|  |  |  |
| --- | --- | --- |
| **TABLE OF CONTENTS**      **TOPIC** | | **PAGE NO** |
| **Abstract** | | v |
| **List of Figures** | | vii |
| **List of Screenshots** | | vii |
| **1. INTRODUCTION** | | 1 |
| 1.1 Further Enhancement | | 2 |
| **2. LITERATURE SURVEY** | | 3 |
| **3. REQUIREMENT** | | 5 |
| 3.1 Requirement Analysis | | 6 |
| 3.2 Requirement Specification | | 6 |
| **4. INPUT AND OUTPUT DESIGN** | | 7 |
| 4.1 Input Design | | 8 |
| 4.1.1 Objectives | | 8 |
| 4.2 Output Design | | 8 |
| **5. MODULES** | | 9 |
| 5.1 Implementation | | 10 |
| 5.2 Modules Description | | 10 |
| 5.3 Model Execution | | 10 |
| **6. SOFTWARE ENVIRONMENT** | | 11 |
| 6.1 Python |  | 12 |
| 6.1.1 | Python Identifiers | 12 |
| 6.1.2 | Lines and Indentation | 12 |
| 6.1.3 | Command Line Arguments | 13 |
| 6.1.4 | Python Lists | 14 |
| 6.1.5 | Properties of Dictionary Keys | 14 |
| 6.2 Django | | 15 |
| 6.2.1 Create a Project | | 16 |
| 6.2.2 Create an Application | | 17 |

|  |  |
| --- | --- |
| **7. SOURCE CODE** | 19 |
| **8. SYSTEM ANALYSIS** | 24 |
| 8.1 Existing System | 25 |
| 8.2 Proposed System | 25 |
| **9. SYSTEM DESIGN** | 26 |
| 9.1 System Architecture | 27 |
| 9.2 Data Flow Diagram | 27 |
| 9.3 Uml Diagrams | 27 |
| 9.4 Use Case Diagram | 28 |
| 9.5 Class Diagram | 28 |
| **10. SYSTEM STUDY** | 29 |
| 10.1 Feasibility Study | 30 |
| 10.1.1 Economical Feasibility | 30 |
| 10.1.2 Technical Feasibility | 30 |
| 10.1.3 Social Feasibility | 30 |
| **11. SYSTEM TEST** | 31 |
| 11.1 Test strategy and approach | 32 |
| **12. TEST CASES** | 33 |
| **13. SCREEN SHOTS** | 35 |
| **14. CONCLUSION** | 39 |
| **15. REFERENCES** | 41 |

**ABSTRACT**

### Predicting Flight Delays with Error Calculation

Flight delay is a major problem in the aviation sector. During the last two decades, the growth of the aviation sector has caused air traffic congestion, which has caused flight delays. Flight delays result not only in the loss of fortune also negatively impact the environment. Flight delays also cause significant losses for airlines operating commercial flights. Therefore, they do everything possible in the prevention or avoidance of delays and cancellations of flights by taking some measures. In this paper, using machine learning models such as Logistic Regression, Decision Tree Regression, Bayesian Ridge, Random Forest Regression and Gradient Boosting Regression we predict whether the arrival of a particular flight will be delayed or not.

|  |  |  |  |
| --- | --- | --- | --- |
| **List of Figures**  **TOPIC** |  |  | **PAGE NO** |
| Django Web Structure |  |  | 15 |
| Django Administration Process |  |  | 16 |
| System Architecture |  |  | 27 |
| Data Flow Diagram |  |  | 27 |
| Use Case Diagram |  |  | 28 |
| Class Diagram |  |  | 28 |
| **List of Screenshots**  **TOPIC** |  |  | **PAGE NO** |
| Home page |  |  | 36 |
| Register Form |  |  | 36 |
| User Login Form |  |  | 36 |
| User Home Page |  |  | 36 |
| Preprocessed Data |  |  | 37 |
| Algorithm codes |  |  | 37 |
| Admin Login page |  |  | 37 |
| Admin home Page |  |  | 37 |
| Admin Adding Data |  |  | 38 |
| Admin ViewData |  |  | 38 |
| Admin View Results |  |  | 38 |
| Server Side results |  |  | 38 |

# INTRODUCTION

Statistical modelling is a mathematical way of making approximations from input data. These approximations are then used to make predictions. Statistical models help in predicting the future probabilistic behavior of a system based on past statistical data. Predictive modelling has been used in many fields, for example in crime cases to detect the likeliness of an email being spam and flight delays. In evaluation of how different models perform in modelling of flight delays, regression models have been found efficient in predicting flight delays since they highlighted the various causes of flight delays . However, they could not categorize complex data. Econometric models have been used to model scheduled flight cancellation and to show how delays from one airport were propagated to other destinations. These models did not provide a complete vindication since they ignored variables that were difficult to quantify. When subjected to social-economic situations, the models showed discriminative and subjective results. Among the models used, random forest has been found to have superior performance. Prediction accuracy may vary due to factors such as time of forecast and airline dynamics. A developed multiple regression model has shown that distance, day and scheduled departure are key factors in predicting flight delay.

## FURTHER ENHANCEMENT

The future scope of this paper can include the application of more advanced, modern and innovative preprocessing techniques, automated hybrid learning and sampling algorithms, and deep learning models adjusted to achieve better performance. To evolve a predictive model, additional variables can be introduced. e.g., a model where meteorological statistics are utilized in developing error-free models for flight delays. In this paper we used data from the US only, therefore in future, the model can be trained with data from other countries as well. With the use of models that are complex and hybrid of many other models provided with appropriate processing power and with the use of larger detailed datasets, more accurate predictive models can be developed. Additionally, the model can be configured for other airports to predict their flight delays as well and for that data from these airports would be required to incorporate into this research.

# LITERATURE SURVEY

**LITERATURE SURVEY**

## 1) Capacity And Delay Analysis For Airport Manoeuvring Areas Using Simulation

### AUTHORS: E. Cinar, F. Aybek, A. Caycar, C. Cetek

To investigate the air traffic flow in a highly complex system such as an airport manoeuvring area, a two-stage method based on fast- and real-time simulation techniques is applied. The first stage involves the analysis with fast- and real-time simulations of a baseline model created to determine the congestion points. Based on the analysis, improvements to be performed in the layout of the manoeuvring area are proposed. In the second stage, alternative scenarios implementing these improvements are generated and evaluated in a fast-time simulation environment. Based on the results of simulations of different runway configurations, the main areas of congestion in the baseline airport model are determined. Congestion nodes are identified in the departure queue points and in the taxiway system.

