	838	-1.0	574	0.0
4002231	9	6	1010	1.0	1910	-15.0
1143520	3	16	730	-2.0	2139	-11.0
5414693	12	5	1855	-7.0	936	-31.0

98197 rows × 6 columns

In [28]:
```
data=pd.concat([df,df_cat],axis=1)
data
```

Out[28]:

	MONTH	DAY	SCHEDULED_DEPARTURE	DEPARTURE_DELAY	DISTANCE	ARRIVAL_DELAY	AIRLINE_AS	AIRLINE_B6	AIRLINE_DL	AIRLINE_EV	...
4330572	9	27	1515	7.0	496	4.0	0	1	0	0	...
2153991	5	17	1430	4.0	2343	3.0	0	1	0	0	...
2268611	5	24	1655	-9.0	978	-9.0	1	0	0	0	...
5344954	12	1	1025	53.0	2343	44.0	0	0	0	0	...
1728777	4	21	800	-8.0	937	-19.0	0	0	0	0	...
...	...	...	...	...	...	...	...	...	...	...	...
3542391	8	8	1955	82.0	954	91.0	1	0	0	0	...
3777973	8	23	838	-1.0	574	0.0	0	0	0	0	...
4002231	9	6	1010	1.0	1910	-15.0	0	0	0	0	...
1143520	3	16	730	-2.0	2139	-11.0	0	0	1	0	...
5414693	12	5	1855	-7.0	936	-31.0	0	0	0	0	...

98197 rows × 664 columns

In [29]:
```
data.shape
```

Out[29]: (98197, 664)

In [30]:
```
final_data = data.sample(n=60000)
final_data
```

Out[30]:

	MONTH	DAY	SCHEDULED_DEPARTURE	DEPARTURE_DELAY	DISTANCE	ARRIVAL_DELAY	AIRLINE_AS	AIRLINE_B6	AIRLINE_DL	AIRLINE_EV	...
3194391	7	19	1133	-8.0	1008	-32.0	0	0	0	1	...
3403025	7	31	1500	5.0	3417	-10.0	0	0	1	0	...
1986115	5	7	630	-5.0	888	-16.0	0	0	0	0	...
435889	1	29	1525	36.0	642	25.0	0	0	0	0	...


```
X=final_data.drop("DEPARTURE_DELAY",axis=1)
Y=final_data.DEPARTURE_DELAY
```

```
X
```

	MONTH	DAY	SCHEDULED_DEPARTURE	DISTANCE	ARRIVAL_DELAY	AIRLINE_AS	AIRLINE_B6	AIRLINE_DL	AIRLINE_EV	AIRLINE_F9	...	DESTINA
3194391	7	19	1133	1008	-32.0	0	0	0	1	0	...	
3403025	7	31	1500	3417	-10.0	0	0	1	0	0	...	
1986115	5	7	630	888	-16.0	0	0	0	0	0	...	
435889	1	29	1525	642	25.0	0	0	0	0	0	...	
2612747	6	14	1525	971	-22.0	0	0	0	0	0	...	
...	...	...	...	...	...	...	...	...	...	...	...	
3366975	7	29	1345	140	-7.0	0	0	0	1	0	...	
627702	2	11	1310	594	-16.0	0	0	0	0	0	...	
530792	2	5	615	1065	-24.0	0	1	0	0	0	...	
3161417	7	17	1020	406	-15.0	0	0	0	0	0	...	
1560546	4	10	1529	1605	-10.0	0	0	0	0	0	...	

60000 rows × 663 columns

```
Y
```

```
3194391    -8.0
3403025     5.0
1986115    -5.0
435889     36.0
2612747    -4.0
           ...
3366975    -1.0
627702     -5.0
530792     -7.0
3161417    -8.0
```

```
3194391    -8.0
3403025     5.0
1986115    -5.0
435889     36.0
2612747    -4.0
           ...
3366975    -1.0
627702     -5.0
530792     -7.0
3161417    -8.0
1560546     8.0
Name: DEPARTURE_DELAY, Length: 60000, dtype: float64
```

```
from sklearn.model_selection import train_test_split, cross_val_score, cross_val_predict
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, random_state=0)
```

```
from sklearn.ensemble import RandomForestRegressor
reg_rf = RandomForestRegressor()
reg_rf.fit(X_train,y_train)
```

```
RandomForestRegressor()
```

```
y_pred = reg_rf.predict(X_test)
```

```
reg_rf.score(X_train,y_train)
```

```
0.9884289222356428
```

```
reg_rf.score(X_test,y_test)
```

```
0.9248202523874065
```

```
metrics.r2_score(y_test,y_pred)
```

```
0.9248202523874065
```



```
pp=pd.DataFrame({'Actual':y_test,'Predicted':y_pred})
pp
```

