

Project: Machine Learning Model Deployment with IBM Cloud Watson Studio

Phase_5: Project Documentation & Submission

Problem Statement: Sign Language Translation

Problem Definition:

- The objective of this project is to develop a predictive analytics solution by training a machine learning model using IBM Cloud Watson Studio and deploying it as a web service. The primary goal is to gain proficiency in predictive analytics and create a model capable of making real-time predictions.
- Building a real-time hand gesture recognition system for sign language translation or human-computer interaction is an exciting and challenging project. This system involves using machine learning and computer vision techniques to interpret gestures made by a person's hand in real time and then translating those gestures into meaningful actions or messages.
- A gesture translator is a system that interprets hand gestures or body movements and converts them into meaningful commands or text. This technology can be used in various applications. In this project, we'll focus on building a simple sign language recognition model using IBM Watson Studio
- In a world characterized by diverse languages, cultures, and communication styles, the ability to understand and be understood is paramount. Gesture translation technology is a remarkable innovation that addresses this challenge by enabling effective communication between individuals who use non-verbal gestures, such as sign language, and those who may not be familiar with these forms of expression.

Design Thinking:

1. Predictive Use Case:

Determine a specific use case for predictive analytics. This could involve predicting customer churn, forecasting product demand, or any other relevant scenario where predictive insights are valuable

Objectives:

- The first step is to clearly define the problem we aim to solve with predictive analytics.
- This involves identifying a specific use case that has practical value.

Action Steps:

- Conduct stakeholder interviews and gather insights to understand the business needs.
- Identify potential use cases, such as predicting customer churn, forecasting sales, or optimizing resource allocation.
- Select one use case that aligns with business goals and data availability.

2. Dataset Selection:

Choose an appropriate dataset that aligns with the selected predictive use case. The dataset should contain relevant features and historical data necessary for model training.

Objectives:

To build an effective predictive model, we need high-quality data that includes historical records and relevant features

Action Steps:

- Explore available data sources, including internal databases and external datasets.
- Evaluate the quality and completeness of potential datasets.
- Select a dataset that aligns with the chosen predictive use case and contains the necessary attributes for model training.

3. Model Training:

Select a suitable machine learning algorithm for the predictive task. Utilize IBM Cloud Watson Studio's tools and resources to preprocess the dataset, train the machine learning model, and evaluate its performance.

Objectives:

Train a machine learning model that can make accurate predictions based on historical data.

Action Steps:

- Preprocess the selected dataset by handling missing values, feature engineering, and data scaling.
- Choose an appropriate machine learning algorithm (e.g., regression, classification, time series forecasting) based on the nature of the problem.
- Split the data into training and validation sets for model evaluation.
- Train and fine-tune the model using IBM Cloud Watson Studio's tools.
- The model's performance through metrics like accuracy, precision, recall, or RMSE (Root Mean Square Error).

4. Model Deployment:

Deploy the trained machine learning model as a web service using IBM Cloud Watson Studio's deployment capabilities. This step ensures that the model can be accessed and utilized via API endpoints.

Objectives:

Deploy the trained machine learning model as a web service to make it accessible for real-time predictions.

Action Steps:

- Utilize IBM Cloud Watson Studio's deployment capabilities to package the model.
- Set up API endpoints to enable external applications to interact with the model.
- Ensure scalability, security, and reliability of the deployed service.

5. Integration:

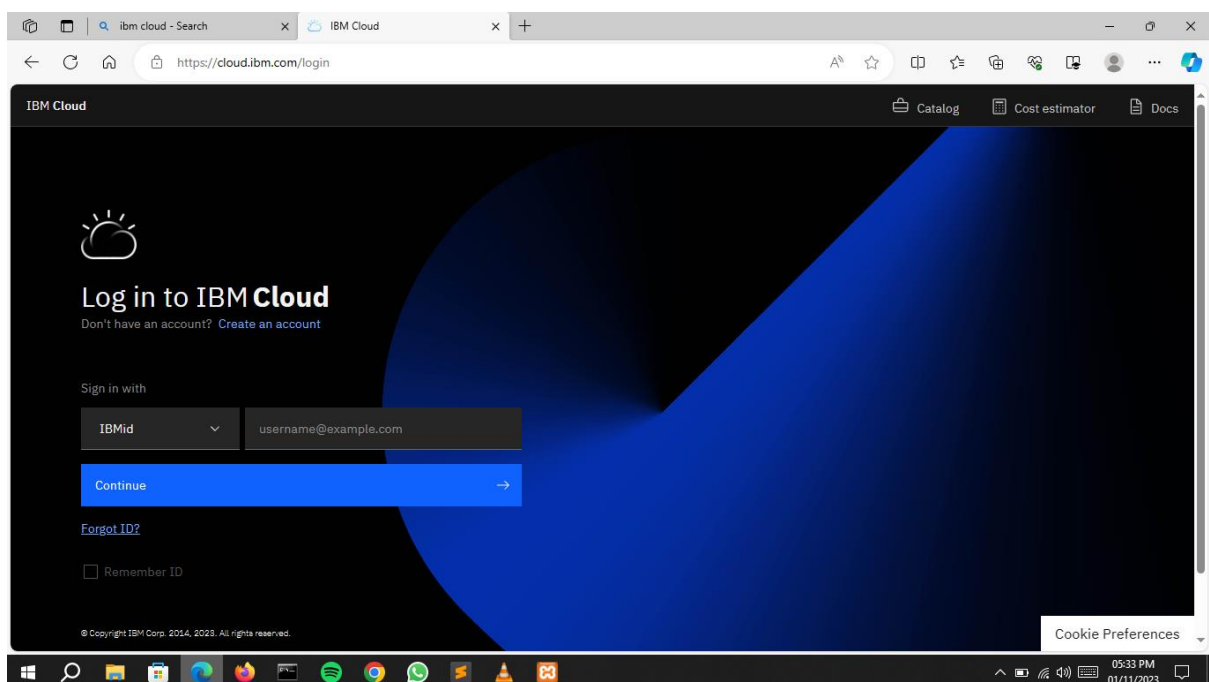
Integrate the deployed model into applications or systems where real-time predictions are required. This integration will enable the model to provide predictions on new data as it becomes available.