## 2) Flight Arrival Delay Prediction Using Gradient Boosting Classifier

### AUTHORS: Navoneel, et al., Chakrabarty

The basic objective of the proposed work is to analyse arrival delay of the flights using data mining and four supervised machine learning algorithms: random forest, Support Vector Machine (SVM), Gradient Boosting Classifier (GBC) and k-nearest neighbour algorithm, and compare their performances to obtain the best performing classifier. To train each predictive model, data has been collected from BTS, United States Department of Transportation. The data included all the flights operated by American Airlines, connecting the top five busiest airports of United States, located in Atlanta, Los Angeles, Chicago, Dallas/Fort Worth, and New York, in the years 2015 and 2016. Aforesaid supervised machine learning algorithms were evaluated to predict the arrival delay of individual scheduled flights.

## 3) Prediction Of Weatherinduced Airline Delays Based On Machine Learning Algorithms

### AUTHORS : Y. J. Kim, S. Briceno, D. Mavris, Sun Choi

The primary goal of the model proposed in this paper is to predict airline delays caused by inclement weather conditions using data mining and supervised machine learning algorithms. US domestic flight data and the weather data from 2005 to 2015 were extracted and used to train the model. To overcome the effects of imbalanced.

# REQUIREMENT

**REQUIREMENT ANALYSIS**

The project involved analyzing the design of few applications so as to make the application more users friendly. To do so, it was really important to keep the navigations from one screen to the other well ordered and at the same time reducing the amount of typing the user needs to do. In order to make the application more accessible, the browser version had to be chosen so that it is compatible with most of the Browsers.

## REQUIREMENT SPECIFICATION

**Functional Requirements**

 Graphical User interface with the User.

### Software Requirements

For developing the application the following are the Software Requirements:

1. Python
2. Django

**Operating Systems supported**

1. Windows 10 64 bit OS

**Technologies and Languages used to Develop**

1. Python

### Debugger and Emulator

* Any Browser (Particularly Chrome) **Hardware Requirements**

For developing the application the following are the Hardware Requirements:

* Processor: Intel i3
* RAM: 4 GB

Space on Hard Disk: minimum 1 TB

# INPUT AND OUTPUT DESIGN

**INPUT DESIGN**

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

* What data should be given as input?
* How the data should be arranged or coded?
* The dialog to guide the operating personnel in providing input.
* Methods for preparing input validations and steps to follow when error occur.

**OBJECTIVES**

1. Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.
2. It is achieved by creating user-friendly screens for the data entry to handle large volume of data.
3. When the data is entered it will check for its validity. Data can be entered with the help of screens. Thus the objective of input design is to create an input layout that is easy to follow

**OUTPUT DESIGN**

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output.

1. Designing computer output should proceed in an organized, well thought out manner; the right output must be developed while ensuring that each output element is designed so that people will find the system can use easily and effectively.

2.Select methods for presenting information.

3.Create document, report, or other formats that contain information produced by the system.

# MODULES

**IMPLEMENTATION:**

**MODULES:**

* User  Admin
* Data preprocess
* Model Execution

**MODULES DESCRIPTION:**

**User:**

The User can register the first. While registering he required a valid user email and mobile for further communications. Once the user register then admin can activate the customer.The user can see the results in the browser. The all error scores displayed and graphical representation can be displayed.

**Admin:**

Admin can login with his credentials. Once he login he can activate the users. The activated user only login in our applications. We have studied from various sources to find out which parameters will be most appropriate to predict the departure and arrival delays. After several searches we conclude the dataset parameters are Day, Departure Delay, Airline, Flight Number, Destination Airport, Origin Airport, Day of Week, Taxi out. So this data we consider for further process.

**Data Preprocess:**

The admin provided data has been stored in the sqlite database. To process our methodology we need to perform data cleaning process. By using pandas data frame we can fill the missing values with its mean type. Once data cleaned the data will be displayed on the browser.

## Model Execution

Machine learning models such as Logistic Regression, Decision Tree Regression, Bayesian Ridge, Random Forest Regression and Gradient Boosting Regression we predict result. The MSE is appropriate for our regression problems since it is differentiable, contributing to the stability of the algorithms. It also heavily punishes the bigger errors over smaller errors. MAE is a risk providing metric which tells the expected value of the absolute error loss.

# SOFTWARE ENVIRONMENT

## PYTHON

Python is a general-purpose interpreted, interactive, object-oriented, and high- level programming language. An [interpreted language,](https://en.wikipedia.org/wiki/Interpreted_language) Python has a design philosophy that emphasizes code [readability (](https://en.wikipedia.org/wiki/Readability)notably using [whitespace i](https://en.wikipedia.org/wiki/Whitespace_character)ndentation to delimit [code](https://en.wikipedia.org/wiki/Code_block) blocks rather than curly brackets or keywords), and a syntax that allows programmers to express concepts in fewer [lines of code](https://en.wikipedia.org/wiki/Source_lines_of_code)  than might be used in languages such as [C++o](https://en.wikipedia.org/wiki/C%2B%2B)r [Java.](https://en.wikipedia.org/wiki/Java_(programming_language)) It provides constructs that enable clear programming on both small and large scales.

### Python Identifiers

A Python identifier is a name used to identify a variable, function, class, module or other object. An identifier starts with a letter A to Z or a to z or an underscore (\_) followed by zero or more letters, underscores and digits (0 to 9).Python does not allow punctuation characters such as @, $, and % within identifiers. Starting an identifier with a single leading underscore indicates that the identifier is private.

### Lines and Indentation

Python provides no braces to indicate blocks of code for class and function definitions or flow control. Blocks of code are denoted by line indentation, which is rigidly enforced.The number of spaces in the indentation is variable, but all statements within the block must be indented the same amount. For example − if True: print "True"

else:

print "False"

Multi-Line Statements

Statements in Python typically end with a new line. Python does, however, allow the use of the line continuation character (\) to denote that the line should continue. For example total = item\_one + \ item\_two + \ item\_three

Statements contained within the [], {}, or () brackets do not need to use the line continuation character. For example − days = ['Monday', 'Tuesday', 'Wednesday',

'Thursday', 'Friday']

#### Quotation in Python

Python accepts single ('), double (") and triple (''' or """) quotes to denote string literals, as long as the same type of quote starts and ends the string.The triple quotes are used to span the string across multiple lines. For example, all the following are legal – word = 'word'

sentence = "This is a sentence." paragraph = """This is a paragraph. It is made up of multiple lines and sentences."""

Using Blank Lines

A line containing only whitespace, possibly with a comment, is known as a blank line and Python totally ignores it.In an interactive interpreter session, you must enter an empty physical line to terminate a multiline statement.

Waiting for the User #!/usr/bin/python

raw\_input("\n\nPress the enter key to exit.")

Here, "\n\n" is used to create two new lines before displaying the actual line. Once the user presses the key, the program ends. This is a nice trick to keep a console window open until the user is done with an application.