	Actual	Predicted
5648606	5.0	-0.08
1190313	89.0	78.74
177785	-3.0	-1.34
225285	0.0	1.88
2814995	-16.0	-0.17
...	...	...
3071475	-5.0	-5.51
378775	-7.0	-3.29
2913785	8.0	31.97
3023908	-4.0	-3.49
1468738	-5.0	-2.38

12000 rows × 2 columns

```
from sklearn.model_selection import RandomizedSearchCV
#Randomized Search CV

# Number of trees in random forest
n_estimators = [int(x) for x in np.linspace(start = 10, stop = 200, num = 12)]
# Number of features to consider at every split
max_features = ['auto', 'sqrt']
# Maximum number of levels in tree
max_depth = [int(x) for x in np.linspace(5, 30, num = 6)]
# Minimum number of samples required to split a node
min_samples_split = [2, 5, 10, 15, 100]
# Minimum number of samples required at each leaf node
min_samples_leaf = [1, 2, 5, 10]
```



```
# Create the random grid

random_grid = {'n_estimators': n_estimators,
               'max_features': max_features,
               'max_depth': max_depth,
               'min_samples_split': min_samples_split,
               'min_samples_leaf': min_samples_leaf}
```

```
# Random search of parameters, using 5 fold cross validation,search across 100 different combinations
rf_random = RandomizedSearchCV(estimator = reg_rf, param_distributions = random_grid,scoring='neg_mean_squared_error', n_iter = 
```

```
rf_random.fit(X_train,y_train)
```

```
Fitting 5 folds for each of 10 candidates, totalling 50 fits
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=5, min_samples_split=5, n_estimators=148; total time=   4.5s
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=5, min_samples_split=5, n_estimators=148; total time=   4.8s
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=5, min_samples_split=5, n_estimators=148; total time=   4.5s
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=5, min_samples_split=5, n_estimators=148; total time=   4.5s
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=5, min_samples_split=5, n_estimators=148; total time=   5.2s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=2, min_samples_split=10, n_estimators=182; total time=  10.2s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=2, min_samples_split=10, n_estimators=182; total time=   9.5s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=2, min_samples_split=10, n_estimators=182; total time=   8.7s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=2, min_samples_split=10, n_estimators=182; total time=   9.5s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=2, min_samples_split=10, n_estimators=182; total time=   9.0s
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_split=100, n_estimators=44; total time=  38.5s
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_split=100, n_estimators=44; total time=  36.6s
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_split=100, n_estimators=44; total time=  36.7s
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_split=100, n_estimators=44; total time=  37.7s
```

```
rf_random.best_params_
```

```
{'n_estimators': 113,
 'min_samples_split': 15,
 'min_samples_leaf': 1,
 'max_features': 'auto',
 'max_depth': 20}
```

```
p=rf_random.predict(X_test)
```

```
metrics.r2_score(y_test,p)
```

```
0.9250107343678358
```

```
print('MAE:', metrics.mean_absolute_error(y_test,p))
print('MSE:', metrics.mean_squared_error(y_test,p))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test,p)))
```

```
MAE: 5.965876005891397
MSE: 107.43791945480484
RMSE: 10.365226454583848
```

```
zz=pd.DataFrame({'Actual':y_test,'Predicted':p})
zz
```

	Actual	Predicted
5648606	5.0	-0.709323
1190313	89.0	77.249742
177785	-3.0	-1.227238
225285	0.0	1.782757
2814995	-16.0	-2.132059
...	...	...
3071475	-5.0	-4.382068
378775	-7.0	-3.076641


```
from sklearn.ensemble import GradientBoostingRegressor
gbr=GradientBoostingRegressor(random_state=0)
```

```
GBR=gbr.fit(X_train,y_train)
pre=GBR.predict(X_test)
```

```
print('MAE:', metrics.mean_absolute_error(y_test,pre))
print('MSE:', metrics.mean_squared_error(y_test,pre))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test,pre)))
```

```
MAE: 5.965703796197842
MSE: 101.54180172868605
RMSE: 10.076795211211055
```

```
metrics.r2_score(y_test,pre)
```

```
0.9291260927125069
```

```
gg=pd.DataFrame({'Actual':y_test,'Predicted':pre})
gg
```