Objectives:

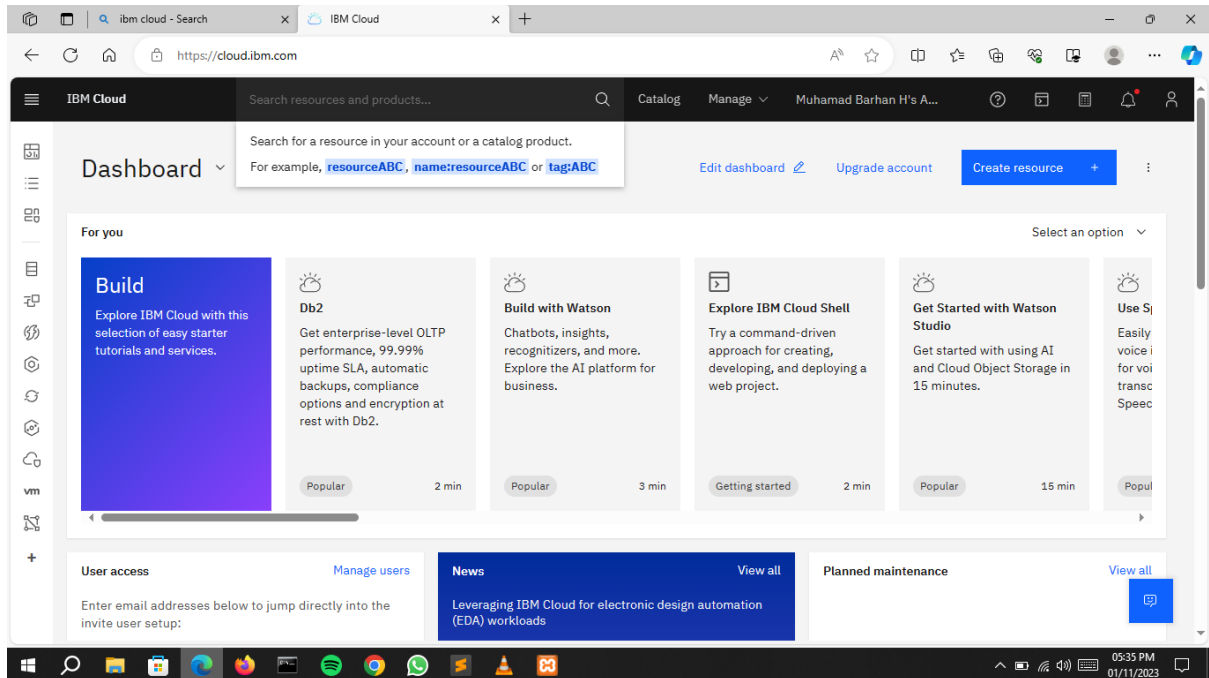
Integrate the deployed model into relevant applications or systems to enable real-time predictions.

Action Steps:

