**AWS Build-a-thon Project Report**

ON

**“Predicting the Energy Output of Wind Turbine based on Weather Conditions”**

Submitted for the aws build-a-thon of

**Machine Learning**

BY

**NAME: RAHUL P.BHERE**

**A project report submitted to:**

Smart Bridge



**ACKNOWLEDGMENT:**

The success and outcome of this project required guidance and assistance from **Smart Bridge**. I am extremely privileged to have got this all along with the completion of my project. All that I have done is only due to such supervision and assistance and I would not forget to thank them. I thank **Smart Bridge**, for providing me an opportunity to do the project work in the AWS build-a-thon **Predicting the Energy Output of Wind Turbine based on Weather Conditions** and giving me all support and guidance which made me complete the project duly. I am extremely thankful to smart bridge for providing such a nice support and guidance.

**1. INTRODUCTION**

**1.1 Web Summarization**

Wind energy plays an increasing role in the supply of energy worldwide. The energy output of a wind farm is highly dependent on the weather conditions present at its site. If the output can be predicted more accurately, energy suppliers can coordinate the collaborative production of different energy sources more efficiently to avoid costly overproduction. In this project, we predict energy prediction based on weather data and analyze the important parameters as well as their correlation on the energy output.

**1.2 Purpose**

The main purpose is to map weather data for energy production. We wish to show that even data that is publicly available for weather stations close to wind farms can be used to give a good prediction of the energy output. Furthermore, we examine the impact of different weather conditions on the energy output of wind farms.

# 2. LITERATURE SURVEY

# 2.1 Existing problem

There exist a number of technological, environmental and political challenges linked to supplementing existing electricity generation capacities with wind energy. Here, mathematicians and statisticians could make a substantial contribution at the interface of meteorology and decision-making, in connection with the generation of forecasts tailored to the various operational decision problems involved. Indeed, while wind energy may be seen as an environmentally friendly source of energy, full benefits from its usage can only be obtained if one is able to accommodate its variability and limited predictability. Based on a short presentation of its physical basics, the importance of considering wind power generation as a stochastic process is motivated. The conventional moving-average statistical models were proven to be less efficient in forecasting the wind energy, as the wind speed is inherently variable quantity.

# 2.2 Proposed solution

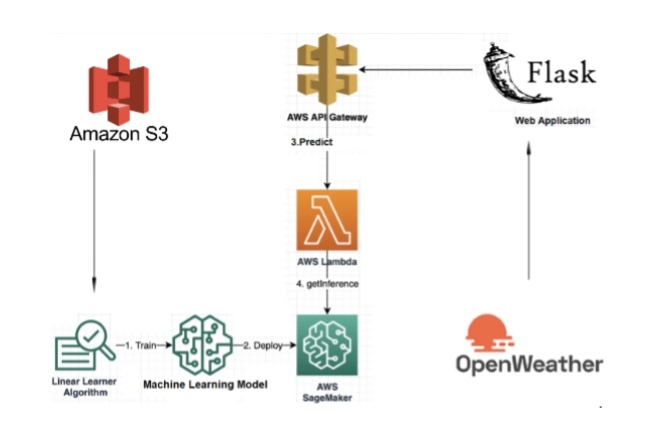
To overcome the disadvantages of conventional models , advanced ensemble models such as XGBoost , can be used to map the inherently variable attribute to a complex function.It is an optimized distributed gradient boosting library designed to be highly efficient , flexible and portable. . It implements machine learning algorithms under the Gradient Boosting framework. XGBoost provides a parallel tree boosting (also known as GBDT, GBM) that solve many data science problems in a fast and accurate way. The same code runs on major distributed environment (Hadoop, SGE, MPI) and can solve problems beyond billions of examples.

# 3.THEORETICAL ANALYSIS

# 3.1 PROJECT SCOPE:

## The main purpose is to map weather data for energy production. We wish to show that even data that is publicly available for weather stations close to wind farms can be used to give a good prediction of the energy output. Furthermore, we examine the impact of different weather conditions on the energy output of wind farms.

