

## SPRINT – 2 PROJECT DOCUMENT

|              |  |
|--------------|--|
| Date         | 5 November 2022                                |
| Team ID      | PNT2022TMID07489                               |
| Project Name | Flight Delay Prediction Using Machine Learning |

### DEVELOPMENT PHASE:

#### SPRINT-2:

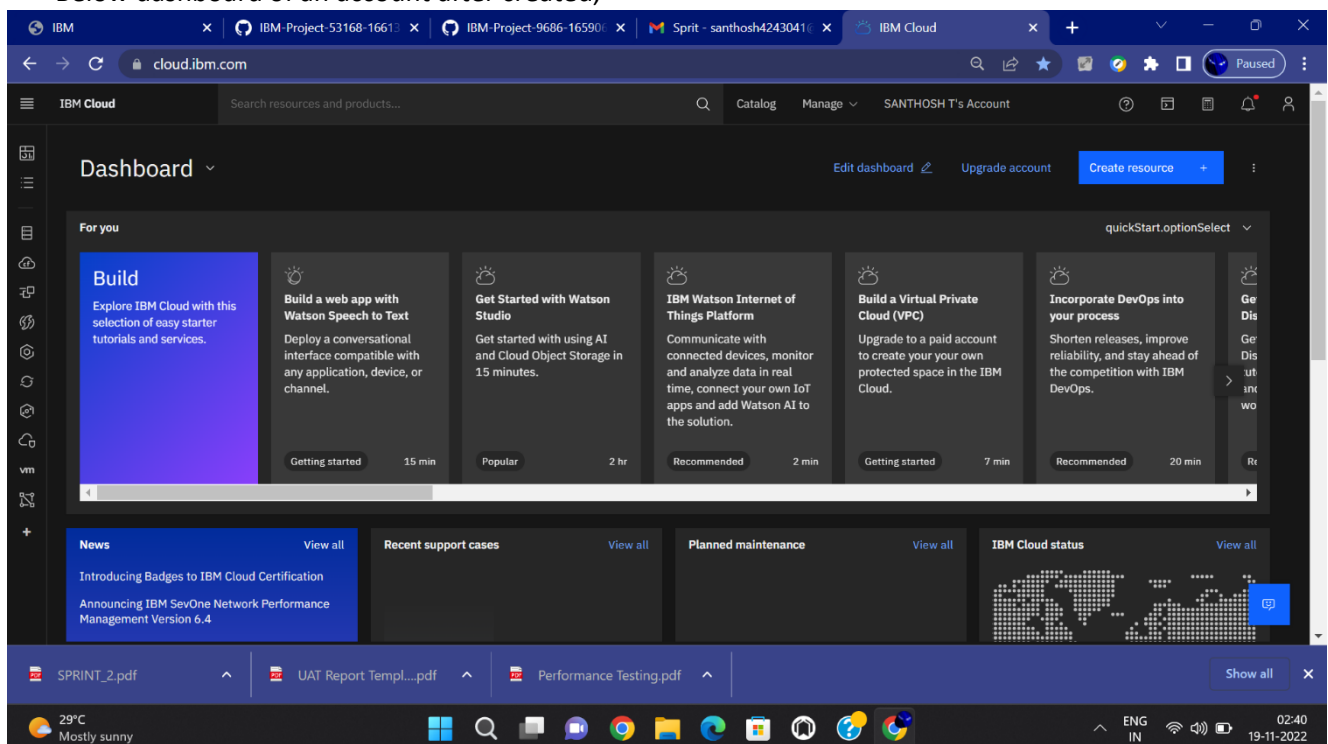
- Creating IBM cloud account & Required Resources
- Deploy our model in IBM Watson
- Creating Dashboard using HTML/CSS
- Create web app and Hosting in flask
- Testing web app

### Creating IBM cloud account & Required Resources:

#### Creating IBM cloud account:

First, need to create IBM Cloud account by using SI Mail Id and SI Password which is provided by IBM in profile.