	Actual	Predicted
5648606	5.0	0.392461
1190313	89.0	72.110190
177785	-3.0	-0.602797
225285	0.0	1.300160
2814995	-16.0	-2.792184
...	...	...
3071475	-5.0	-3.240711
378775	-7.0	-4.056999


```
def predict(MONTH, DAY,SCHEDULED_DEPARTURE,DISTANCE, ARRIVAL_DELAY,AIRLINE,ORIGIN_AIRPORT,DESTINATION_AIRPORT,DAY_OF_WEEK):
    AIRLINE_index = np.where(X.columns==AIRLINE)[0][0]
    ORIGIN_index = np.where(X.columns==ORIGIN_AIRPORT)[0][0]
    DESTINATION_index = np.where(X.columns==DESTINATION_AIRPORT)[0][0]
    DAY_OF_WEEK_index = np.where(X.columns==DAY_OF_WEEK)[0][0]
    x= np.zeros(len(X.columns))
    x[0] = MONTH
    x[1] = DAY
    x[2] = SCHEDULED_DEPARTURE
    x[3] = DISTANCE
    x[4] = ARRIVAL_DELAY
    if AIRLINE_index >=0:
        x[AIRLINE_index] = 1
    if ORIGIN_index >=0:
        x[ORIGIN_index] = 1
    if DESTINATION_index >=0:
        x[DESTINATION_index] = 1
    if  DAY_OF_WEEK_index >= 0:
        x[ DAY_OF_WEEK_index] = 1

    return gbr.predict([x])[0]
```

```
res= predict(5,6,1515,328,-8.0,'AIRLINE_OO','ORIGIN_AIRPORT_PHX','DESTINATION_AIRPORT_ABQ','DAY_OF_WEEK_TUESDAY')
res
```

```
C:\Users\Devi\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X does not have valid feature names, but GradientBo
ostingRegressor was fitted with feature names
  warnings.warn(
```

```
-3.9007748765834256
```

```
if(res<=-15):
  print("Flight is delayed")
else:
  print("Flight is not delayed")
```

```
Flight is not delayed
```

Design code:

First 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>Mainentry Form</title>
   <link rel="stylesheet" href="C:\Users\Devi\Documents\Flight
Design\Mainentry.css">
   <link href="https://fonts.googleapis.com/icon?family=Material+Icons"
rel="stylesheet">


</head>
<body style="background-color: aliceblue;">
   <table>
   <tr>

       <td><h1 style="font-family: 'Roboto', sans-serif;">Enjoy the Journey and
Embarce the destination when yoy get there</h1></td>
   </tr>
   <tr>
       <td><image src="C:\Users\Devi\Downloads\fxbluvvrng20zfsk8xoc.gif"
height="500" width="800"></image></td>
       <td><h1 style="color: rgba(130, 179, 238, 0.819);">Welcome!!! <br>Flight
Delay Prediction <img src="C:\Users\Devi\Downloads\black-
24dp\2x\outline_flight_takeoff_black_24dp.png"></h1>
           <br><h2><a href="C:\Users\Devi\Documents\Application Building\Build
HTML Pages\Flight Design\Register.html" style="text-decoration: none;
background-color:
azure;">Register</a>        &
nbsp;  
                      <a
href="C:\Users\Devi\Documents\Application Building\Build HTML
Pages\Flight Design\Login.html" style="text-decoration: none; background-color:
azure;">Login</a></h2>
       </td>
   </tr>
   </table>