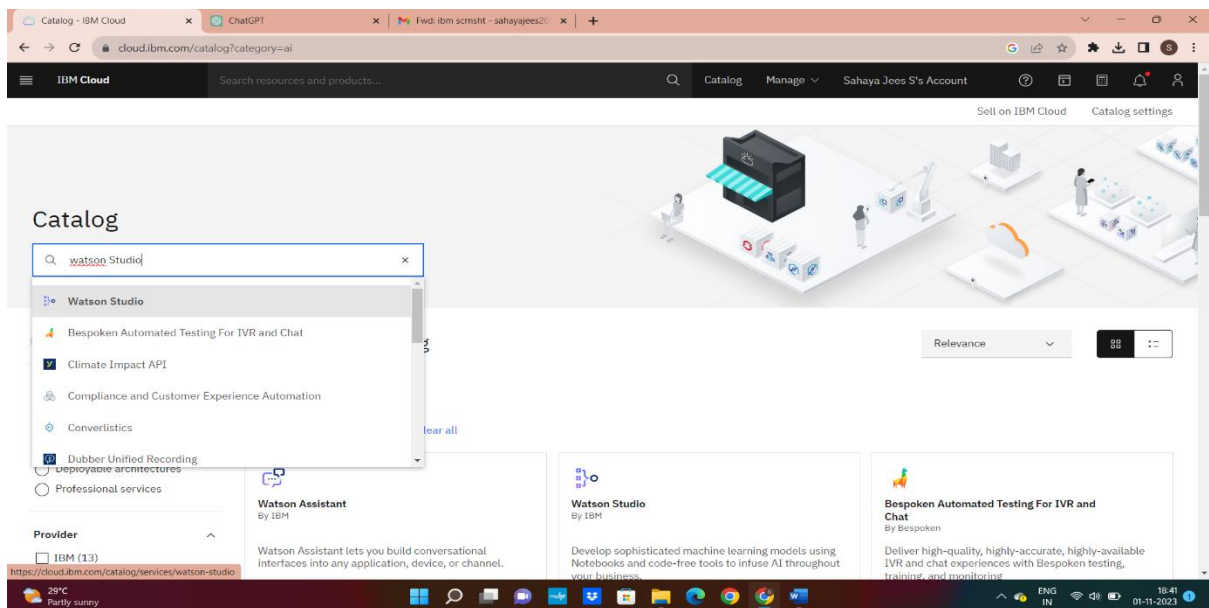
- Collaborate with the development team to integrate the model into the target applications.
 - Establish a mechanism for data input and model output, ensuring seamless communication.
 - Implement monitoring and error-handling procedures to maintain the model's performance in production.
-
- ❖ Now we are going to create the gesture translator for that we will do the primary steps now.
 - ❖ To access the IBM Cloud Watson Studio, we have to create an IBM Cloud Account.
 - ❖ To create it follow the below steps.
-
- ❖ **Go to the IBM Cloud website:** Visit the IBM Cloud website at <https://cloud.ibm.com/>.
-
- ❖ **Sign up or Log in:** If you don't already have an IBM Cloud account, click "Sign Up" to create one. If you have an account, log in with your credentials.



- ❖ **Navigation to IBM Watson Studio:** Once logged in, you'll be taken to the IBM Cloud dashboard. In the top menu, click on "Catalog."

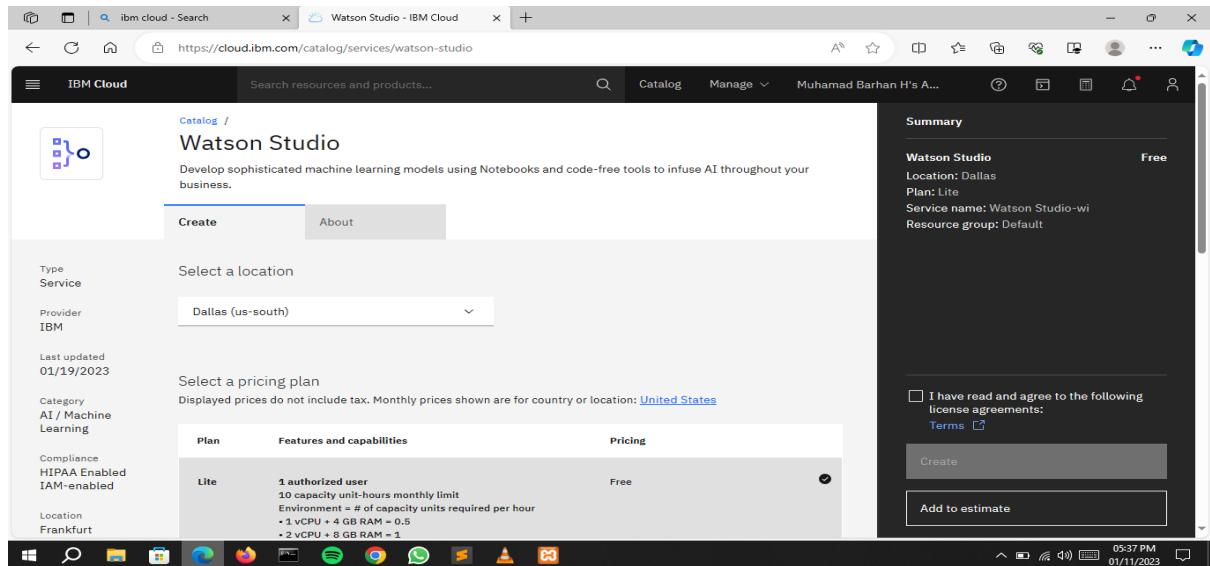


- ❖ **Search For Watson Studio:** In the catalog, you can use the search bar to find "Watson Studio" or browse through the services to locate it.



- ❖ **Create a Watson Studio service:** Click on the Watson Studio service to access its details. Then, click the "Create" button to set up a new Watson Studio instance.

- ❖ **Configure your Watson Studio service:** You'll need to provide some basic information for your Watson Studio service, such as a name, region, and resource group. You can also choose the pricing plan that suits your needs.



