Sprint-2 objectives:

- Build a dashboard
- Build python code (flask)

Dashboard of flight delay insights:

Dashboard.html:

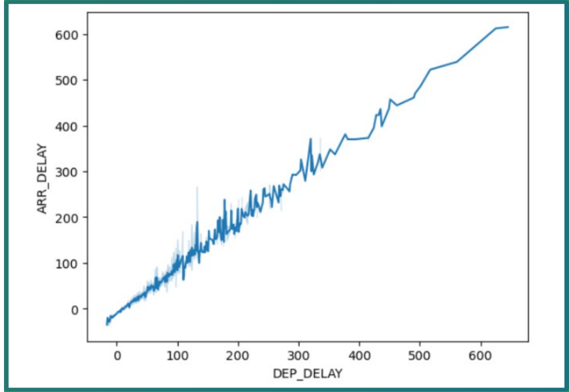
```
<html>
<head>
<title>Dashboard analysis</title>
<link rel="stylesheet" href="{{ url_for('static', filename='des.css') }}">
</head>
<body>
<div class="topnav">
<a href="{{ url_for('formm')}}">Predict</a>
<a href="{{ url_for('home')}}">Logout</a>
</div>

<h1 style="text-align:center;">Flight Delay Dashboard</h1>

<div class="row">
  <div class="column">
    <img class="img1" src="{{url_for('static', filename='relation.jpg')}}"/>
    <figcaption>This graph shows the delay in arrival vs the delay in departure.
    Therefore it can be inferred that flights that have delayed departure will have a delayed
arrival.
    </figcaption>
  </div>
  <div class="column">
    <img class="img2" src="{{url_for('static', filename='corr.jpg')}}"/>
    <figcaption>This heatmap depicts the correlation between the different columns of data
     in the dataset. As we can see there is a high correlation between arr_delay and dep_delay
     as we have previously inferred.
    </figcaption>
  </div>
</div>
<br><br>
<div class="row">
  <div class="column">
    <img class="img1" src="{{url_for('static', filename='dom.jpg')}}"/>
    <figcaption>This graph shows the delay in different days across the month. There is no
explicit pattern that can be concluded from the data. Therefore, flight delay may occur at
anytime of the month.</figcaption>
  </div>
  <div class="column">
    <img class="img2" src="{{url_for('static', filename='reason.jpg')}}"/>
    <figcaption>The bar graph shows the different reasons due to which flight delay occurs. The
main reason behind flight delays are weather conditions. Since weather could be unpredictable
at times, we need a reliable source of delay prediction.</figcaption>
  </div>
</div>
<br><br>
<div class="row">
  <div class="column">
    <img class="img1" src="{{url_for('static', filename='dow.jpg')}}"/>
    <figcaption>This figure shows the delay during different stages of the week. Again, there is no
    specific pattern that can be derived that can be used to predict delays. Therefore flight delay
can occur at anytime/anyday. This calls for the need for a reliable weather prediction
model.</figcaption>
  </div>
  <div class="column">
    <img class="img2"  src="{{url_for('static', filename='rf.jpg')}}"/>
    <figcaption>This shows the confusion matrix of the model we are going to use to predict flight
    delays: Random Forest Model. The best model possible from the given dataset was found to
be Random Forest with an accuracy of about 84%.</figcaption>
  </div>
</div>
</form>
</body>
</html>
```