Below dashboard of an account after created,



## Creating IBM Cloud Required Resources:

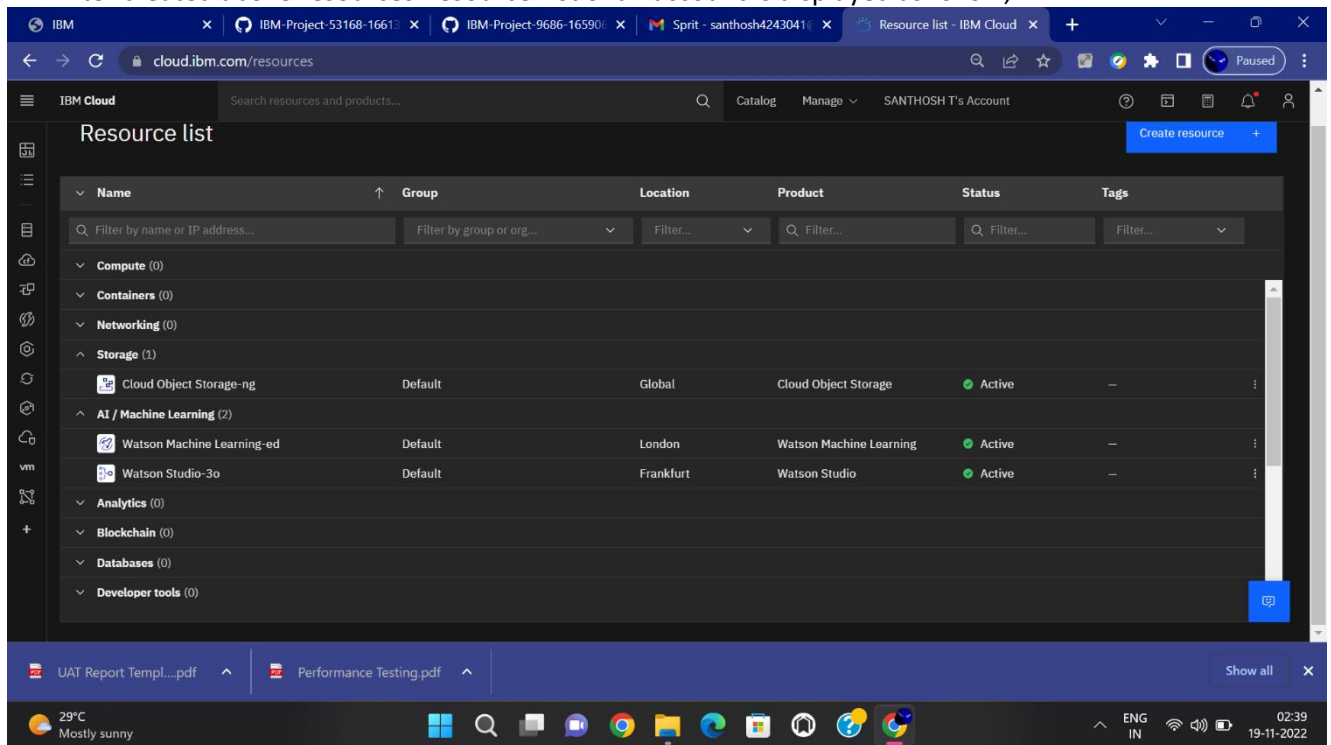
After creating IBM cloud account, to deploy ML model, need to create following resources such as,

Cloud Object Storage

Watson Machine Learning

Watson Studio

After created above resources Resource List of an account is displayed as follow,



All the resource are in active state.

All the required cloud resources are created successfully.

## Deploy our model in IBM Watson:

To deploy ML model in IBM cloud, need to create project in IBM Watson. After successful creation of project import .ipynb file of sprint-1 which ML models are build in Jupyter notebook.

Upload required datasets and import it.

Deploy model using following code,

```
!pip install -U ibm-watson-machine-learning
from ibm_watson_machine_learning import APIClient
import json
import numpy as np
wml_cred={
    "apikey":"okbr7ARnOQjyplTOyvNFC2QVkfCF6q7afpci065Hucby8",
    "url":"https://us-south.ml.cloud.ibm.com"
}
wml_clients=APIClient(wml_cred)
wml_clients.spaces.list()
space_id="6d7c1218-3aca-4256-be3d-d610732530b1"
```

```
wml_clients.set.default_space(space_id)
wml_clients.software_specifications.list(500)
MODEL_NAME="randomforest"
DEPLOYMENT_NAME="rf_deployment"
DEMO_MODEL=rf
soft_sepc_id=wml_clients.software_specifications.get_id_by_name("runtime-22.1-py3.9")
```