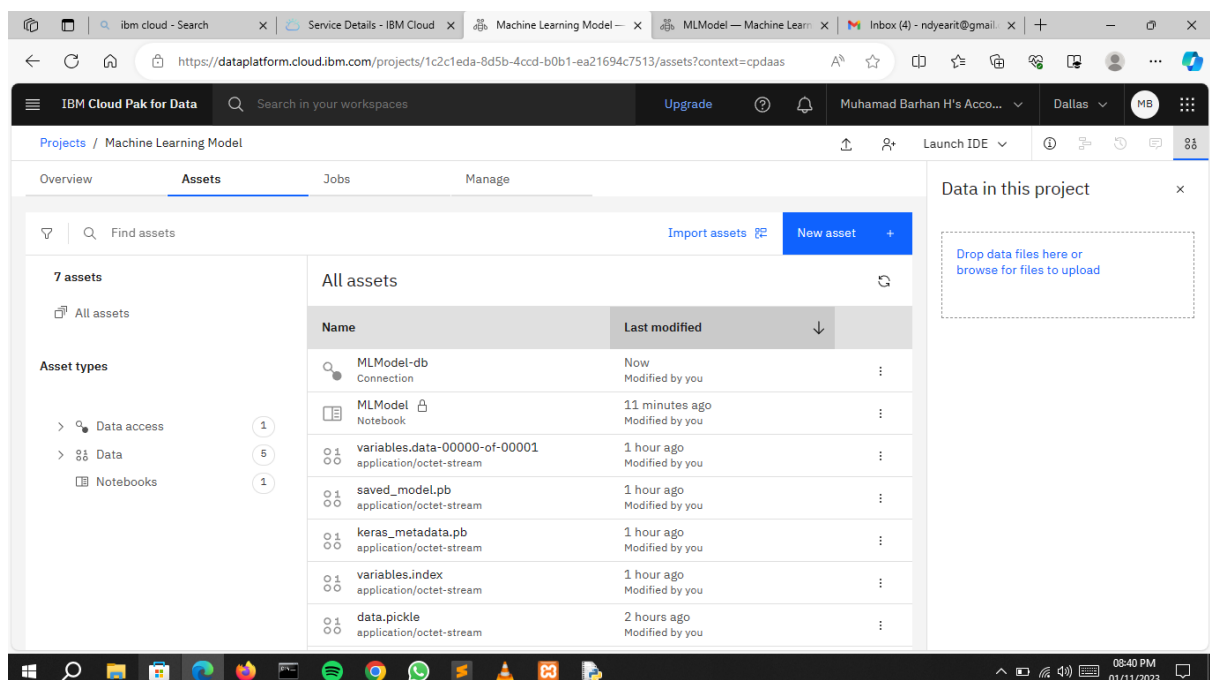
- ❖ **Create the service:** After configuring the service, click the "Create" button to create your Watson Studio instance.

Overview of dataset:

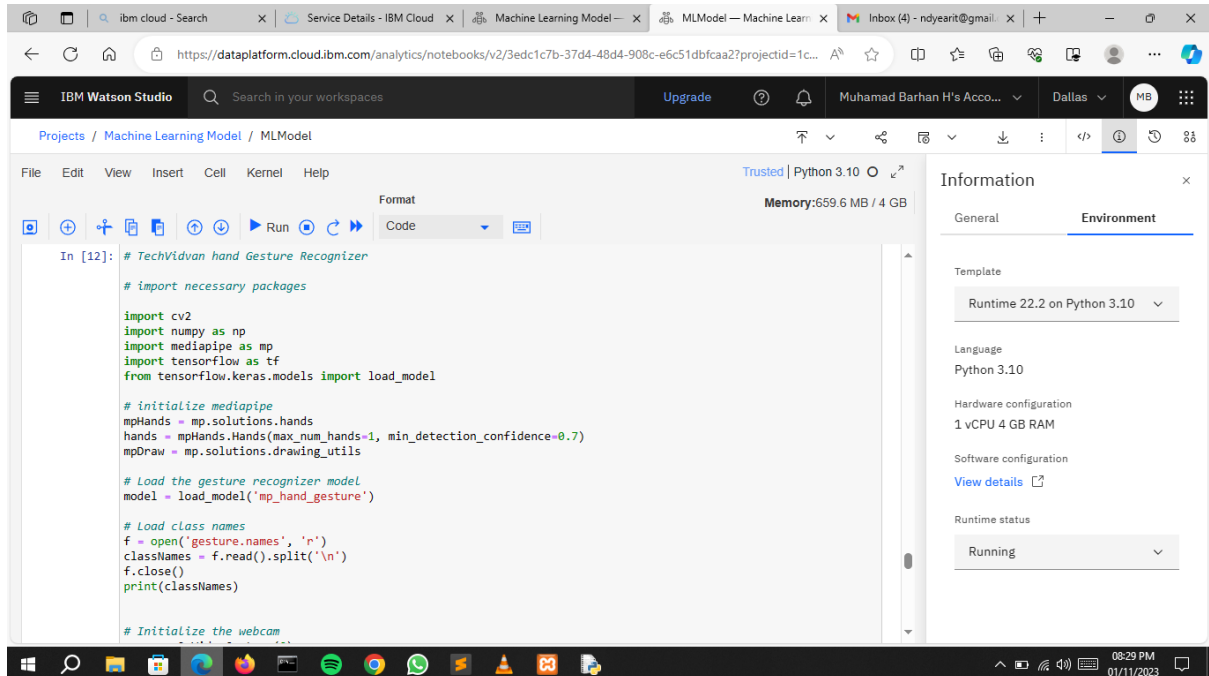
In IBM Cloud Watson Studio, you can work with datasets for training and testing machine learning models. Here is an overview of the process to prepare and split your dataset into training and testing sets.