### Command Line Arguments

Many programs can be run to provide you with some basic information about how they should be run. Python enables you to do this with -h −

$ python -h usage: python [option] ... [-c cmd | -m mod | file | -] [arg] ...

Options and arguments (and corresponding environment variables):

1. cmd : program passed in as string (terminates option list)
2. : debug output from parser (also PYTHONDEBUG=x)

-E : ignore environment variables (such as PYTHONPATH)

-h : print this help message and exit

### Python Lists

The list is a most versatile datatype available in Python which can be written as a list of comma-separated values (items) between square brackets. Important thing about a list is that items in a list need not be of the same type.Creating a list is as simple as putting different comma-separated values between square brackets. For example −A tuple is a sequence of immutable Python objects. Creating a tuple is as simple as putting different comma-separated values. Optionally you can put these comma-separated values between parentheses also. For example − tup1 = ('physics', 'chemistry', 1997, 2000);

The empty tuple is written as two parentheses containing nothing − tup1 = ();

Accessing Values in Tuples

To access values in tuple, use the square brackets for slicing along with the index or indices to obtain value available at that index. For example − Live Demo

#!/usr/bin/python

tup1 = ('physics', 'chemistry', 1997, 2000); tup2 = (1, 2, 3, 4, 5, 6, 7 ); print "tup1[0]: ", tup1[0]; print "tup2[1:5]: ", tup2[1:5];

When the above code is executed, it produces the following result − tup1[0]: physics tup2[1:5]: [2, 3, 4, 5]

Updating Tuples

### Properties of Dictionary Keys

Dictionary values have no restrictions. They can be any arbitrary Python object, either standard objects or user-defined objects. However, same is not true for the keys.

There are two important points to remember about dictionary keys −

1. More than one entry per key not allowed. Which means no duplicate key is allowed. When duplicate keys encountered during assignment, the last assignment wins. For example –

Live Demo

#!/usr/bin/python

dict = {'Name': 'Zara', 'Age': 7, 'Name': 'Manni'} print "dict['Name']: ", dict['Name']

When the above code is executed, it produces the following result − dict['Name']: Manni

1. Keys must be immutable. Which means you can use strings, numbers or tuples as dictionary keys but something like ['key'] is not allowed. Following is a simple example Live Demo

#!/usr/bin/python

dict = {['Name']: 'Zara', 'Age': 7} print "dict['Name']: ", dict['Name']

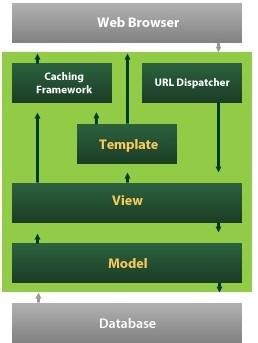
When the above code is executed, it produces the following result − Traceback (most recent call last):

File "test.py", line 3, in <module> dict = {['Name']: 'Zara', 'Age': 7};

TypeError: unhashable type: 'list'

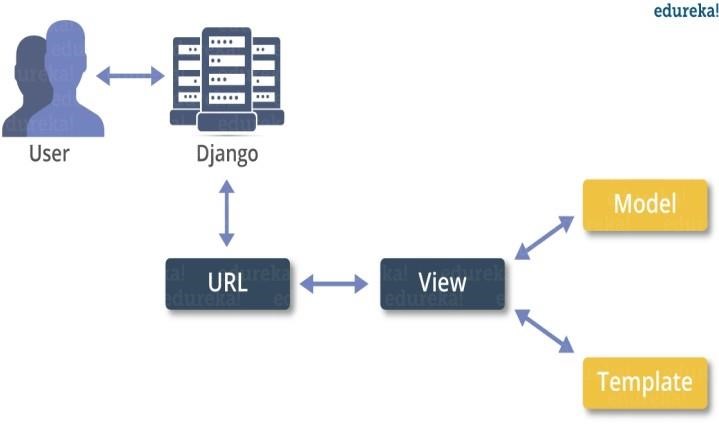
## DJANGO

Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. so you can focus on writing your app without needing to reinvent the wheel. It’s free and open source.



6.1 : Django Web Structure

Django also provides an optional administrative [create, read, update and delete i](https://en.wikipedia.org/wiki/Create%2C_read%2C_update_and_delete)nterface that is generated dynamically through [introspection a](https://en.wikipedia.org/wiki/Introspection_(computer_science))nd configured via admin models



6.2 : Django Administration Process

### Create a Project

Whether you are on Windows or Linux, just get a terminal or a cmd prompt and navigate to the place you want your project to be created, then use this code −

$ django-admin startproject myproject

This will create a "myproject" folder with the following structure − myproject/ manage.py myproject/

init .py settings.py urls.py wsgi.py

The Project Structure

The “myproject” folder is just your project container, it actually contains two elements –

manage.py − This file is kind of your project local django-admin for interacting with your project via command line (start the development server, sync db...). To get a full list of command accessible via manage.py you can use the code –

$ python manage.py help

The “myproject” subfolder − This folder is the actual python package of your project. It contains four files −

init .py − Just for python, treat this folder as package. settings.py − As the name indicates, your project settings.

urls.py − All links of your project and the function to call. A kind of ToC of your project. wsgi.py − If you need to deploy your project over WSGI.

DATABASES = {

'default': {

'ENGINE': 'django.db.backends.sqlite3',

'NAME': 'database.sql',

'USER': '',

'PASSWORD': '',

'HOST': '',

'PORT': '',

}

}

### Create an Application

We assume you are in your project folder. In our main “myproject” folder, the same folder then manage.py –

$ python manage.py startapp myapp

You just created myapp application and like project, Django create a “myapp” folder with the application structure − myapp/ init .py

admin.py models.py tests.py views.py init .py − Just to make sure python handles this folder as a package. admin.py − This file helps you make the app modifiable in the admin interface. models.py − This is where all the application models are stored.

tests.py − This is where your unit tests are. views.py − This is where your application views are.

Get the Project to Know About Your Application

INSTALLED\_APPS = (

'django.contrib.admin',

'django.contrib.auth',

'django.contrib.contenttypes',

'django.contrib.sessions',

'django.contrib.messages',

'django.contrib.staticfiles',

'myapp',

)

myapp/forms.py

#-\*- coding: utf-8 -\*- from django import forms class LoginForm(forms.Form):

user = forms.CharField(max\_length = 100)

password = forms.CharField(widget = forms.PasswordInput())

As seen above, the field type can take "widget" argument for html rendering; in our case, we want the password to be hidden, not displayed. Many others widget are present in Django: DateInput for dates, CheckboxInput for checkboxes, etc.