</body>
</html>
```

Register 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>Registration Form</title>

   <script src="C:\Users\Devi\Documents\Application Building\Build HTML
Pages\Flight Design\app.js"></script>
   <style>
      @import
url('https://fonts.googleapis.com/css2?family=Montserrat:wght@200;300&fami
ly=Roboto&display=swap');
      @import
url('https://fonts.googleapis.com/css2?family=Are+You+Serious&family=Monts
errat:wght@200&family=Roboto&display=swap');
   </style>
</head>
<body background="C:\Users\Devi\Pictures\reg.jpg">
  <div class="container" style="text-align: center;">
   <div class="header">
      <h1 style="font-family: 'Roboto', sans-serif; color:rgba(122, 104, 80,
0.992);">Registration Form</h1>
   </div>
  </div>
  <div class="form" style="text-align: center;"><br><br>
   <div class="form-group">
      <label for="" style="font-family: 'Are You Serious', cursive; font-size:
50px;">UserName :</label>
      <input  type="text" placeholder="username" id="username"
autocomplete="off" required>

   </div>
   <div>
      <label for="" style="font-family: 'Are You Serious', cursive; font-size:
50px;">Password :</label>
      <input type="password" placeholder="password" id="password"
autocomplete="off" required>
      <i class="ion-ios-checkmark"></i>
      <i class="ion-android-alert"></i>
   </div>
   <div>
      <label for="" style="font-family: 'Are You Serious', cursive; font-size:
50px;">Confirm Password :</label>
```

```
        <input type="password" placeholder="confirm password" 
id="confirmpassword" autocomplete="off" required>

    </div>
    <div>
        <label for="" style="font-family: 'Are You Serious', cursive; font-size: 
50px;">Email :</label>
        <input type="email" placeholder="Mail id" id="email" autocomplete="off">
        <i class="ion-ios-checkmark"></i>
        <i class="ion-android-alert"></i>
    </div>
    <div>
        <label for="" style="font-family: 'Are You Serious', cursive; font-size: 
50px;">Phone Number :</label>
        <input type="number" placeholder="must be 10 numbers" 
id="phonenumber" autocomplete="off">

    </div><br><br>
    <input type=button  value="submit">
   </div>

</body>
</html>
```

Login 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>Loign Form</title>
    <style>
        @import 
url('https://fonts.googleapis.com/css2?family=Montserrat:wght@200;300&fami
ly=Roboto&display=swap');
        @import 
url('https://fonts.googleapis.com/css2?family=Are+You+Serious&family=Monts
errat:wght@200&family=Roboto&display=swap');
    </style>
</head>
<body background="C:\Users\Devi\Downloads\takeoff.gif">
    <div class="container" style="text-align: center;">
        <div class="header">
            <h1 style="font-family: 'Roboto', sans-serif; color:rgba(122, 104, 80, 
0.992);">Login Form</h1>
        </div><br><br><br>
        <div class="form" style="text-align: center;"><br><br>
```

```
        <div class="form-group">
          <label for="" style="font-family: 'Are You Serious', cursive; font-size:
50px;">UserName</label>
          <input  type="text" placeholder="username" id="username"
autocomplete="off" required>
        </div>
      </div>
      <div>
        <label for="" style="font-family: 'Are You Serious', cursive; font-size:
50px;">Password</label>
        <input type="password" placeholder="password" id="password"
autocomplete="off" required>
      </div><br><br><br><br>
      <input type=button value="submit">
    </div>