- ❖ **Data upload:** First, you need to upload your dataset to Watson Studio. You can do this by following these steps:
 - Log in to your IBM Cloud account and access Watson Studio.
 - Create a new project or use an existing one.
 - Within your project, create a new data asset or data file that represents your dataset.
 - Upload your dataset to the data asset.

- ❖ **Data Exploration and Preprocessing:** Before splitting the dataset, it's a good practice to explore and preprocess your data. This may involve tasks like handling missing values, data transformation, and feature engineering. Watson Studio provides tools and notebooks for this purpose
- ❖ **Data Splitting:** To split your dataset into training and testing sets, you can use Python or other programming languages within Watson Studio's integrated environment.
- ❖ **Saving the Split Datasets:** After splitting the dataset, you can save the training and testing datasets back to your project in Watson Studio. This makes them easily accessible for building and evaluating machine learning models.
- ❖ **Machine Learning Model Building:** You can use the training dataset to train your machine learning model within Watson Studio's environment
- ❖ **Model Evaluation:** Once your model is trained, you can use the testing dataset to evaluate its performance. Watson Studio provides tools for model evaluation and metrics calculation



Sample Code:



The screenshot shows the IBM Watson Studio interface. The top navigation bar includes the IBM logo, search bar, and user profile. The main workspace displays a Jupyter notebook titled "MLModel". The code in the notebook is as follows:

```
In [12]: # TechVidvan hand Gesture Recognizer

# import necessary packages

import cv2
import numpy as np
import mediapipe as mp
import tensorflow as tf
from tensorflow.keras.models import load_model

# initialize mediapipe
mpHands = mp.solutions.hands
hands = mpHands.Hands(max_num_hands=1, min_detection_confidence=0.7)
mpDraw = mp.solutions.drawing_utils

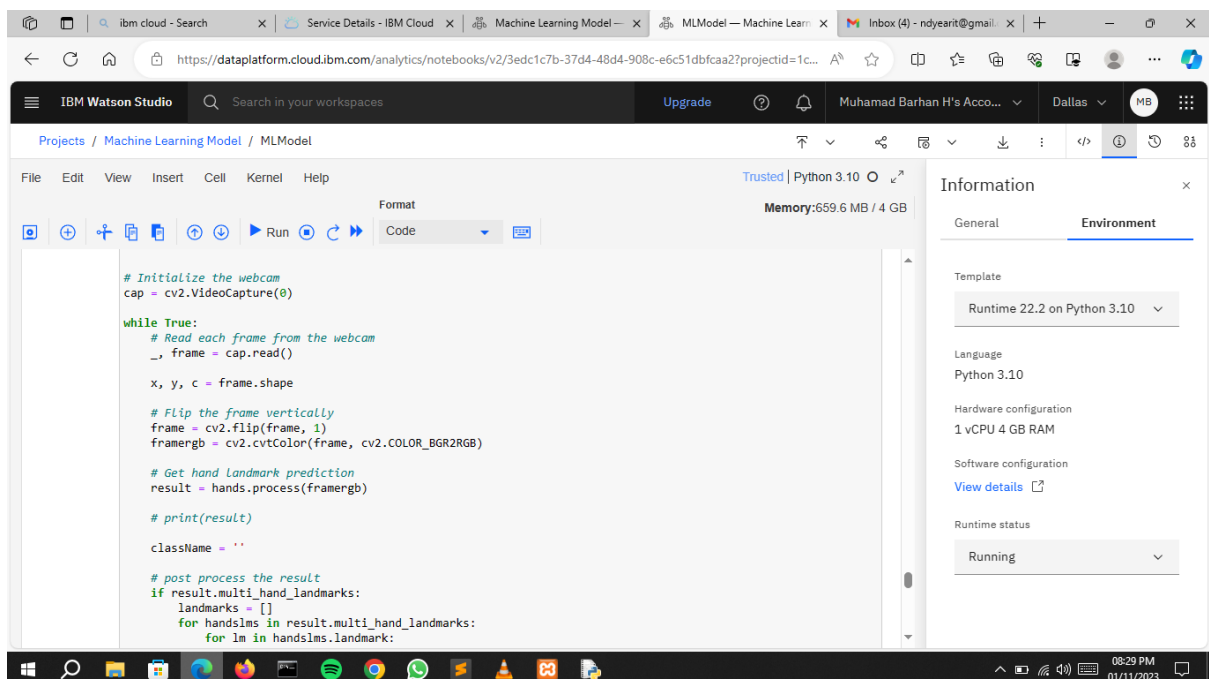
# Load the gesture recognizer model
model = load_model('mp_hand_gesture')

# Load class names
f = open('gesture.names', 'r')
classNames = f.read().split('\n')
f.close()
print(classNames)

# Initialize the webcam
```