# SOURCE CODE

## SOURCE CODE

### User side views.py

**from** django.shortcuts **import** render,HttpResponse **from** django.contrib **import** messages **from** users.forms **import** UserRegistrationForm

**from** users.models **import** UserRegistrationModel,FlightDataModel **import** io,csv

**from** django.conf **import** settings

**from** .FlightDataPreproces **import** DPDataPrePRocess **from** .models **import** FlightDataModel **from** django\_pandas.io **import** read\_frame **def** UserRegisterAction(request):

**if** request.method == **'POST'**: form = UserRegistrationForm(request.POST) **if** form.is\_valid(): print(**'Data is Valid'**) form.save()

messages.success(request, **'You have been successfully registered'**) form = UserRegistrationForm()

**return** render(request, **'UserRegister.html'**, {**'form'**: form})

**else**: print(**"Invalid form"**) **else**:

form = UserRegistrationForm()

**return** render(request, **'UserRegister.html'**, {**'form'**: form})

**def** UserLoginCheck(request):

**if** request.method == **"POST"**:

loginid = request.POST.get(**'loginid'**) pswd = request.POST.get(**'pswd'**) print(**"Login ID = "**, loginid, **' Password = '**, pswd) **try**:

check = UserRegistrationModel.objects.get(loginid=loginid,

password=pswd) status = check.status print(**'Status is = '**, status) **if** status == **"activated"**:

request.session[**'id'**] = check.id request.session[**'loggeduser'**] = check.name request.session[**'loginid'**] = loginid request.session[**'email'**] = check.email print(**"User id At"**, check.id, status)

**return** render(request, **'users/UserHome.html'**, {})

**else**:

messages.success(request, **'Your Account Not at activated'**) **return** render(request, **'UserLogin.html'**

**except** Exception **as** e:

print(**'Exception is '**, str(e)) **pass**

messages.success(request, **'Invalid Login id and password'**)

**return** render(request, **'UserLogin.html'**, {})

**def** UserUploadForm(request): **return** render(request,**'users/uploadform.html'**,{})

**def** UserDataUpload(request):

template = **"users/UserHome.html"** data = FlightDataModel.objects.all() prompt = {

**'order'**: **'Order of the CSV should be name, email, address, phone, profile'**,**'profiles'**: data

} **if** request.method == **"GET"**: **return** render(request, template, prompt)

csv\_file = request.FILES[**'file'**] **if not** csv\_file.name.endswith(**'.csv'**): messages.error(request, **'THIS IS NOT A CSV FILE'**)

data\_set = csv\_file.read().decode(**'UTF-8'**) **try**:

io\_string = io.StringIO(data\_set) next(io\_string) **for** column **in** csv.reader(io\_string, delimiter=**','**, quotechar=**"|"**): \_, created = FlightDataModel.objects.update\_or\_create(

DAY = column[1],

DEPARTURE\_TIME= column[2],

FLIGHT\_NUMBER= column[3],

DESTINATION\_AIRPORT= column[4],

ORIGIN\_AIRPORT= column[5],

DAY\_OF\_WEEK= column[6],

TAXI\_OUT= column[7])

**except** Exception **as** ex:

print(**'error at'**, ex)

context = {}

**return** render(request, **'users/UserHome.html'**, context)

**def** DataPreProcessing(request): qs = FlightDataModel.objects.all() dataset = read\_frame(qs) print(**"Dataset "**,dataset) x = DPDataPrePRocess() data = x.process\_data(datasetname = dataset)

**return** render(request, **'users/PreProcessData.html'**,{**'data'**:qs})

**def** UsermachineLearning(request): qs = FlightDataModel.objects.all() dataset = read\_frame(qs) x = DPDataPrePRocess()

lg\_dict = x.MyLogiSticregression(dataset)

*#lg\_dict = {}*

dt\_dict = x.MyDecisionTree(dataset) rf\_dict = x.MyRandomForest(dataset) br\_dict = x.MyBayesianRidge(dataset)

gbr\_dict = x.MyGradientBoostingRegressor(dataset)

**return**

render(request,**'users/UsrMachineLearningRslt.html'**,{**'lg\_dict'**:lg\_dict,**'dt\_dic t'**:dt\_dict,**'rf\_dict'**:rf\_dict,**'br\_dict'**:br\_dict,**'gbr\_dict'**:gbr\_dict}) **def** UserGraphs(request): qs = FlightDataModel.objects.all() dataset = read\_frame(qs) x = DPDataPrePRocess()

lg\_dict = {}

dt\_dict = x.MyDecisionTree(dataset) rf\_dict = x.MyRandomForest(dataset) br\_dict = x.MyBayesianRidge(dataset)

gbr\_dict = x.MyGradientBoostingRegressor(dataset)

**return** render(request, **'users/UserGraphs.html'**,

{**'lg\_dict'**: lg\_dict, **'dt\_dict'**: dt\_dict, **'rf\_dict'**: rf\_dict, **'br\_dict'**: br\_dict,**'gbr\_dict'**: gbr\_dict})

### FlightDataProcess.py

**import** pandas **as** pd

**import** matplotlib.pyplot **as** plt **import** matplotlib matplotlib.use(**'TkAgg'**) **from** sklearn.model\_selection **import** train\_test\_split **from** sklearn.linear\_model **import** LogisticRegression **from** sklearn.tree **import** DecisionTreeRegressor **from** sklearn.metrics **import** accuracy\_score **from** sklearn.ensemble **import** RandomForestRegressor **from** sklearn.linear\_model **import** BayesianRidge **from** sklearn.ensemble **import** GradientBoostingRegressor **from** sklearn **import** metrics **import** numpy **as** np **class** DPDataPrePRocess:

**def** process\_data(self,datasetname):

dataset = datasetname

dataset = dataset[[**'DAY'**, **'DEPARTURE\_TIME'**, **'FLIGHT\_NUMBER'**,

**'DESTINATION\_AIRPORT'**, **'ORIGIN\_AIRPORT'**, **'DAY\_OF\_WEEK'**,**'TAXI\_OUT'**] dataset.fillna dataset.dropna() dataset = dataset.fillna(0) dataset.fillna(method=**'ffill'**) print(dataset.isnull().values.any()) print(dataset.dtypes) dataset.to\_csv(**'file1.csv'**) data\_dict = dataset.to\_dict() **return** data\_dict

**def** MyLogiSticregression(self,dataset):

print(**"###Logistic Regression####"**) dataset = dataset[[**'DAY'**,**'DEPARTURE\_TIME'**,**'FLIGHT\_NUMBER'**,**'DESTINATION\_AIRPORT'**,**' ORIGIN\_AIRPORT'**,**'DAY\_OF\_WEEK'**,**'TAXI\_OUT'**]] X = dataset.iloc[:,:1].values y = dataset.iloc[:,2].values

