Phase 5
ML Model Deployment with IBM Watson Studio
(ISP Customer Churn Model)

Objective and Design Thinking :

The project aims to create a predictive model using IBM Watson Studio to identify and forecast customer churn for an Internet Service Provider. The objective is to anticipate customer churn, enabling proactive measures to retain customers. The process began by gathering historical data on customer interactions, followed by data cleaning, exploration, and analysis. Feature engineering and model development were executed in IBM Watson Studio.

Development Phases Overview :

Phase 1:

Defined the problem of customer churn in the ISP industry and collected historical data for analysis.

Phase 2:

Initiated the machine learning model deployment using IBM Watson Studio. The steps involved creating an IBM Cloud account, adding the dataset, preparing the data, and creating the model using SPSS Modular Flow.

Phase 3:

Outlined the model creation process using SPSS Modular Flow, represented through the distribution graph of churn (0 and 1 - False & True). Saved the model at this stage.

Predictive Use Case, Dataset Selection, and Model Training

Use Case Description:

To predict customer churn, the project focused on analyzing customer interactions, service usage, billing, and churn outcomes. Dataset Selection: The dataset was obtained from Koogle.com, containing customer profiles, usage patterns, contract details, and churn outcomes.

Model Training:

Employed machine learning algorithms (e.g., logistic regression, decision trees) to train and evaluate models for churn prediction using IBM Watson Studio.

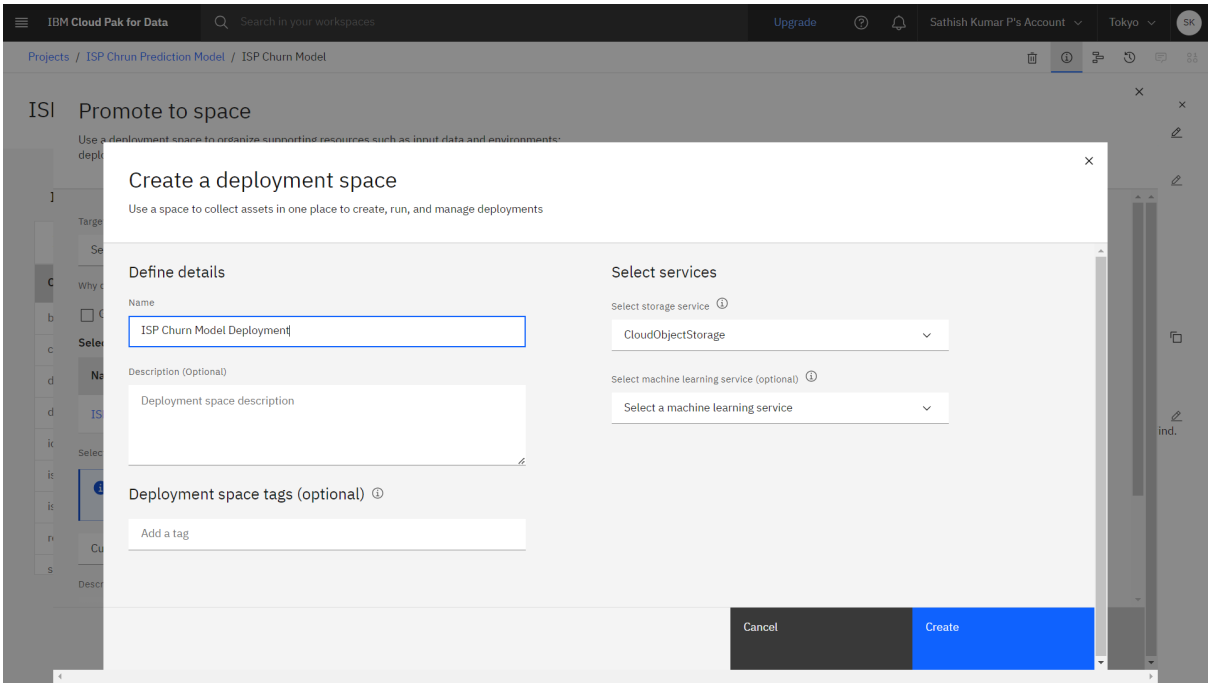
Deployment Process and Integration Steps :

Model Deployment:

Deployed the selected model as an API using Watson Studio, enabling predictions on new data.