The right sidebar shows the "Information" panel with the "Environment" tab selected. It displays the following details:

- Template: Runtime 22.2 on Python 3.10
- Language: Python 3.10
- Hardware configuration: 1 vCPU 4 GB RAM
- Software configuration: [View details](#)
- Runtime status: Running



The screenshot shows the IBM Watson Studio interface. The top navigation bar includes the IBM logo, search bar, and user profile. The main workspace displays a Jupyter notebook titled "MLModel". The code in the notebook is as follows:

```
# Initialize the webcam
cap = cv2.VideoCapture(0)

while True:
    # Read each frame from the webcam
    _, frame = cap.read()

    x, y, c = frame.shape

    # Flip the frame vertically
    frame = cv2.flip(frame, 1)
    framergb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

    # Get hand Landmark prediction
    result = hands.process(framergb)

    # print(result)

    className = ''

    # post process the result
    if result.multi_hand_landmarks:
        landmarks = []
        for lm in result.multi_hand_landmarks:
            for lm in hands.lms.Landmark:
```

The right sidebar shows the "Information" panel with the "Environment" tab selected. It displays the following details:

- Template: Runtime 22.2 on Python 3.10
- Language: Python 3.10
- Hardware configuration: 1 vCPU 4 GB RAM
- Software configuration: [View details](#)
- Runtime status: Running

IBM Watson Studio interface showing a Jupyter Notebook environment. The browser tabs include "ibm cloud - Search", "Service Details - IBM Cloud", "Machine Learning Model", "MLModel - Machine Learning", and "Inbox (4) - ndyearit@gmail.com". The URL is <https://dataplatform.cloud.ibm.com/analytics/notebooks/v2/3edc1c7b-37d4-48d4-908c-e6c51dbfcaa2?projectId=1c...>. The interface shows the "Machine Learning Model" project with the "MLModel" notebook open. The code in the notebook is as follows:

```
# post process the result
if result.multi_hand_landmarks:
    landmarks = []
    for hands in result.multi_hand_landmarks:
        for lm in hands.landmark:
            # print(id, lm)
            lmx = int(lm.x * x)
            lmy = int(lm.y * y)

            landmarks.append([lmx, lmy])

# Drawing Landmarks on frames
mpDraw.draw_landmarks(frame, hands, mpHands.HAND_CONNECTIONS)

# Predict gesture
prediction = model.predict([landmarks])
# print(prediction)
classID = np.argmax(prediction)
className = classNames[classID]

# show the prediction on the frame
cv2.putText(frame, className, (10, 50), cv2.FONT_HERSHEY_SIMPLEX,
            1, (0,0,255), 2, cv2.LINE_AA)

# Show the final output
```

The "Information" panel on the right shows the environment details:

- General: Template: Runtime 22.2 on Python 3.10
- Language: Python 3.10
- Hardware configuration: 1 vCPU 4 GB RAM
- Software configuration: View details
- Runtime status: Running

The bottom status bar shows the time as 08:30 PM on 01/11/2023.

IBM Watson Studio interface showing a Jupyter Notebook environment. The browser tabs include "ibm cloud - Search", "Service Details - IBM Cloud", "Machine Learning Model", "MLModel - Machine Learning", and "Inbox (4) - ndyearit@gmail.com". The URL is <https://dataplatform.cloud.ibm.com/analytics/notebooks/v2/3edc1c7b-37d4-48d4-908c-e6c51dbfcaa2?projectId=1c...>. The interface shows the "Machine Learning Model" project with the "MLModel" notebook open. The code in the notebook is as follows:

```
lmy = int(lm.y * y)

landmarks.append([lmx, lmy])

# Drawing Landmarks on frames
mpDraw.draw_landmarks(frame, hands, mpHands.HAND_CONNECTIONS)

# Predict gesture
prediction = model.predict([landmarks])
# print(prediction)
classID = np.argmax(prediction)
className = classNames[classID]

# show the prediction on the frame
cv2.putText(frame, className, (10, 50), cv2.FONT_HERSHEY_SIMPLEX,
            1, (0,0,255), 2, cv2.LINE_AA)

# Show the final output
cv2.imshow("Output", frame)

if cv2.waitKey(1) == ord('q'):
    break

# release the webcam and destroy all active windows
cap.release()
```

The "Information" panel on the right shows the environment details:

- General: Template: Runtime 22.2 on Python 3.10
- Language: Python 3.10
- Hardware configuration: 1 vCPU 4 GB RAM
- Software configuration: View details
- Runtime status: Running

The bottom status bar shows the time as 08:30 PM on 01/11/2023.

Integration techniques provided by the IBM Cloud Watson Studio:

Website: To integrate your gesture translator in your website you should follow the below steps.