## **3.2 Block Digram**



***Fig. 2.****Block Digram*

**3.3 Software designing**

**1. AWS Cloud**

**2. Amazon S3**

**3. AWS API Gateway**

**4. AWS Lambda**

**5. Amazon Sagemaker**

Amazon SageMaker is a fully managed service that provides every developer and data scientist with the ability to build, train, and deploy machine learning (ML) models quickly. SageMaker removes the heavy lifting from each step of the machine learning process to make it easier to develop high quality models. he steps to be followed are

1. Training SageMaker Model

2. Deploying SageMaker Model

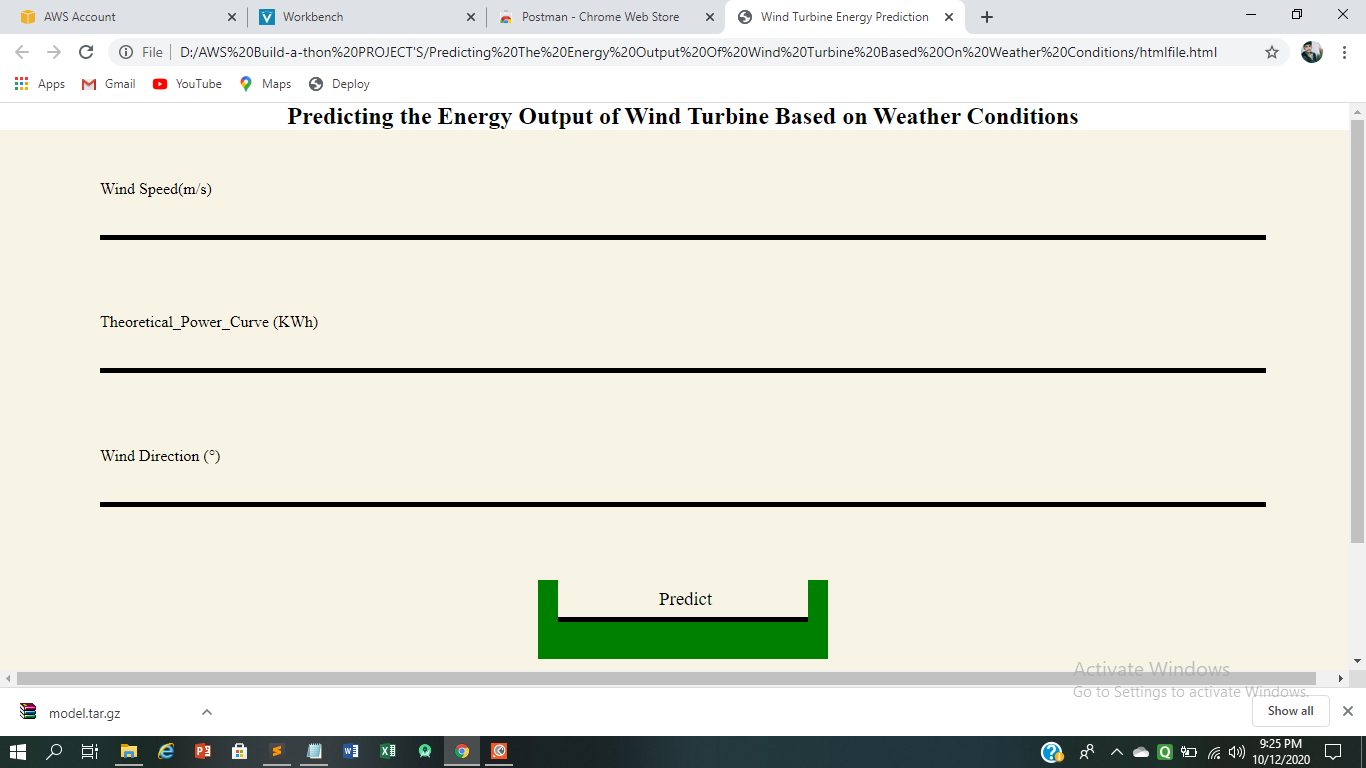
