

## SPRINT – 2 PROJECT DOCUMENT

Date	5 November 2022
Team ID	PNT2022TMID32527
Project Name	Flight Delay Prediction Using Machine Learning

### DEVELOPMENT PHASE:

#### SPRINT-2:

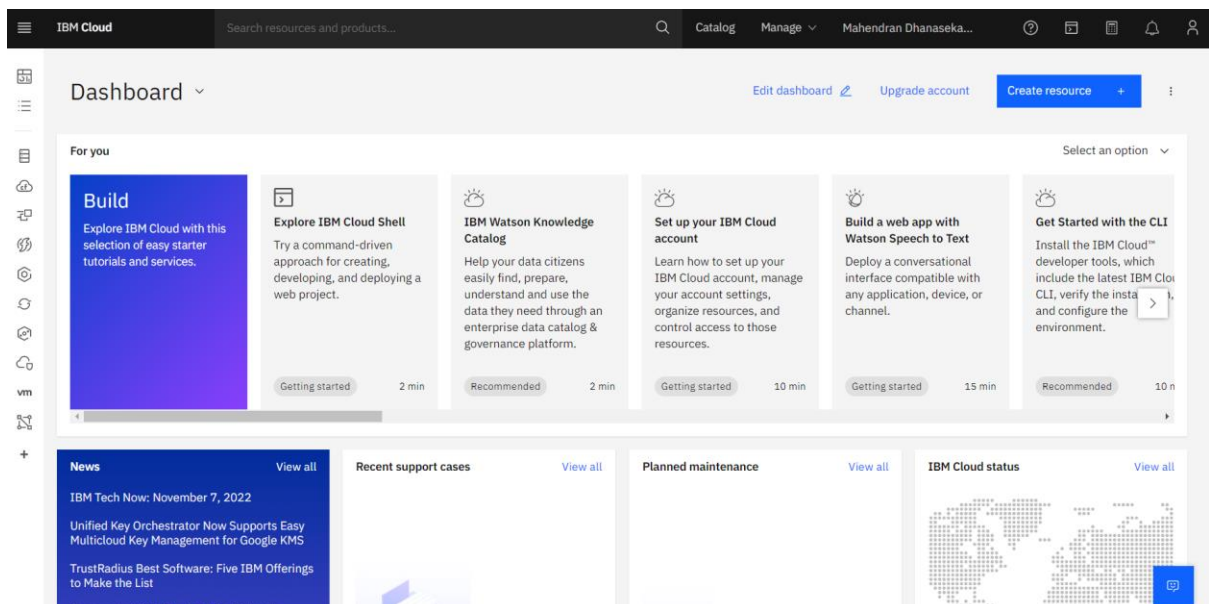
- Creating IBM cloud account & Required Resources
- Deploy our model in IBM Watson
- Creating Dashboard using HTML/CSS
- Create web app and Hosting in falk
- Testing web app

### Creating IBM cloud account & Required Resources:

#### Creating IBM cloud account:

Frist, 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:

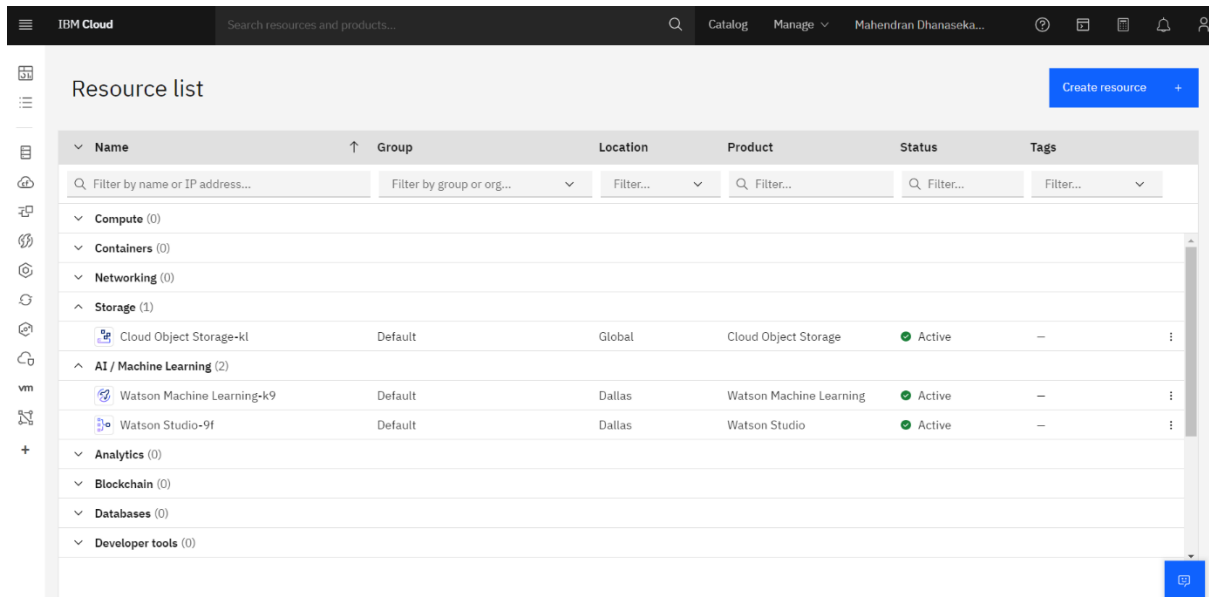
After creating IBM cloud account, to deploy ML model, need to create following resources such as,