❖ Set Up an IBM Watson Studio Service:

- If you don't already have an IBM Cloud account, sign up using the steps provided in the previous phases.
- Create an instance of IBM Watson Assistant in your IBM Cloud account.

This step is explained in phase 3.

❖ Build and Train Your Studio:

- Define your gesture translator skills, and responses within the Watson Studio tool.

This step is explained in phase 3.

❖ Obtain API Credentials:

- Get the API credentials (API key and URL) for your Watson Studio instance from the IBM Cloud dashboard.

❖ Integrate into Your Website:

- You can integrate Watson Studio into your website using the provided APIs. You have one option:
 - API Integration: You can use the Watson Studio API to build a custom gesture translator for your website.

❖ Test and Deploy:

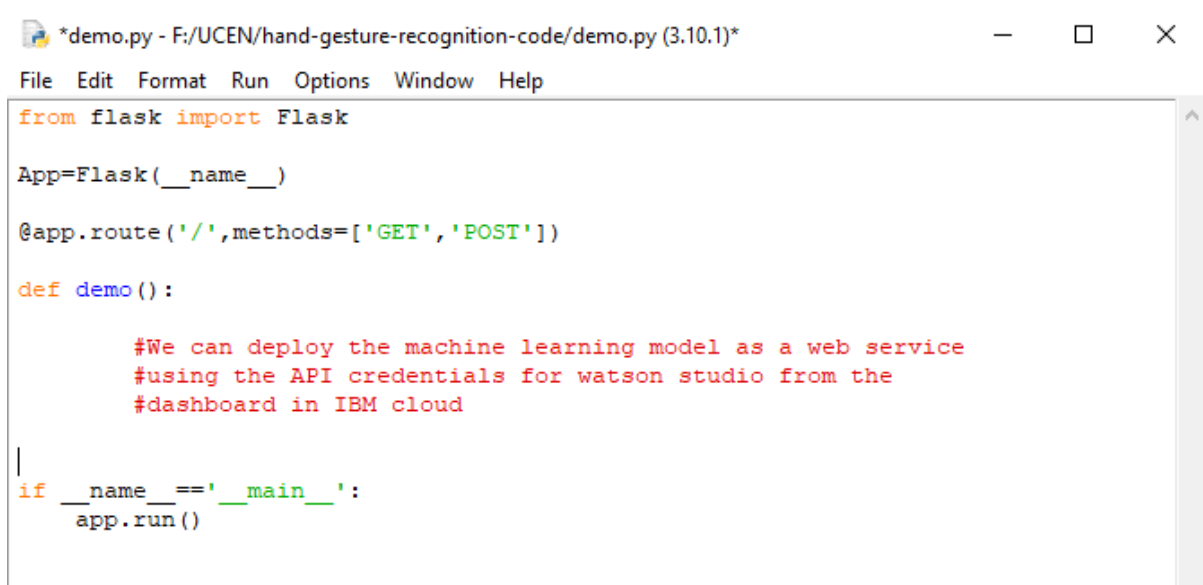
- Test the gesture translator to ensure it works as expected on your website.
- Deploy it to your live website.

❖ Continuous Improvement:

- Monitor the performance of your gesture translator and collect user feedback to make improvements over time.

IBM Watson Studio provides documentation and resources to help you with the integration process.