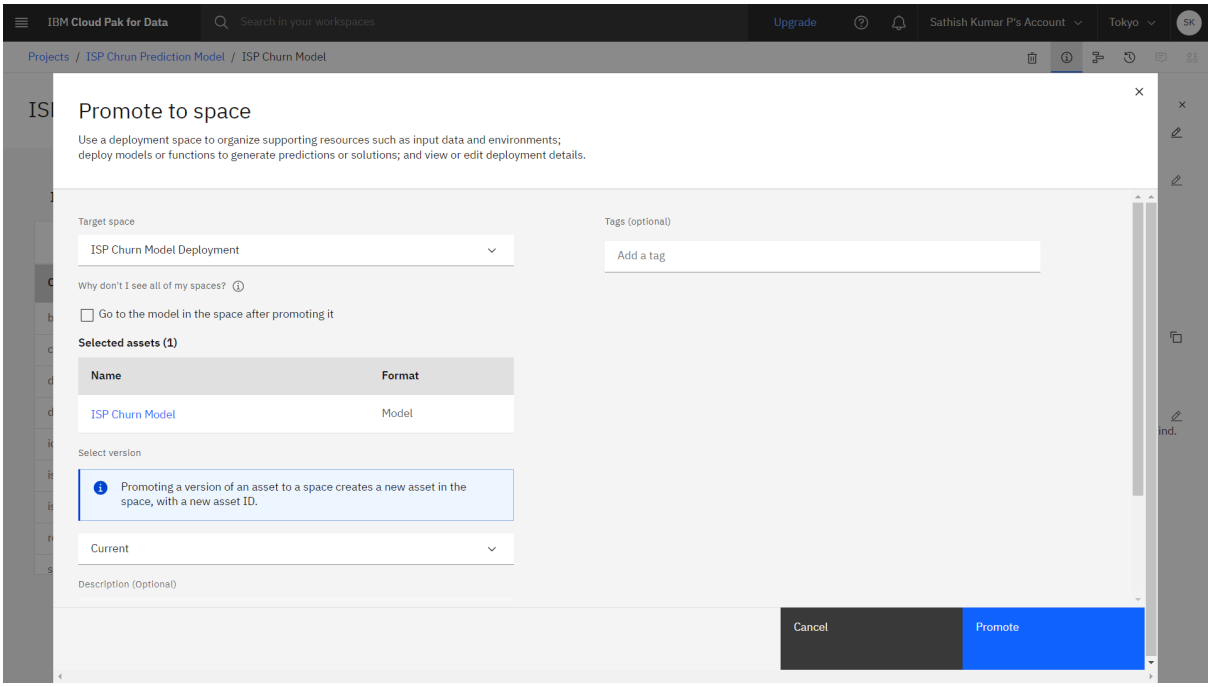
Step:01

We need to create a deployment space to deploy the model



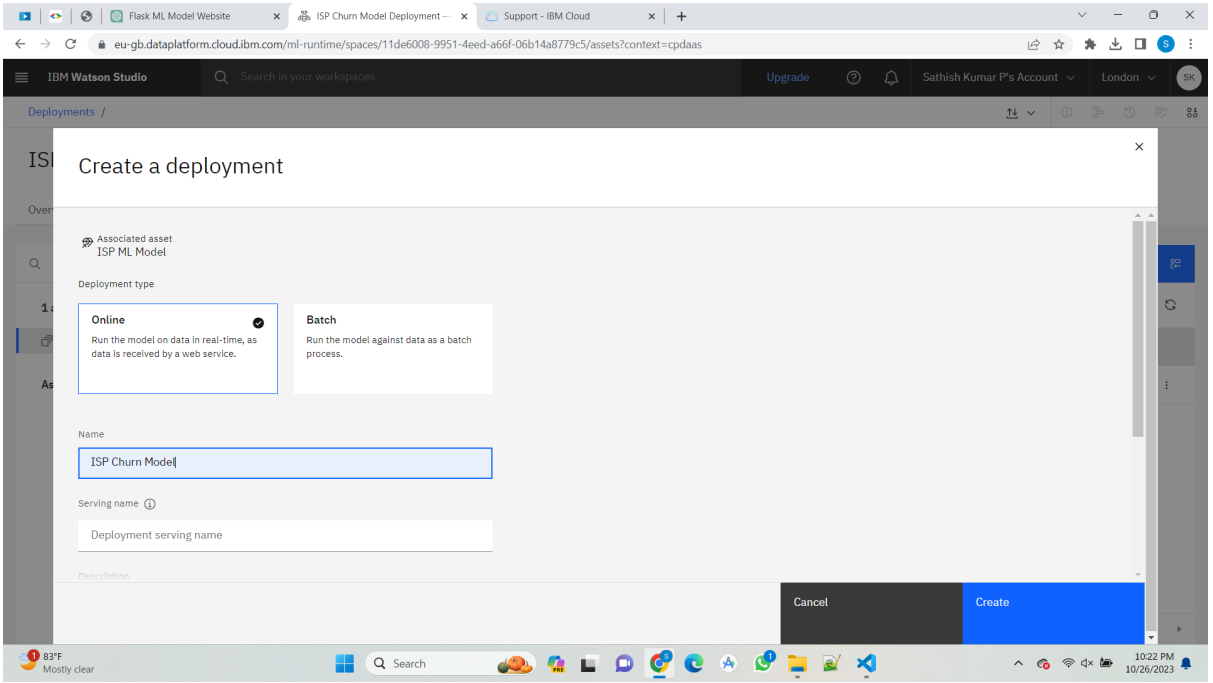
Step:02

Then we need to promote the model to the deployment space



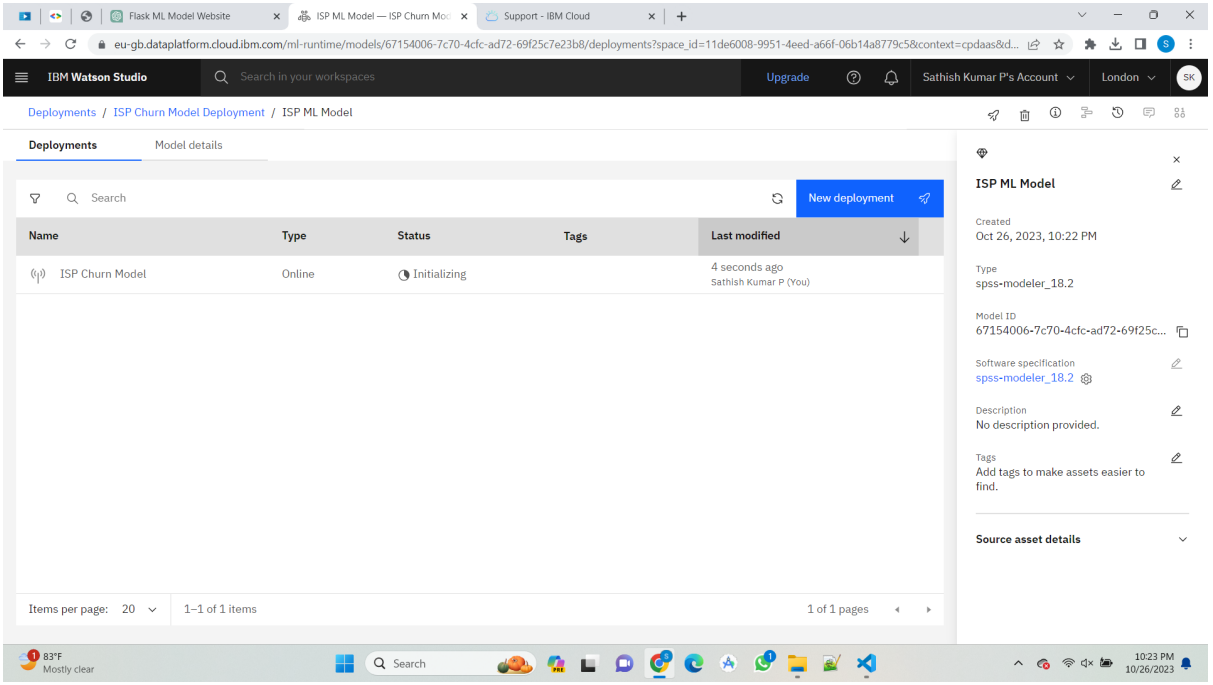
Step:03

The next step is go to the deployment space and click the three dots in the model and click the option deploy as online



Step:04

The it takes few minutes to model be online



Integration Steps:

Integrated the deployed model API with the ISP's infrastructure, setting up the necessary endpoints for real-time predictions. Accessing and Utilizing the Deployed

Step:01

Create a web page using html,css and javascript to get the input and predict.