Dashboard analysis

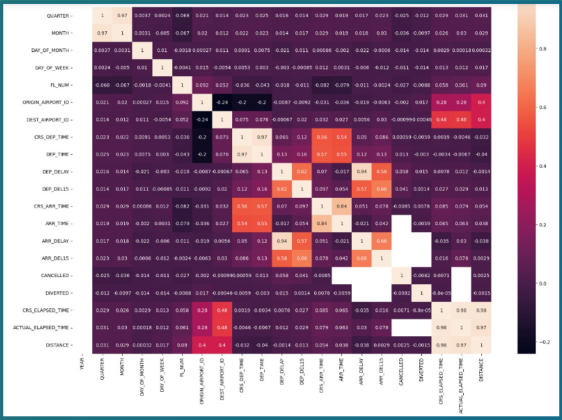
127.0.0.1:5000/dash

Predict Logout

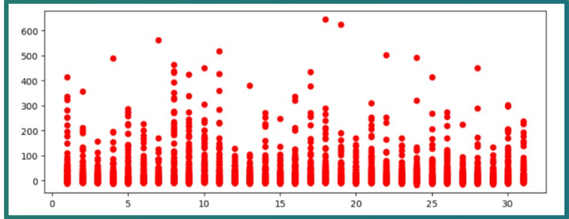
Flight Delay Dashboard



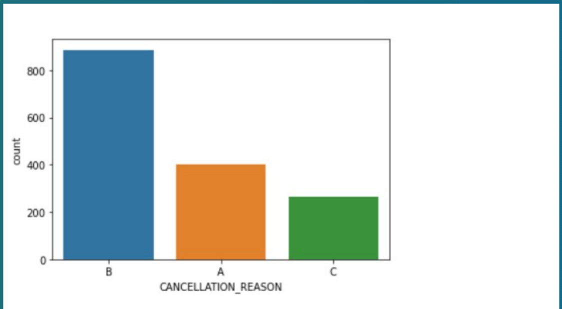
This graph shows the delay in arrival vs the delay in departure. Therefore it can be inferred that flights that have delayed departure will have a delayed arrival.



This heatmap depicts the correlation between the different columns of data in the dataset. As we can see there is a high correlation between arr_delay and dep_delay as we have previously inferred.

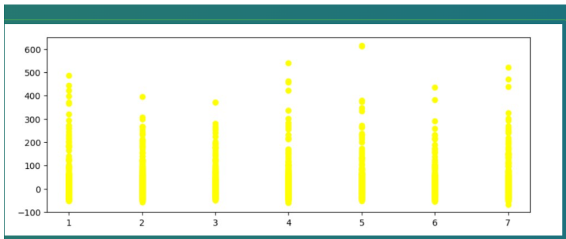


This graph shows the delay in different days across the month. There is no explicit pattern that can be concluded from the data. Therefore, flight delay may occur at anytime of the month.



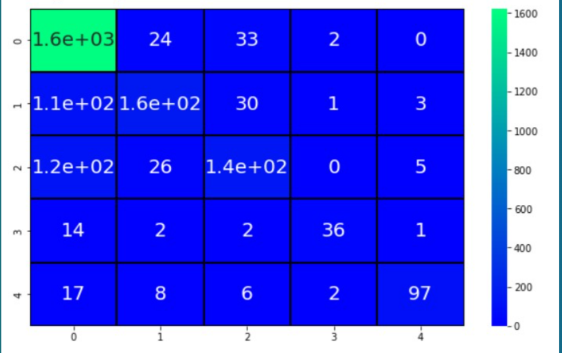
Reason for Cancellation of flight: A - Airline/Carrier; B - Weather; C - National Air System;

The bar graph shows the different reasons due to which flight delay occurs. The main reason behind flight delays are weather conditions. Since weather could be unpredictable at times, we need a reliable source of delay prediction.



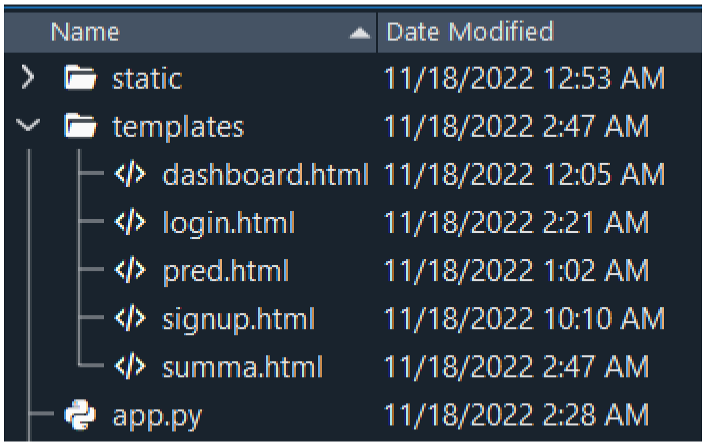
This figure shows the delay during different stages of the week. Again, there is no specific pattern that can be derived that can be used to predict delays. Therefore flight delay can occur at anytime/anyday. This calls for the need for a reliable weather prediction model.

```
Testing Sensitivity for Random Forest 0.9360230547550432
Testing Specificity for Random Forest 0.8716577540106952
Testing Precision for Random Forest 0.9854368932038835
Testing accuracy for Random Forest 0.8368506493506493
```



This shows the confusion matrix of the model we are going to use to predict flight delays: Random Forest Model. The best model possible from the given dataset was found to be Random Forest with an accuracy of about 84%.