</body>
</html>
```

Main 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>Home Page</title>
  <style>
    @import
url('https://fonts.googleapis.com/css2?family=Are+You+Serious&family=Bunge
e+Shade&family=Montserrat:wght@200&family=Roboto&display=swap');
    @import
url('https://fonts.googleapis.com/css2?family=Are+You+Serious&family=Bunge
e+Shade&family=Monoton&family=Montserrat:wght@200&family=Roboto&dis
play=swap');
    @import
url('https://fonts.googleapis.com/css2?family=Are+You+Serious&family=Bunge
e+Shade&family=Kolker+Brush&family=Monoton&family=Montserrat:wght@20
0&family=Roboto&display=swap');
    @import
url('https://fonts.googleapis.com/css2?family=Are+You+Serious&family=Bunge
e+Shade&family=Island+Moments&family=Monoton&family=Montserrat:wght@
200&family=Nabla&family=Roboto&family=Rubik+Bubbles&display=swap');
    </style>
</head>
<body style="background-color: azure;">
```

```
    <h1 style="font-family: 'Bungee Shade', cursive; text-align: center; color:
rgb(147, 66, 137); background-color: rgba(0, 0, 255, 0.334);">Flight Delay
Prediction</h1><br>
    <h2 style="text-align:right ;"><a href="C:\Users\Devi\Documents\Application
Building\Build HTML Pages\Flight Design\prediction.html" style="color:
rgba(123, 51, 51, 0.919); font-family: 'Rubik Bubbles',
cursive;">Prediction</a></h2>
    <h1 style="font-family: 'Monoton', cursive; color:  rgba(49, 51, 126,
0.536);">Our Mission</h1>
    <h2 style="font-family: 'Kolker Brush', cursive; font-size: 50px; color:rgb(188,
142,
80)">          &n
bsp;          The
basic problem of this exercise is to create a model capable of predicting the
arrival delay time of a commercial flight, given a set of parameters known at time
of take-off.</h2><br>

    <h1 style="font-family: 'Monoton', cursive; color: rgba(49, 51, 126,
0.536);">Our Vision</h1>
    <h2 style="font-family: 'Kolker Brush', cursive; font-size: 50px; color:rgb(188,
142,
80)">          &n
bsp;          Our
Goal was to Predict Flight delay accurately to our customer. To satisfies our
customer. It's our ,Main  Vision.</h2><br>


</body>
</html>
```

Prediction 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="C:\Users\Devi\Documents\Application Building\Build HTML
Pages\Flight Design\app.js"></script>
    <title>Prediction Page</title>
    <style>
        @import
url('https://fonts.googleapis.com/css2?family=Are+You+Serious&family=Bunge
e+Shade&family=Monoton&family=Montserrat:wght@200&family=Nabla&famil
y=Roboto&display=swap');
        @import
url('https://fonts.googleapis.com/css2?family=Are+You+Serious&family=Bunge
```

```
e+Shade&family=Island+Moments&family=Monoton&family=Montserrat:wght@
200&family=Nabla&family=Roboto&display=swap');
    </style>
</head>
<body style="background-color: white;">
  <h1 style="text-align: center; font-family: 'Nabla', cursive; background-color:
rgba(164, 59, 183, 0.693);">Prediction of Flight Delay</h1>
  <div>
  <h3>
    <img src="C:\Users\Devi\Downloads\1_8o-Xy_yo9G3K0I75QfzlaQ.gif"
width="500" height="650" align="right">
  </h3>
  <div text-align="left">
    <label for="" style="font-family: 'Island Moments', cursive; font-size: 50px;
color: rgb(232, 85, 85);">Flight_Number</label>
    <input type="text" id="flightnnumber" autocomplete="off">
  </div>

  <div text-align="left">
    <label for="" style="font-family: 'Island Moments', cursive; font-size: 50px;
color: rgb(232, 85, 85);">Month</label>
    <input type="text" id="month" autocomplete="off">
  </div>
  <div text-align="left">
    <label for="" style="font-family: 'Island Moments', cursive; font-size: 50px;
color: rgb(232, 85, 85);">Day</label>
    <input type="text" id="day" autocomplete="off">
  </div>
  <div text-align="left">
    <label for="" style="font-family: 'Island Moments', cursive; font-size: 50px;
color: rgb(232, 85, 85);">sch_dept</label>
    <input type="text" id="sch_dept" autocomplete="off">
  </div>
  <div text-align="left">
    <label for="" style="font-family: 'Island Moments', cursive; font-size: 50px;
color: rgb(232, 85, 85);">distance</label>
    <input type="text" id="distance" autocomplete="off">
  </div>
  <div text-align="left">
    <label for="" style="font-family: 'Island Moments', cursive; font-size: 50px;
color: rgb(232, 85, 85);">arrival_delay</label>
    <input type="text" id="arrival_delay" autocomplete="off">
  </div>
  <div text-align="left">
    <label for="" style="font-family: 'Island Moments', cursive; font-size: 50px;
color: rgb(232, 85, 85);">airline</label>
    <input type="text" id="airline" autocomplete="off">
  </div>
  <div text-align="left">
```