index.html

```
<!DOCTYPE html>
<html>
<head>
    <title>ISP Churn Model</title>
    <link rel="stylesheet" type="text/css" href="styles.css">
</head>
<body>
    <h1>ISP Churn Model</h1>
    <form method="post" action="/predict">
        <label for="subscription_age">Subscription Age:</label>
        <input type="number" name="subscription_age" required><br>
```

```
        <label for="bill_avg">Average Bill Amount:</label>
        <input type="number" name="bill_avg" required><br>

        <label for="service_failure_count">Service Failure Count:</label>
        <input type="number" name="service_failure_count" required><br>

        <label for="download_avg">Average Download Speed:</label>
        <input type="number" name="download_avg" required><br>

        <label for="upload_avg">Average Upload Speed:</label>
        <input type="number" name="upload_avg" required><br>

        <label for="download_over_limit">Downloads Over Limit (0 for No, 1 for
Yes):</label>
        <input type="number" name="download_over_limit" required><br>

        <input type="submit" value="Predict">
    </form>
</body>
</html>
```

Step:02

Then the next step is to create a flask program to integrate the ML model we copy the code from the below deploy model and connect to the web page.

Flask code:

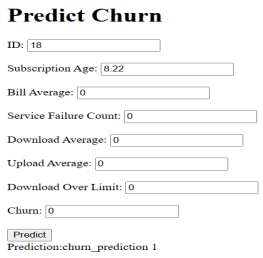
app.py

```
import time
from flask import Flask, render_template, request, jsonify
import requests

app = Flask(__name__)

API_KEY = "iost7rxsblFLvgbMMQkjQvP_xowI0o8J_vochvkLafFN"

ENDPOINT_URL="https://private.eu-de.ml.cloud.ibm.com/ml/v4/deployments/intern
et_churn/predictions?version=2021-05-01"
```

```
def make_prediction(id, subscription_age, bill_avg, service_failure_count,
download_avg, upload_avg, download_over_limit, churn):
    max_retries = 5
    for attempt in range(max_retries):
        try:
            input_data = {
                "input_data": [
                    {
                        "fields": ["ID", "Subscription_Age", "Bill_Average",
"Service_Failure_Count", "Download_Average", "Upload_Average",
"Download_Over_Limit", "Churn"],
                        "values": [[id, subscription_age, bill_avg, service_failure_count,
download_avg, upload_avg, download_over_limit, churn]]
                    }
                ]
            }


            token_response = requests.post('https://iam.cloud.ibm.com/identity/token',
data={"apikey": API_KEY, "grant_type": 'urn:ibm:params:oauth:grant-type:apikey'})
            token_response.raise_for_status()
            mltoken = token_response.json()["access_token"]


            headers = {'Content-Type': 'application/json', 'Authorization': 'Bearer ' +
mltoken}
            response_scoring = requests.post(ENDPOINT_URL, json=input_data,
headers=headers)
            response_scoring.raise_for_status()


            prediction_result = response_scoring.json()

            return prediction_result
        except Exception as e:
            if attempt < max_retries - 1:
                time.sleep(1)
            else:
                return {"error": str(e)}
```

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

@app.route('/predict', methods=['POST'])
def predict():
    try:
        id = request.form['id']
        subscription_age = float(request.form['subscription_age'])
        bill_avg = float(request.form['bill_avg'])
        service_failure_count = float(request.form['service_failure_count'])
        download_avg = float(request.form['download_avg'])
        upload_avg = float(request.form['upload_avg'])
        download_over_limit = float(request.form['download_over_limit'])
        churn = float(request.form['churn'])

            prediction = make_prediction(id, subscription_age, bill_avg,
service_failure_count, download_avg, upload_avg, download_over_limit, churn)

        return jsonify({"prediction": prediction})

    except Exception as e:
        return jsonify({"error": str(e)})

if __name__ == '__main__':
    app.run(debug=True)
```

Sample output:



Model for Real-Time Predictions:

The deployed model can be accessed through the API, allowing real-time predictions as customers interact with the ISP's services. This aids in proactive retention efforts by identifying customers likely to churn, enabling timely intervention to prevent churn.

Conclusion :

This project using IBM Watson Studio successfully developed a predictive model to anticipate ISP customer churn, enabling proactive measures. The deployed model, accessible via an API, empowers real-time predictions, assisting the ISP in curbing churn and ensuring sustained business success within the competitive market.