In [115]:

```
model_props={
    wml_clients.repository.ModelMetaNames.NAME:MODEL_NAME,
    wml_clients.repository.ModelMetaNames.TYPE:"scikit-learn_1.0",
    wml_clients.repository.ModelMetaNames.SOFTWARE_SPEC_UID: soft_sepc_id
}
```

In [116]:

```
model_details=wml_clients.repository.store_model(model=DEMO_MODEL,meta_props=model_props,training_data=x_train,
                                                training_target=y_train.values.ravel())
```

In [117]:

```
model_details
model_id=wml_clients.repository.get_model_id(model_details)
dep_props={
    wml_clients.deployments.ConfigurationMetaNames.NAME:DEPLOYMENT_NAME,
    wml_clients.deployments.ConfigurationMetaNames.ONLINE:{}
}
```

In [125]:

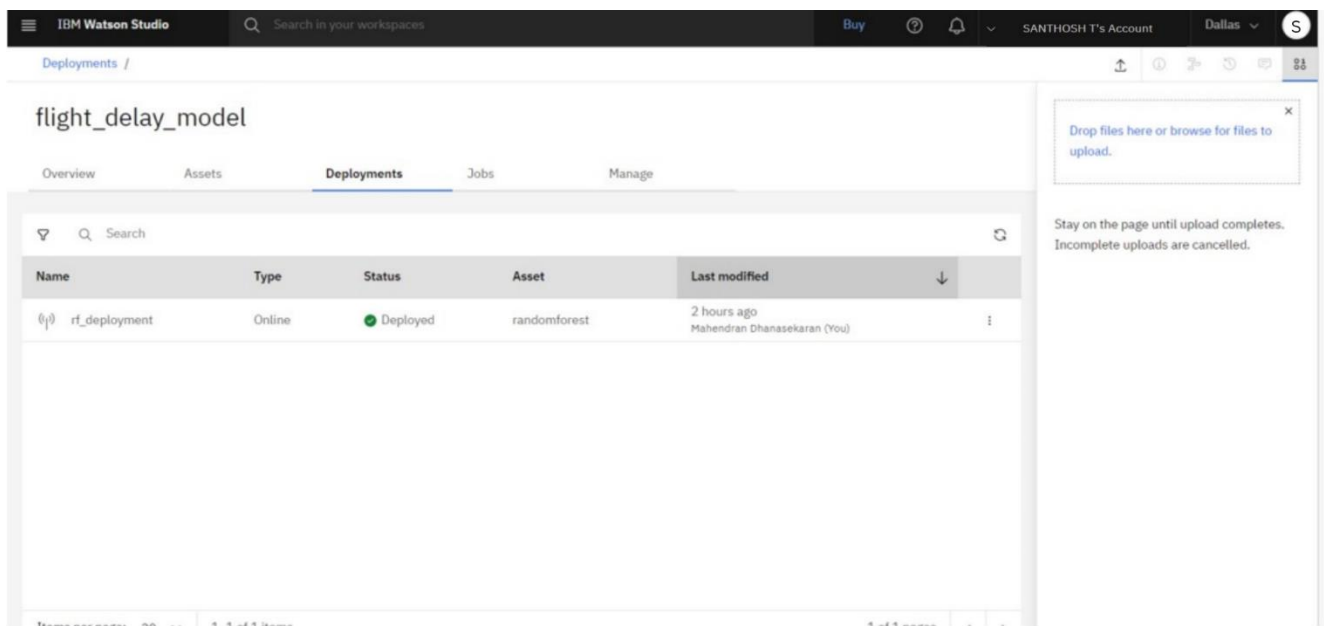
```
deployment=wml_clients.deployments.create(artifact_uid=model_id,meta_props=dep_props)
```

NOTE: APIKey must need to create to deploy and connect API

After successful of deployment, deployed is appeared in Deployment section as follow,