Python code (in flask):



```
Name                Date Modified
> static            11/18/2022 12:53 AM
v templates         11/18/2022 2:47 AM
   dashboard.html   11/18/2022 12:05 AM
   login.html       11/18/2022 2:21 AM
   pred.html        11/18/2022 1:02 AM
   signup.html      11/18/2022 10:10 AM
   summa.html       11/18/2022 2:47 AM
  app.py            11/18/2022 2:28 AM
```

app.py:

```
import os
from pymongo import MongoClient
from flask import Flask, request, render_template
import requests
client = MongoClient('localhost', 27017)
db = client.login

login = db.users
# NOTE: you must manually set API_KEY below using information retrieved from your IBM
Cloud account.
API_KEY = "FDu8w9acEuLpZiojHIEoW5Rc2uHT9889GjnPT5QZ0-LN"
token_response = requests.post('https://iam.cloud.ibm.com/identity/token', data={"apikey":
 API_KEY, "grant_type": 'urn:ibm:params:oauth:grant-type:apikey'})
mltoken = token_response.json()["access_token"]

header = {'Content-Type': 'application/json', 'Authorization': 'Bearer ' + mltoken}
app=Flask(__name__)
@app.route('/', methods=('GET','POST'))
def home():
    if request.method=='POST':
        email=request.form['em']
        uname=request.form['unme']
        password=request.form['pswd']
        login.insert_one({"email":email,"username":uname,"password":password})
        return render_template('login.html')
    return render_template('login.html')

@app.route('/sign')
def sign():
    return render_template('signup.html')
```

```
@app.route('/dash', methods=('GET','POST'))
def dashh():
    if request.method=='POST':
        coll=login.find()
        uname=request.form['uname']
        password=request.form['psw']
        for i in coll:
          if(uname==i['username'] and password==i['password']):
              return render_template('dashboard.html')
    if request.method=='GET':
      return render_template('dashboard.html')

    return render_template('login.html')

@app.route('/form')
def formm():
    return render_template('summa.html')

@app.route('/predict',methods=['POST'])
def predict():
    '''
    For rendering results on HTML GUI
    '''

    b=int(request.form["month"])
    c=request.form["daym"]
    d=request.form["dayw"]
    e=request.form["fnum"]
    f=int(request.form["airport"])
    g=int(request.form["airportd"])
    h=request.form["dtime"]
    i=request.form["atime"]
    j=request.form["ttime"]
    if b==1 or b==2 or b==3:
        a=1
        l=2
    elif b==4 or b==5:
        a=2
        l=3
    elif b==6:
        a=2
        l=0
    elif b==7 or b==8:
        a=3
        l=0
    elif b==9:
        a=3
        l=1
    elif b==10 or b==11:
        l=1
        a=4
    elif b==12:
        a=4
        l=2
    ff=f
    gg=g
    if ff==gg:
        return render_template('summa.html', prediction_text='No delay(same airport!)')
    if ff<gg:
        ff,gg=gg,ff
    if gg==1 and ff==2:
        k=594
    elif gg==1 and ff==3:
        k=760

    elif gg==1 and ff==4:
        k=907

    elif gg==1 and ff==5:
        k=2182

    elif gg==2 and ff==3:
        k=509

    elif gg==2 and ff==4:
        k=528

    elif gg==2 and ff==5:
        k=1927

    elif gg==3 and ff==4:
        k=1029

    elif gg==3 and ff==5:
        k=2422

    elif gg==4 and ff==5:
        k=1399
        #print (a,b,c,d,e,f,g,h,i,j,k,l)

        payload_scoring = {"input_data": [{"field":
    [["QUARTER","MONTH","DAY_OF_MONTH","DAY_OF_WEEK","FL_NUM","ORIGIN","DEST","
    CRS_DEP_TIME","CRS_ARR_TIME","CRS_ELAPSED_TIME","DISTANCE","SEASON"]],
    "values": [[a,b,c,d,e,f,g,h,i,j,k,l]]}]}

        response_scoring =
    requests.post('https://us-south.ml.cloud.ibm.com/ml/v4/deployments/7d6c3b49-ec70-4cfe-ab88-
    4f6dbe6a7997/predictions?version=2022-11-16', json=payload_scoring,
         headers={'Authorization': 'Bearer ' + mltoken})
        #print("Scoring response")
        predictions=response_scoring.json()
        m=predictions['predictions'][0]['values'][0][0]



        if m==0:
            return render_template('pred.html', prediction_text='No delay is predicted to happen. HAVE
    A NICE FLIGHT!!')
        elif m==1:
            return render_template('pred.html', prediction_text='Delay in flight departure is predicted to
    happen')
        elif m==2:
            return render_template('pred.html', prediction_text='Delay in both flight departure and
    arrival is predicted to happen')
        elif m==3:
            return render_template('pred.html', prediction_text='Flight is predicted to get Diverted')
        elif m==4:
            return render_template('pred.html', prediction_text='Flight is predicted to get Cancelled!')
        else:
            return render_template('pred.html', prediction_text='output {}'.format(m))


    if __name__ == "__main__":
        os.environ.setdefault('FLASK_ENV', 'development')
        app.run(debug=False)

    #THIS PAGE IS USED FOR NAVIGATION BETWEEN PAGES AND DISPLAYING THE
    CORRECT PAGES DONE USING FLASK.
```