Cloud Object Storage

Watson Machine Learning

Watson Studio

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



Name	Group	Location	Product	Status	Tags
Filter by name or IP address... Filter by group or org... Filter... Filter... Filter... Filter...					
Compute (0)					
Containers (0)					
Networking (0)					
Storage (1)					
Cloud Object Storage-kl	Default	Global	Cloud Object Storage	Active	—
AI / Machine Learning (2)					
Watson Machine Learning-k9	Default	Dallas	Watson Machine Learning	Active	—
Watson Studio-9f	Default	Dallas	Watson Studio	Active	—
Analytics (0)					
Blockchain (0)					
Databases (0)					
Developer tools (0)					

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":"okbr7ARnOQjyplTOyvNFC2QVkCF6q7afpci065Hucby8",
    "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]:

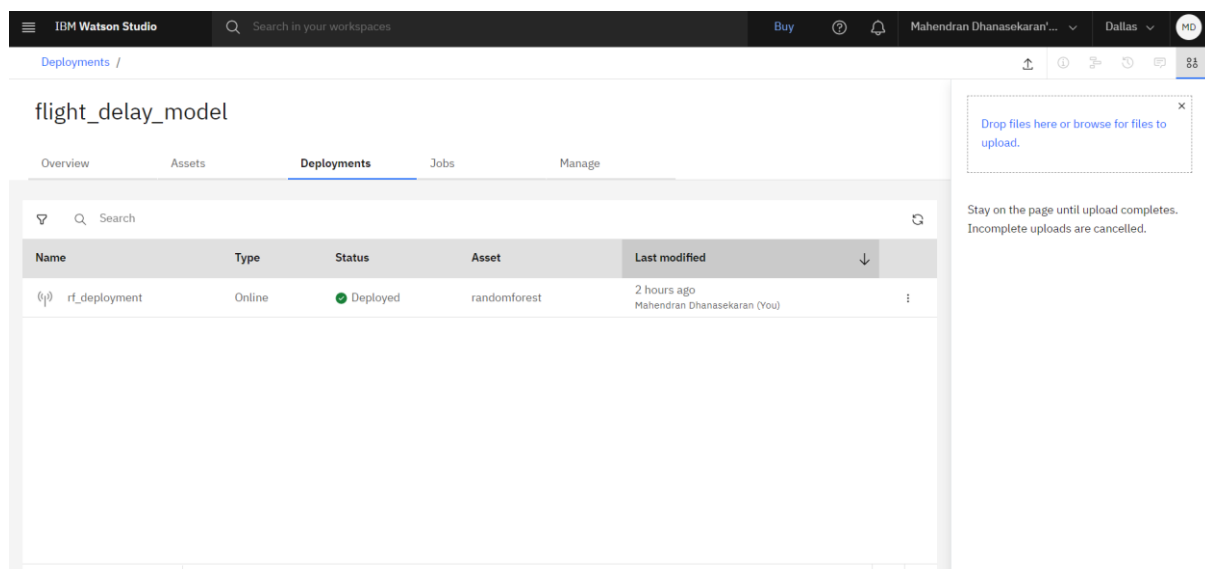
```
model_details
model_id=wml_clients.repository.get_model_id(model_details)
dep_props={
    wml_clients.deployments.ConfigurationMetaNames.NAME:DEPLOYMENT_NAME,
    wml_clients.deployments.ConfigurationMetaNames.ONLINE:{}
}
```

In [125]:

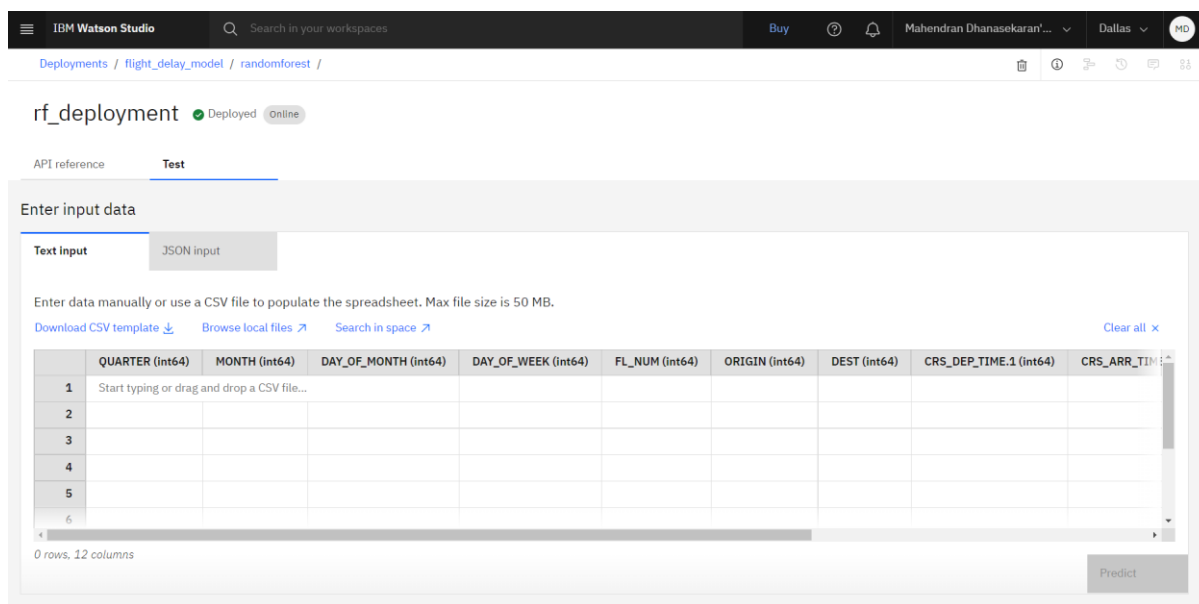
```
deployment=wml_clients.deployments.create(artifact_uid=model_id,meta_props=dep_props)
```

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

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



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

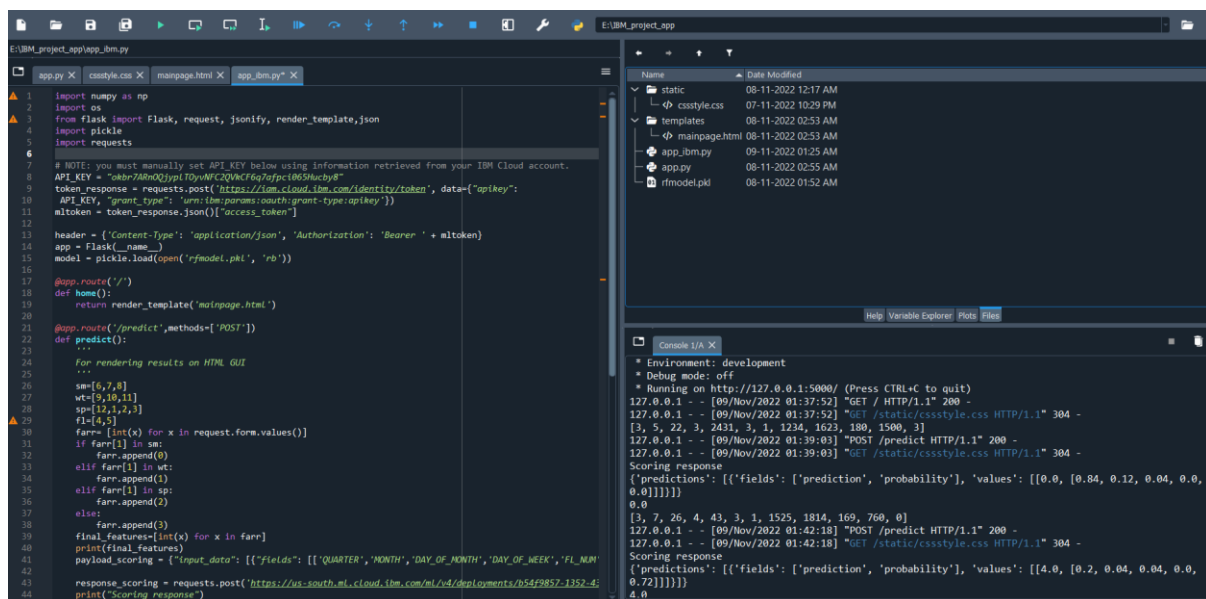
## Create web app and Hosting in falsk:

First thing, need to create directory as follow,

Name	Date Modified
static	08-11-2022 12:17 AM
└─ csstyle.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 = "okbr7ARnQjypLT0vNfC2QVhCF6g7afpc1065Hucby8"
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=[6,7,8]
27     wt=[9,10,11]
28     sp=[1,1,1,1]
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/b54f9857-1352-4:
43     print("Scoring response")
44
```

Console I/O X

```
* 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 -
[3, 5, 22, 3, 2831, 3, 1, 1234, 1623, 180, 1500, 3]
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]]]]}]
[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?CSS page 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 first screenshot shows the 'Flight Delay Prediction' web application interface. The form includes the following fields:

- 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

The second screenshot shows the same form with example data entered:

- 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

A blue 'Predict' button is visible below the input fields. Below the button, the text reads: 'here is a chance to cancel the flight 4.0'.

Output is predicted by ML model successfully.