X\_train,X\_test,y\_train,y\_test = train\_test\_split(X,y, test\_size=1/3,random\_state=0) model = LogisticRegression() model.fit(X\_train,y\_train) y\_pred = model.predict(X\_test)

lgDict = {}

lg\_MAE = metrics.mean\_absolute\_error(y\_pred.round(), y\_test) lg\_MSE = metrics.mean\_squared\_error(y\_pred.round(), y\_test) lg\_EVS = metrics.explained\_variance\_score(y\_test, y\_pred, sample\_weight=**None**, multioutput=**'uniform\_average'**) lg\_MedianAE = metrics.median\_absolute\_error(y\_test, y\_pred) lg\_R2Score = metrics.r2\_score(y\_test, y\_pred, sample\_weight=**None**, multioutput=**'uniform\_average'**)

lgDict.update({**'lg\_MAE'**:lg\_MAE,**'lg\_MSE'**:lg\_MSE,**'lg\_EVS'**:lg\_EVS,**'lg\_Medi anAE'**:lg\_MedianAE,**'lg\_R2Score'**:lg\_R2Score})

### Calculation ArrivalDelay.py

**import** pandas **as** pd

**import** matplotlib.pyplot **as** plt **import** matplotlib matplotlib.use(**'TkAgg'**) **from** sklearn.model\_selection **import** train\_test\_split **from** sklearn.linear\_model **import** LogisticRegression **from** sklearn.tree **import** DecisionTreeRegressor **from** sklearn.metrics **import** accuracy\_score **from** sklearn.ensemble **import** RandomForestRegressor **from** sklearn.linear\_model **import** BayesianRidge **from** sklearn.ensemble **import** GradientBoostingRegressor **from** sklearn **import** metrics **import** numpy **as** np **class** ArrivalDelay:

**def** process\_data(self,datasetname):

dataset = datasetname

dataset = dataset[[**'DAY'**, **'DEPARTURE\_TIME'**, **'FLIGHT\_NUMBER'**,

**'DESTINATION\_AIRPORT'**, **'ORIGIN\_AIRPORT'**, **'DAY\_OF\_WEEK'**,**'TAXI\_OUT'**]] dataset.fillna dataset.dropna() **return** data\_dict

**def** MyLogiSticregression(self,dataset): print(**"###Logistic Regression####"**) dataset = pd.read\_csv(dataset) dataset.dropna() dataset = dataset.fillna(0) X = dataset.iloc[:,:3].values y = dataset.iloc[:,2].values

# SYSTEM ANALYSIS

**SYSTEM ANALYSIS**

**EXISTING SYSTEM:**

Supervised automatic learning algorithms Support Vector Machine and the k- nearest neighbor to predict delays in the arrival of operated flights including the five busiest US airports. The precision achieved was very low with gradient booster as a classifier with a limited data set. Applied machine learning algorithms k-Nearest Neighbors to predict delays on individual flights. Flight schedule data and weather forecasts have been incorporated into the model. Sampling techniques were used to balance the data and it was observed that the accuracy of the classifier trained without sampling was more that of the trained classifier with sampling techniques.

**DISADVANTAGES OF EXISTING SYSTEM:**

* Non-parametric nature do not assume a particular functional form of the response under investigation data.
* The predictability may additionally range because of factors such as the number of origin destination pairs and the forecast horizon.
* The forecasts were based on some key attributes.
* **Algorithm**: Multiple Linear Regression, Support Vector Machine, k-nearest neighbor.

**PROPOSED SYSTEM:**

To predict flight delays to train models, we have collected data accumulated by the Bureau of Transportation, U.S. Statistics of all the domestic flights taken in 2015 was used. The US Bureau of Transport Statistics provides statistics of arrival and departure that includes actual departure time, scheduled departure time, and scheduled elapsed time, wheels-off time, departure delay and taxi-out time per airport. Cancellation and Rerouting by the airport and the airline with the date and time and flight labelling along with airline airborne time are also provided. The data set consists of 31 columns and 20277 and it can grow able by our implementation. By using pandas library we can fill the missing values which is essential for processing data for model.

**ADVANTAGES OF PROPOSED SYSTEM:**

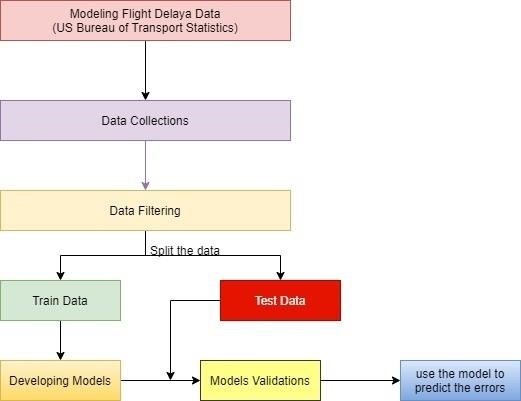
* Supervised learning technique to gather the advantages of having the schedule and real arrival time.
* Algorithms are light computation cost will taken.
* We develop a system that predicts for a delay in flight departure based on certain parameters.

**Algorithm**: Logistic Regression, Decision Tree Regressor, Bayesian Ridge, Random Forest Regressor, Gradient Boosting Regressor

# SYSTEM DESIGN

## SYSTEM DESIGN

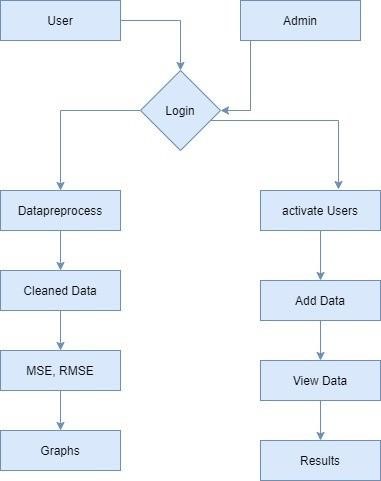
**SYSTEM ARCHITECTURE:**



9.1 : System Architecture

**DATA FLOW DIAGRAM:**

1. The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out on this data, and the output data is generated by this system. 2. The data flow diagram (DFD) is one of the most important modeling tools. It is used to model the system components. These components are the system process, the data used by the process, an external entity that interacts with the system and the information flows in the system.



9.2 : Data Flow Diagram **UML DIAGRAMS**

UML stands for Unified Modeling Language. UML is a standardized general- purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

**GOALS:**

The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.

**USE CASE DIAGRAM:**

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed forwhich actor. Roles of the actors in the system can be depicted.