The screenshot shows the IBM Watson Studio interface. At the top, there's a navigation bar with 'IBM Watson Studio', a search bar, and user account information. Below the navigation bar, the breadcrumb trail is 'Deployments / flight\_delay\_model / randomforest /'. The main content area shows a deployment named 'rf\_deployment' with a green 'Deployed' status and an 'Online' button. Below this, there are tabs for 'API reference' and 'Test'. The 'Test' tab is active, showing an 'Enter input data' section. This section has two tabs: 'Text input' and 'JSON input'. The 'Text input' tab is selected, showing a message: 'Enter data manually or use a CSV file to populate the spreadsheet. Max file size is 50 MB.' Below this message are links for 'Download CSV template', 'Browse local files', and 'Search in space'. A 'Clear all' link is also present. Below these links is a table with 12 columns: 'QUARTER (int64)', 'MONTH (int64)', 'DAY\_OF\_MONTH (int64)', 'DAY\_OF\_WEEK (int64)', 'FL\_NUM (int64)', 'ORIGIN (int64)', 'DEST (int64)', 'CRS\_DEP\_TIME.1 (int64)', 'CRS\_ARR\_TIME.1', and two empty columns. The table has 6 rows. The first row contains the text 'Start typing or drag and drop a CSV file...'. The second row is empty. The third row is empty. The fourth row is empty. The fifth row is empty. The sixth row is empty. Below the table, it says '0 rows, 12 columns'. At the bottom right of the table area is a 'Predict' button.

Testing of deployed model as follow, by giving values of all the features and it gives prediction.



After these, need to copy API requesting codes on required language(python).

## Creating Dashboard using HTML/CSS:

Frontend Dashboard is created using HTML/CSS,

Result as web page like,

## Flight Delay Prediction

Quarter of the year

ex:3

Month in number

ex:12

Day of the Month

ex:28

Day of the week

ex:7

Flight Number

ex:2823

Origin Airport:

ATL

Destination Airport:

ATL

Planned Departure Time(format hhmm)

ex:1723

Planned Arrival Time(format hhmm)

ex:2023

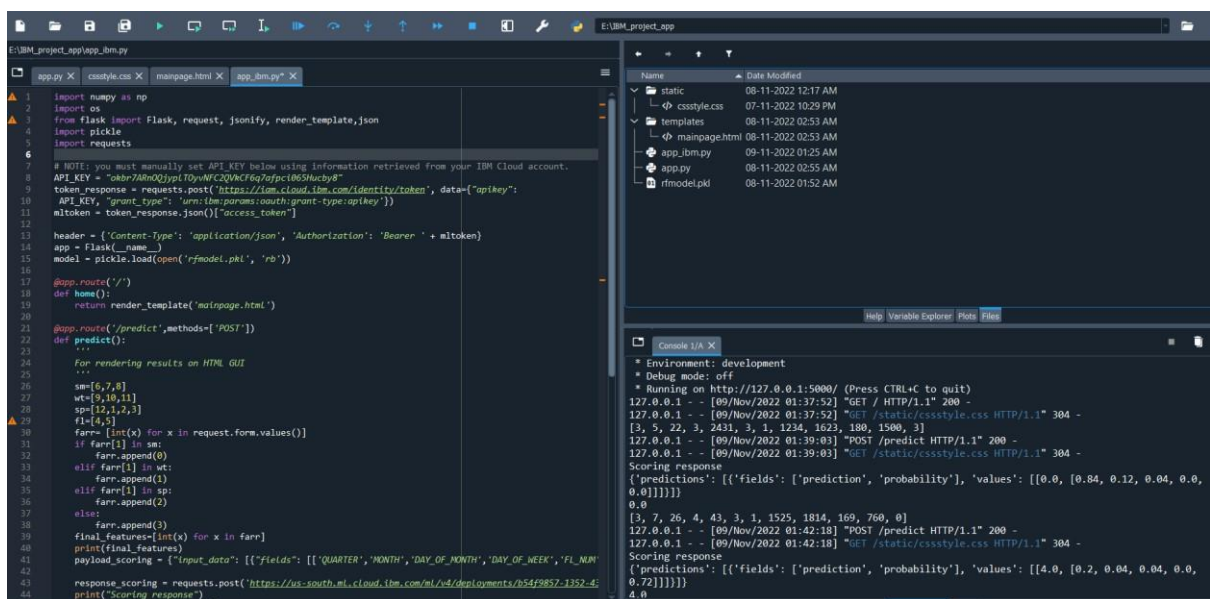
## Create web app and Hosting in flask:

First thing, need to create directory as follow,

| Name          | Date Modified       |
|---------------|---------------------|
| static        | 08-11-2022 12:17 AM |
| cssstyle.css  | 07-11-2022 10:29 PM |
| templates     | 08-11-2022 02:53 AM |
| mainpage.html | 08-11-2022 02:53 AM |
| app_ibm.py    | 09-11-2022 01:25 AM |
| app.py        | 08-11-2022 02:55 AM |
| rfmodel.pkl   | 08-11-2022 01:52 AM |

Then, code the required logic in app.py file with API connection , request and response code.