Deploying as web service:



```
*demo.py - F:/UCEN/hand-gesture-recognition-code/demo.py (3.10.1)*
File Edit Format Run Options Window Help
from flask import Flask

App=Flask(__name__)

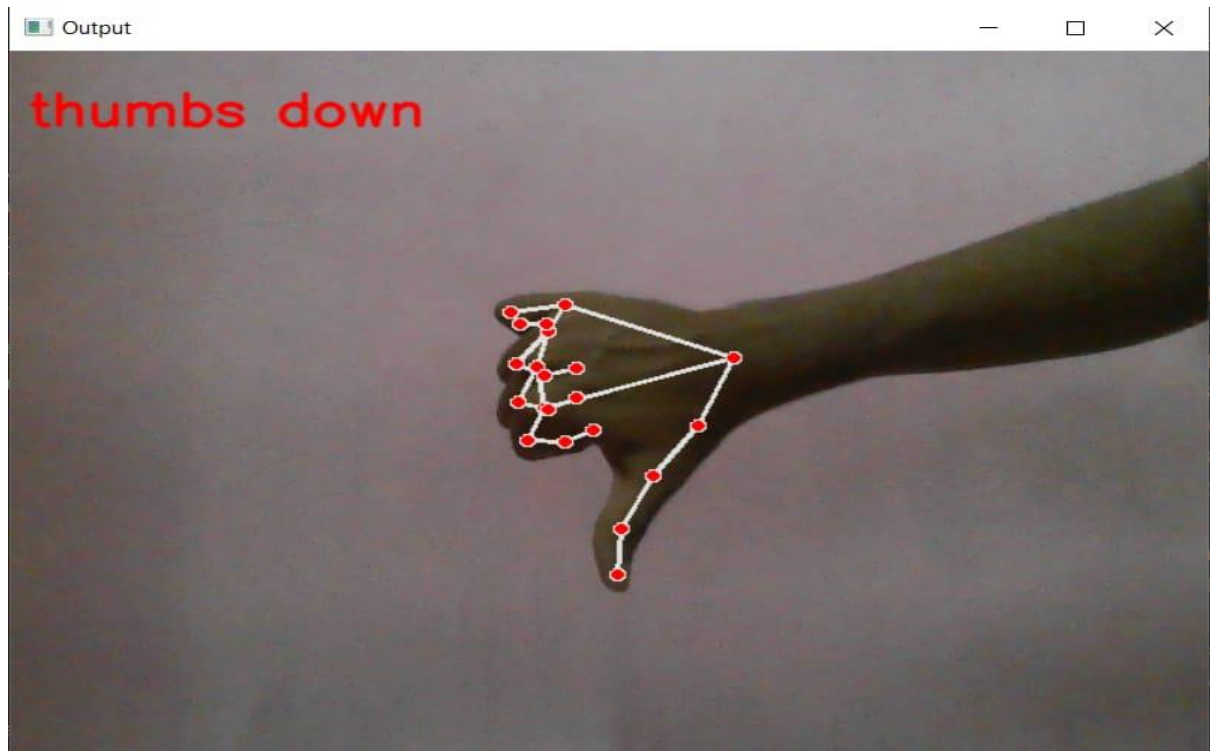
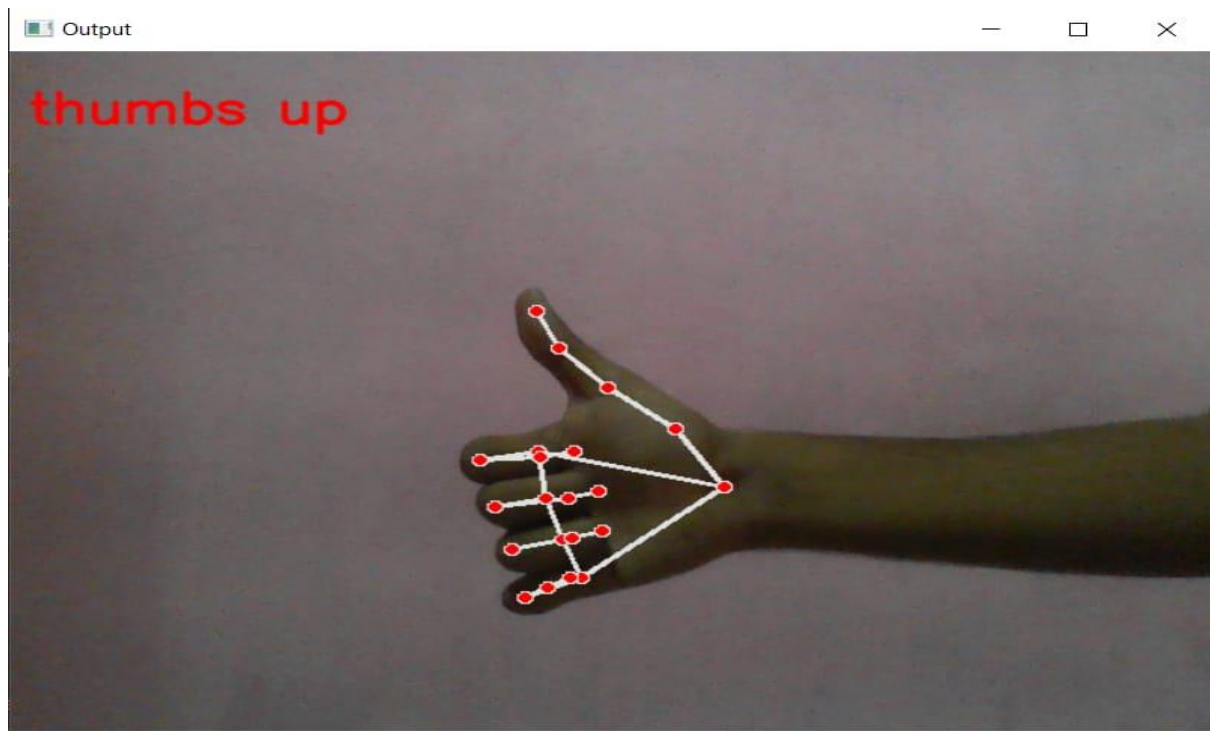
@app.route('/',methods=['GET','POST'])

def demo():

    #We can deploy the machine learning model as a web service
    #using the API credentials for watson studio from the
    #dashboard in IBM cloud

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

Output:



Conclusion:

In conclusion, the application of machine learning to sign language translation represents a promising and transformative field with significant potential to bridge communication gaps between the deaf and hearing communities. Through the use of advanced algorithms and deep learning techniques, we can develop more accurate and efficient sign language translation systems. However, there are still challenges to overcome, such as dialect variations and real-time translation. As technology continues to advance, we can expect further improvements in sign language translation, ultimately enhancing the inclusivity and accessibility for the deaf and hard of hearing individuals in our society.