**Admin**

**user**



**Register**

**Activate**

**Data**

**Preprocess**

**Departure**

**Results**

**Arrival Results**

**Graphs**

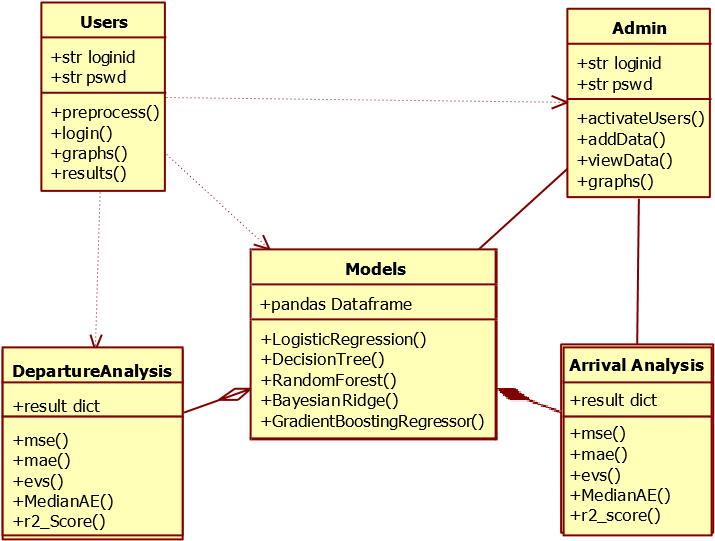
**Add Data**



* 1. : Use Case Diagram

**CLASS DIAGRAM:**

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.



* 1. : Class Diagram

# SYSTEM STUDY

**SYSTEM STUDY**

## FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

### Three key considerations involved in the feasibility analysis are,

 **ECONOMICAL FEASIBILITY**  **TECHNICAL FEASIBILITY**  **SOCIAL FEASIBILITY**

## ECONOMICAL FEASIBILITY

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

## TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

## SOCIAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

# SYSTEM TEST

## SYSTEM TEST

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

**TYPES OF TESTS**

**Unit testing**

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration.

**Integration testing**

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

**White Box Testing**

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

**Black Box Testing**

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document.

**Integration Testing**

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

**Test Results:** All the test cases mentioned above passed successfully. No defects encountered.

# TEST CASES

**Sample Test Cases**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S.no** | **Test Case** | **Excepted Result** | **Result** | **Remarks(IF**  **Fails)** |
| 1. | User Register | If User registration successfully. | Pass | If already user email exist then it fails. |
| 2. | User Login | If User name and password is  correct then it  will getting valid page. | Pass | Un Register Users will not logged in. |
| 3. | Admin Add the Data | A new record will added to our dataset. | Pass | Data types not match then this case failed |
| 4. | Data Cleaning | Data will be cleaned. | Pass | The data will be in in or float format, otherwise  algorithm will not work.. |
| 5. | Mean Square  error calculates | Means Square error calculation | Pass | If noisy data there then the result will not accurate |
| 6. | Mean Absolute  Error calculation | Selected Model  Mean Absolute  Error calculated and sent to browser. | Pass | If noisy data there then the result will not accurate. |
| 7. | Explained  Variance  Score calculated | Selected Model  Explained  Variance  Score calculated calculated and sent to browser | Pass | If noisy data there then the result will not accurate. |
| 8. | Median  Absolute  Error calculated | Median Absolute  Error Calculated and sent to controller | Pass | If noisy data there then the result will not accurate |
| 9. | R2\_Score Calculated | R2\_Score  Calculated and send to browser | Pass | If noisy data there then the result will not accurate |
| 10. | Admin can activate the register users | Admin can  activate the register user id | Pass | If user id not found then it won’t login. |

# SCREEN SHOTS

**Home**

**page**

**Register**

**Form**

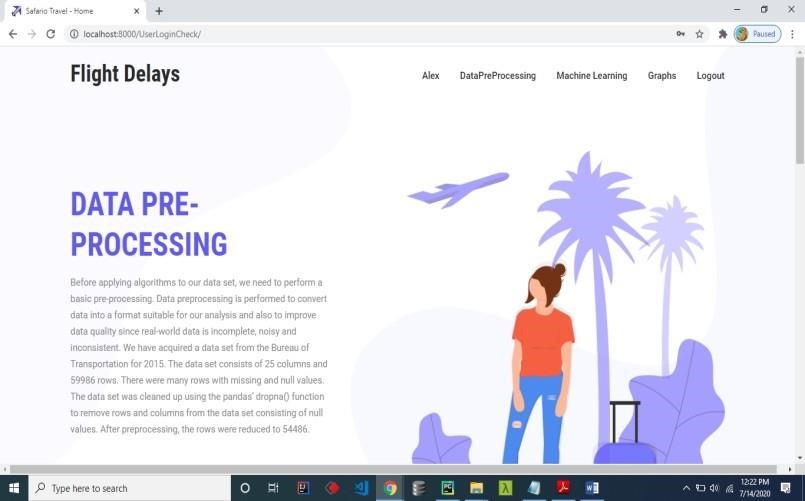
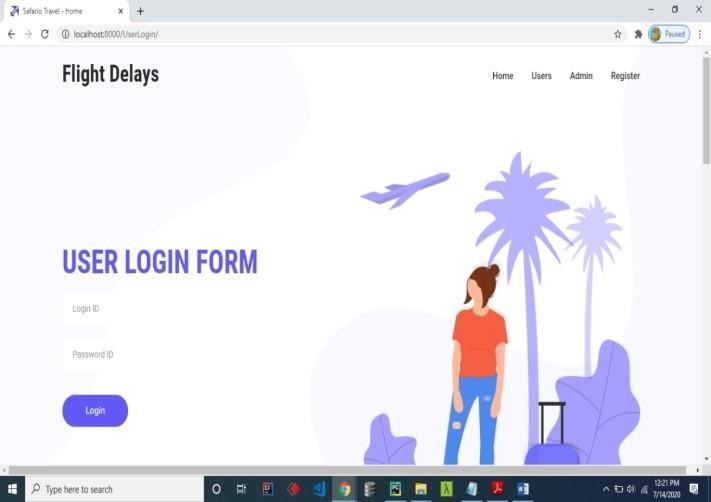
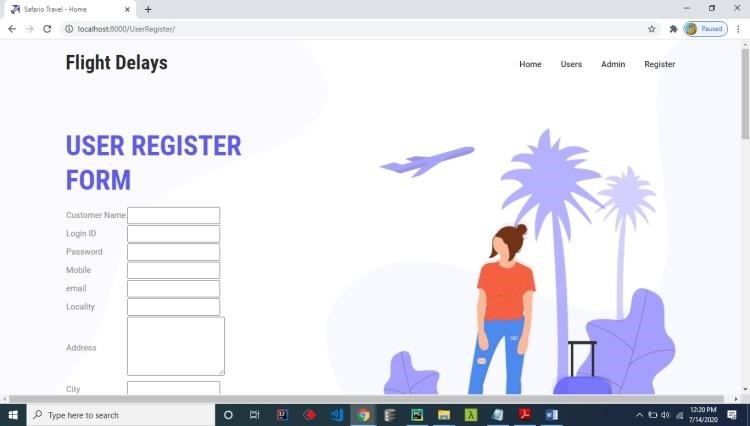
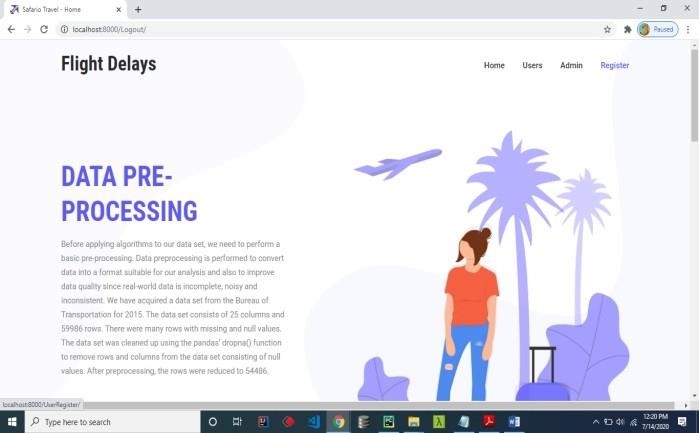
**User**