Spyder IDE looks like,



```
1 import numpy as np
2 import os
3 from flask import Flask, request, jsonify, render_template, json
4 import pickle
5 import requests
6
7 # NOTE: you must manually set API_KEY below using information retrieved from your IBM Cloud account.
8 API_KEY = "data2Pm0QjYpL7DyWfC20Kf6q7zfpcl8S9lucy8"
9 token_response = requests.post("https://iam.cloud.ibm.com/identity/token", data={"apikey":
10 API_KEY, "grant_type": "urn:ibm:params:oauth:grant-type:apikey"})
11 mltoken = token_response.json()["access_token"]
12
13 header = {'Content-Type': 'application/json', 'Authorization': 'Bearer ' + mltoken}
14 app = Flask(__name__)
15 model = pickle.load(open('rfmodel.pkl', 'rb'))
16
17 @app.route('/')
18 def home():
19     return render_template('mainpage.html')
20
21 @app.route('/predict', methods=['POST'])
22 def predict():
23     """
24     For rendering results on HTML GUI
25     """
26     sm=[5,7,0]
27     wt=[9,10,11]
28     sp=[12,1,2,3]
29     fl=[4,5]
30     farr=[int(x) for x in request.form.values()]
31     if farr[1] in sm:
32         farr.append(0)
33     elif farr[1] in wt:
34         farr.append(1)
35     elif farr[1] in sp:
36         farr.append(2)
37     else:
38         farr.append(3)
39     final_features=[int(x) for x in farr]
40     print(final_features)
41     payload_scoring = {"input_data": [{"fields": ["QUARTER", "MONTH", "DAY_OF_MONTH", "DAY_OF_WEEK", "FL_NUM"]}]}
42     response_scoring = requests.post("https://us-south.ml.cloud.ibm.com/ml/v4/deployments/b54f9857-1352-4c
43     print('Scoring response')
```

Environment: development  
Debug mode: off  
Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)

```
127.0.0.1 - - [09/Nov/2022 01:37:52] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [09/Nov/2022 01:37:52] "GET /static/cssstyle.css HTTP/1.1" 304 -
127.0.0.1 - - [09/Nov/2022 01:39:03] "POST /predict HTTP/1.1" 200 -
127.0.0.1 - - [09/Nov/2022 01:39:03] "GET /static/cssstyle.css HTTP/1.1" 304 -
Scoring response
{'predictions': [{'fields': ['prediction', 'probability'], 'values': [[0.0, [0.84, 0.12, 0.04, 0.0, 0.0]]}]]}
0.0
[3, 7, 26, 4, 43, 3, 1, 1525, 1814, 169, 760, 0]
127.0.0.1 - - [09/Nov/2022 01:42:18] "POST /predict HTTP/1.1" 200 -
127.0.0.1 - - [09/Nov/2022 01:42:18] "GET /static/cssstyle.css HTTP/1.1" 304 -
Scoring response
{'predictions': [{'fields': ['prediction', 'probability'], 'values': [[4.0, [0.2, 0.04, 0.04, 0.0, 0.72]]}]]}
4.0
```

Run the app.py file.

Localhost URL is displayed in console, copy and paste in browser then search it , frond end HTML?CSSpage is displayed. Successfully created and hosted web app in flask.

If any error caused as flask in production mode, then

Set FLASK\_ENV=Development,

Then run the app

## Testing web app:

Enter the data on the required fields,

The image displays two screenshots of a web application titled "Flight Delay Prediction".

**Top Screenshot (Input Form):**

- Quarter of the year: 3
- Month in number: 7
- Day of the Month: 26
- Day of the week: 4
- Flight Number: 43
- Origin Airport: JFK
- Destination Airport: ATL
- Planned Departure Time(format hhmm): 1525
- Planned Arrival Time(format hhmm): 1814

**Bottom Screenshot (Form with Example Values and Output):**

- Flight Number: ex.2823
- Origin Airport: ATL
- Destination Airport: ATL
- Planned Departure Time(format hhmm): ex.1723
- Planned Arrival Time(format hhmm): ex.2023
- Estimated Traveling Time(in minutes): ex.180
- Distance(in Kms): ex.2500
- Button: Predict
- Output: here is a chance to cancel the flight 4.0

Output is predicted by ML model successfully.