```
        <label for="" style="font-family: 'Island Moments', cursive; font-size: 50px;
color: rgb(232, 85, 85);">origin</label>
        <input type="text" id="origin" autocomplete="off">
    </div>
    <div text-align="left">
        <label for="" style="font-family: 'Island Moments', cursive; font-size: 50px;
color: rgb(232, 85, 85);">destination </label>
        <input type="text" id="destination" autocomplete="off">
    </div>
    <div text-align="left">
        <label for="" style="font-family: 'Island Moments', cursive; font-size: 50px;
color: rgb(232, 85, 85);"> day_of_week </label>
        <input type="text" id="day_of_week" autocomplete="off">
    </div><br><br>


        <input type=button  value="submit">




</body>
</html>
```

app.py

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


model=pickle.load(open('flightclf.pkl','rb'))


app=Flask(__name__)

@app.route('/')
def index():
    return render_template('index.html')

@app.route('/prediction',methods=["POST"])
def predict():
    if request.method=="POST":
        name=request.form["name"]
        month=request.form["month"]
        if(int(month)>12):
            ans="Please Enter the correct Month"
            return render_template("index.html" ,y=ans)
```

```
dayofmonth=request.form["dayofmonth"]
if(int(dayofmonth)>31):
    ans="Please Enter the correct Day of Month"
    return render_template("index.html" ,y=ans)

dayofweek=request.form["dayofweek"]
if(int(dayofweek)>7):
    ans="Please Enter the correct Day of Week"
    return render_template("index.html" ,y=ans)


origin=request.form["origin"]
destination=request.form['destination']

if(origin==destination):
    ans="Origin airport and destination airport can't be same"
    return render_template("index.html" ,y=ans)

if(origin=="msp"):
    origin1,origin2,origin3,origin4,origin5=0,0,0,1,0
if(origin=="dtw"):
    origin1,origin2,origin3,origin4,origin5=0,1,0,0,0
if(origin=="jfk"):
    origin1,origin2,origin3,origin4,origin5=0,0,1,0,0
if(origin=="sea"):
    origin1,origin2,origin3,origin4,origin5=0,0,0,0,1
if(origin=="alt"):
    origin1,origin2,origin3,origin4,origin5=1,0,0,0,0



if(destination=="msp"):
    destination1,destination2,destination3,destination4,destination5=0,0,0,1,0
if(destination=="dtw"):
    destination1,destination2,destination3,destination4,destination5=0,1,0,0,0
if(destination=="jfk"):
    destination1,destination2,destination3,destination4,destination5=0,0,1,0,0
if(destination=="sea"):
    destination1,destination2,destination3,destination4,destination5=0,0,0,0,1
if(destination=="alt"):
   destination1,destination2,destination3,destination4,destination5=1,0,0,0,0

depthr=request.form['depthr']
deptmin=request.form['deptmin']
if(int(depthr)>23 or int(deptmin)>59):
    ans="Please enter the correct Departure time"
    return render_template("index.html" ,y=ans)
else:
    dept=depthr+deptmin
```

```
        actdepthr=request.form['actdepthr']
        actdeptmin=request.form['actdeptmin']
        if(int(actdepthr)>23 or int(actdeptmin)>59):
            ans="Please enter the correct Actual Departure time"
            return render_template("index.html" ,y=ans)
        else:
            actdept=actdepthr+actdeptmin



        arrtimehr=request.form['arrtimehr']
        arrtimemin=request.form['arrtimemin']
        if(int(arrtimehr)>23 or int(arrtimemin)>59):
            ans="Please enter the correct Arrival time"
            return render_template("index.html" ,y=ans)
        else:
            arrtime=arrtimehr+arrtimemin


        if((int(actdept)-int(dept))<15):
            dept15=0
        else:
            dept15=1

        print(dept15)

total=[[month,dayofmonth,dayofweek,origin1,origin2,origin3,origin4,origin5,des
tination1,destination2,destination3,destination4,destination5,dept,actdept,dept1
5,arrtime]]

        value=model.predict(total)
        print(value)
        if(value==[0.]):
            ans="THE FLIGHT WILL BE ON TIME"
        else:
            ans="THE FLIGHT WILL BE DELAYED"

    return render_template("results.html" ,y=ans)


if __name__=="__main__":
    app.run(debug=False)
```

GitHup Link : IBM-EPBL/IBM-Project-44302-1668781535

Project Demo Link : https://drive.google.com/file/d/1D8-wzZ9uDEsVWoegyphFOYRiVz-FUmqF/view?usp=sharing