**Login**

**Form**

**User Home**

**Page**



**Preprocessed**

**Data**

**Algorithm**

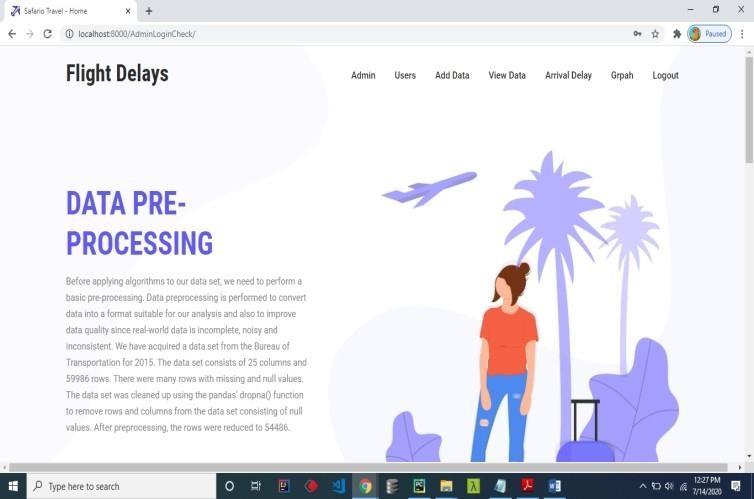
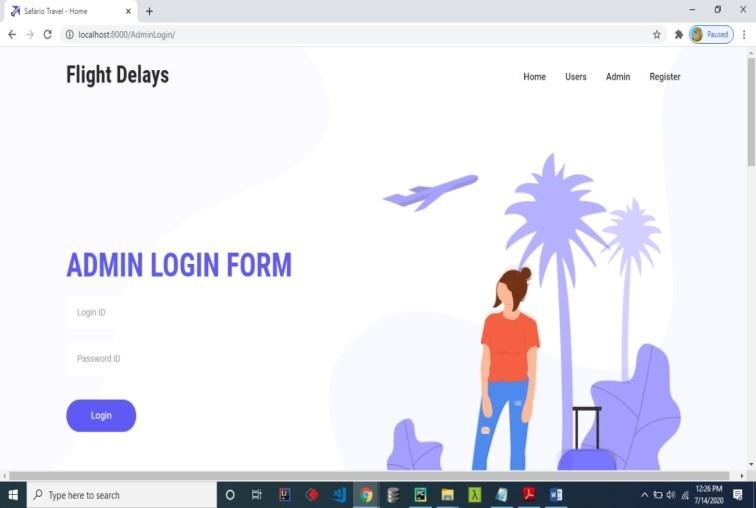
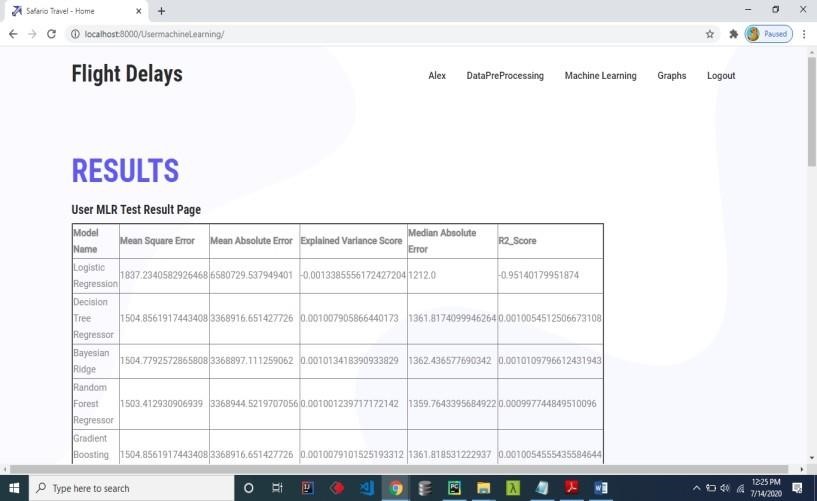
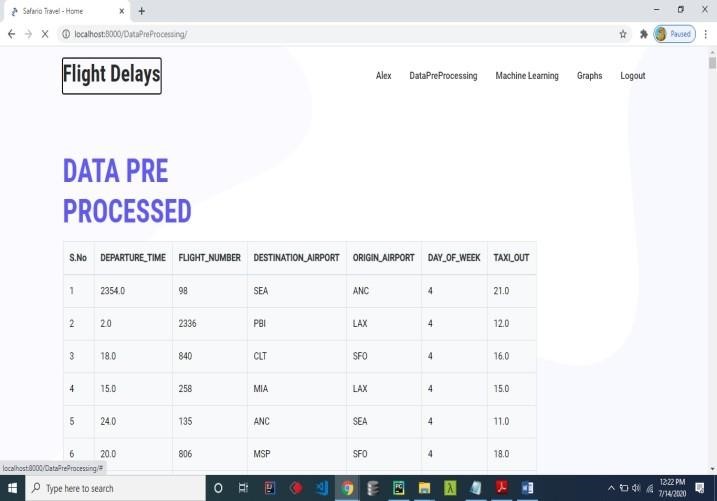
**codes**

**Admin Login**

**page**

**Admin home**

**Page**



**Admin**

**Adding Data**

**Admin View**

**Data**

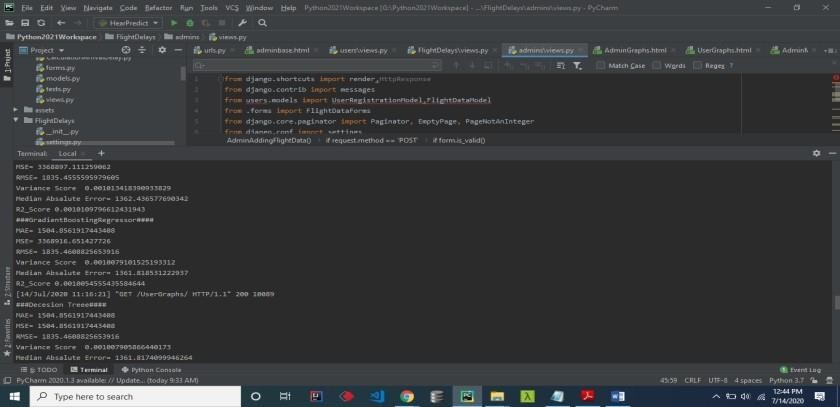
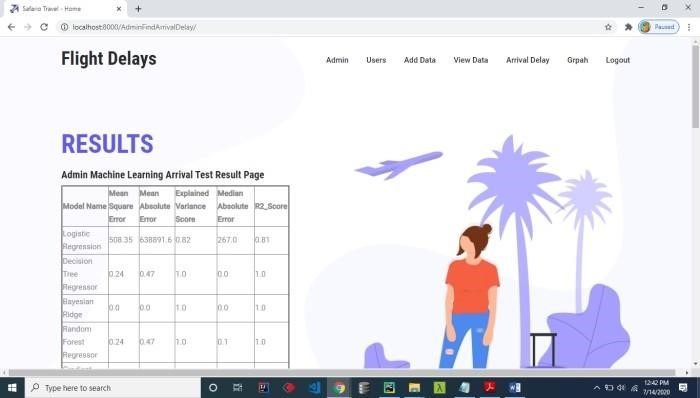
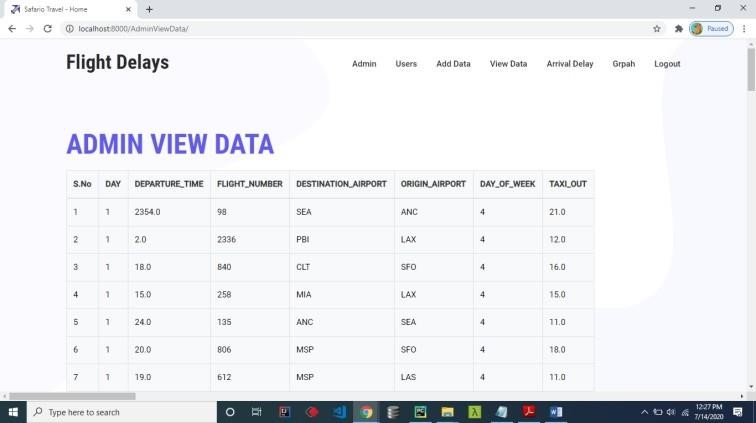
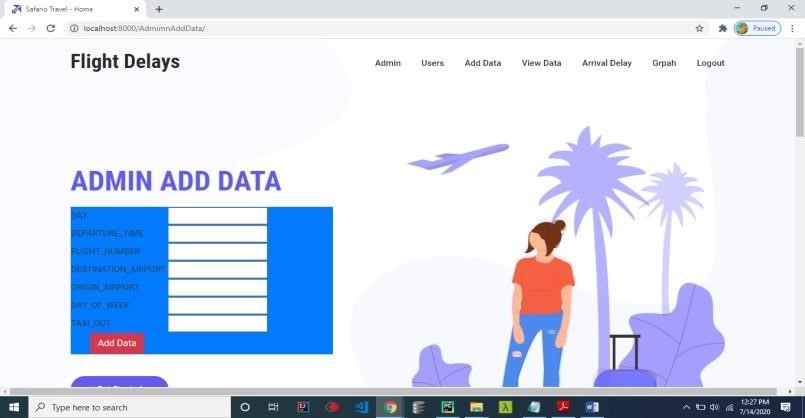
**Admin**

**View**

**Results**

**Server Side**

**Results**



# CONCLUSION

## CONCLUSION

Machine learning algorithms were applied progressively and successively to predict flight arrival & delay. We built five models out of this. We saw for each evaluation metric considered the values of the models and compared them. We found out that: - In Departure Delay, Random Forest Regressor was observed as the best model with Mean Squared Error 2261.8 and Mean Absolute Error 24.1, which are the minimum value found in these respective metrics. In Arrival Delay, Random Forest Regressor was the best model observed with Mean Squared Error 3019.3 and Mean Absolute Error 30.8, which are the minimum value found in these respective metrics. In the rest of the metrics, the value of the error of Random Forest Regressor although is not minimum but still gives a low value comparatively. In maximum metrics, we found out that Random Forest Regressor gives us the best value and thus should be the model selected.

# REFERENCES

## REFERENCES

1. N. G. Rupp, "Further Investigation into the Causes of Flight Delays," in Department of Economics, East Carolina University, 2007.

1. "Bureau of Transportation Statistics (BTS) Databases and Statistics," [Online]. Available: [http://www.transtats.bts.gov.](http://www.transtats.bts.gov/)

1. "Airports Council International, World Airport Traffic Report," 2015,2016.

1. E. Cinar, F. Aybek, A. Caycar, C. Cetek, "Capacity and delay analysis for airport manoeuvring areas using simulation," Aircraft Engineering and Aerospace Technology, vol. 86, no. No. 1,pp. 43-55, 2013.

1. Navoneel, et al., Chakrabarty, "Flight Arrival Delay Prediction Using Gradient Boosting Classifier," in Emerging Technologies in Data Mining and Information Security, Singapore, 2019.

1. Y. J. Kim, S. Briceno, D. Mavris, Sun Choi, "Prediction of weatherinduced airline delays based on machine learning algorithms," in 35th Digital Avionics Systems Conference (DASC), 2016.

1. W.-d. Cao. a. X.-y. Lin, "Flight turnaround time analysis and delay prediction based on Bayesian Network," Computer Engineering and Design, vol. 5, pp. 1770-1772, 2011.

1. J. J. Robollo, Hamsa, Balakrishnan, "Characterization and Prediction of Air Traffic Delays".

1. S. Sharma, H. Sangoi, R. Raut, V. C. Kotak, S. Oza, "Flight Delay Prediction System Using Weighted Multiple Linear Regression," International Journal of Engineering and Computer Science, vol. 4, no. 4, pp. 11668 - 11677, April 2015.