3. Creating Lambda

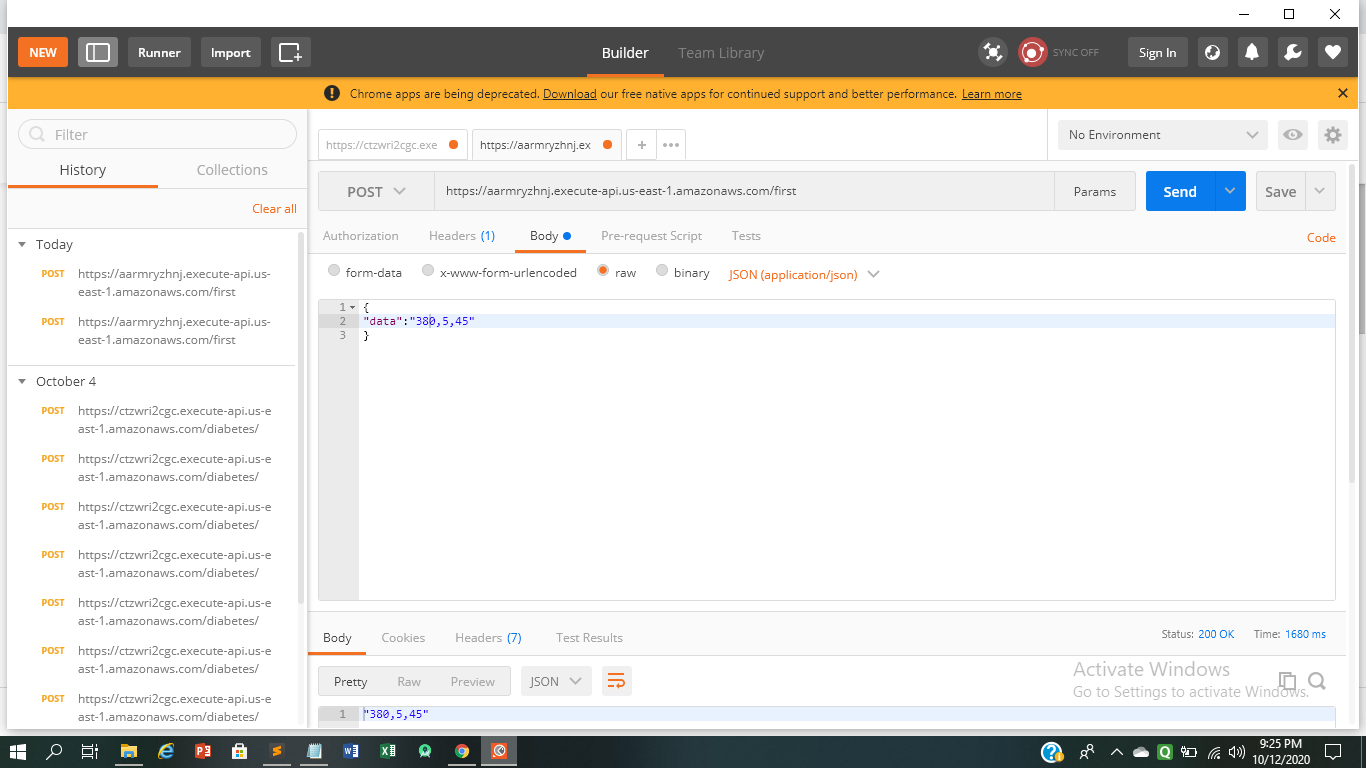
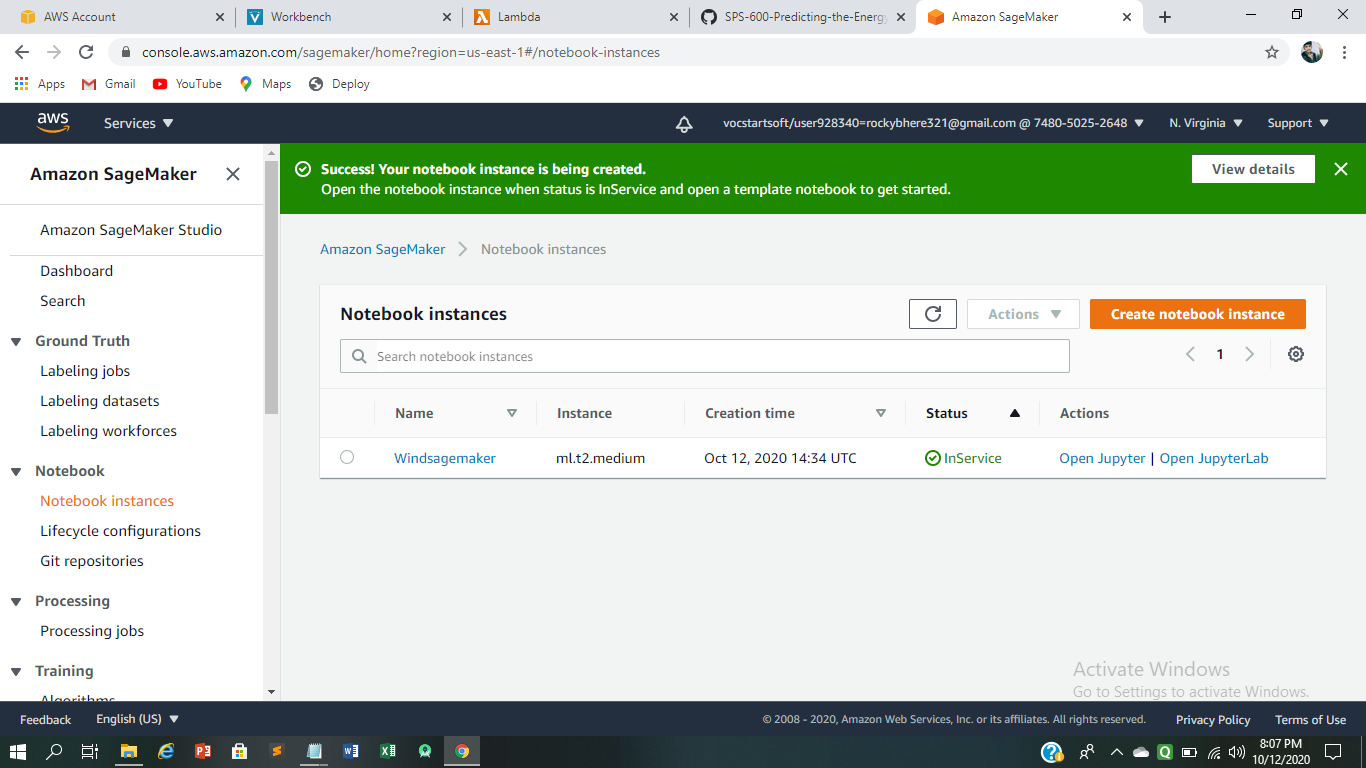
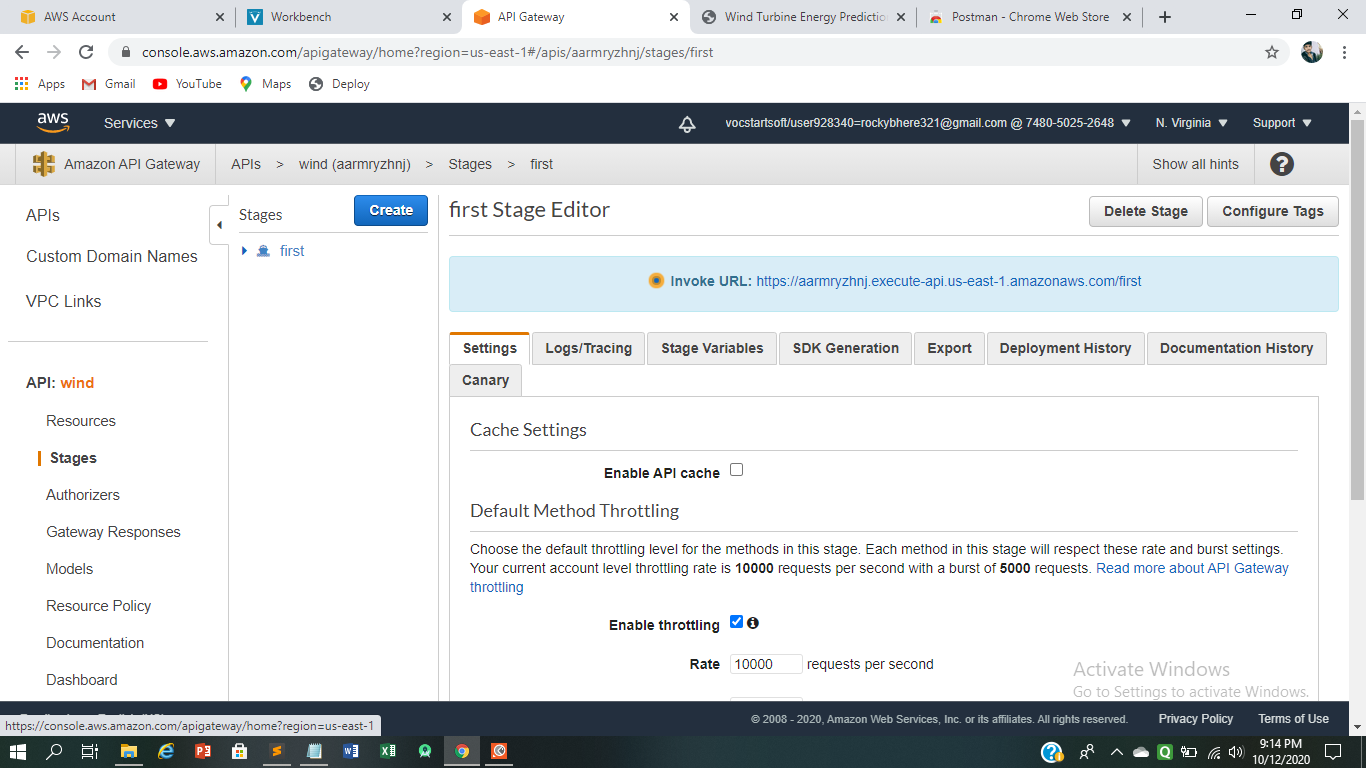
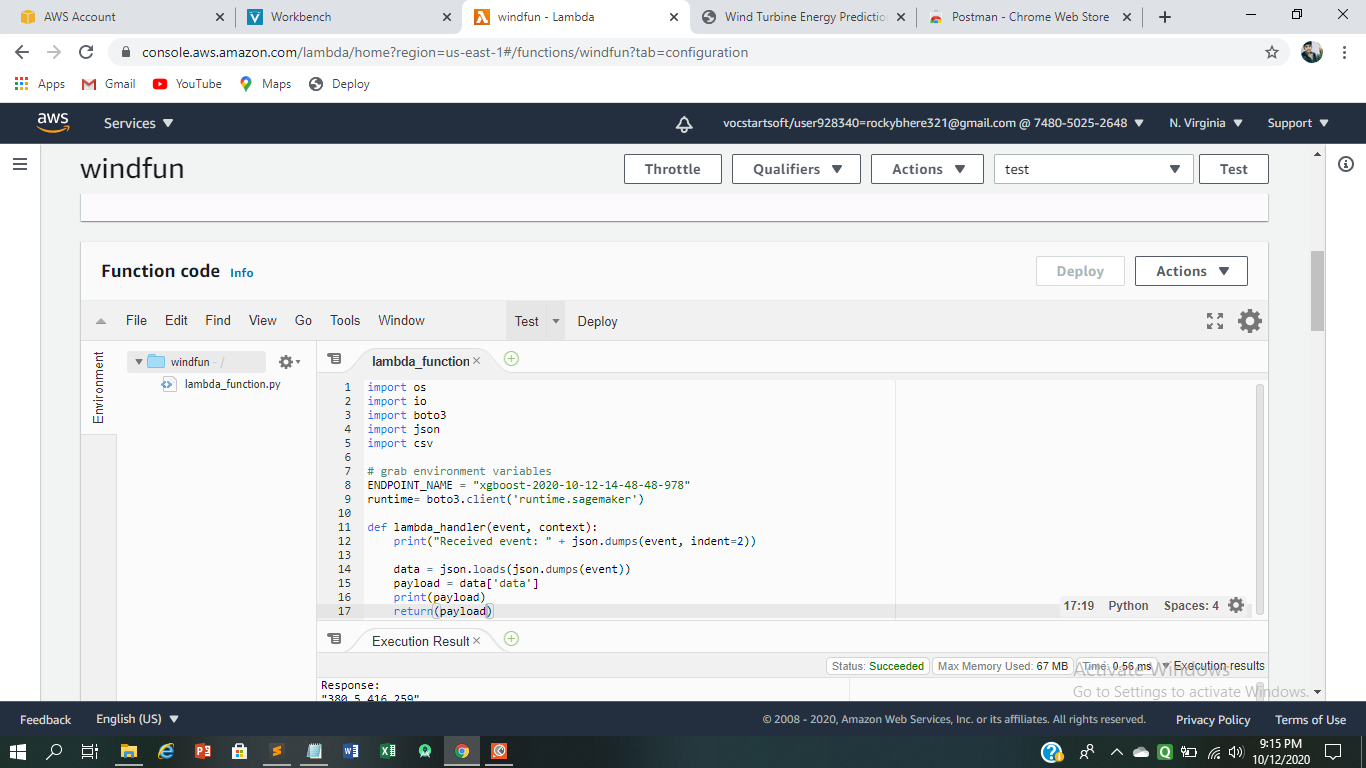
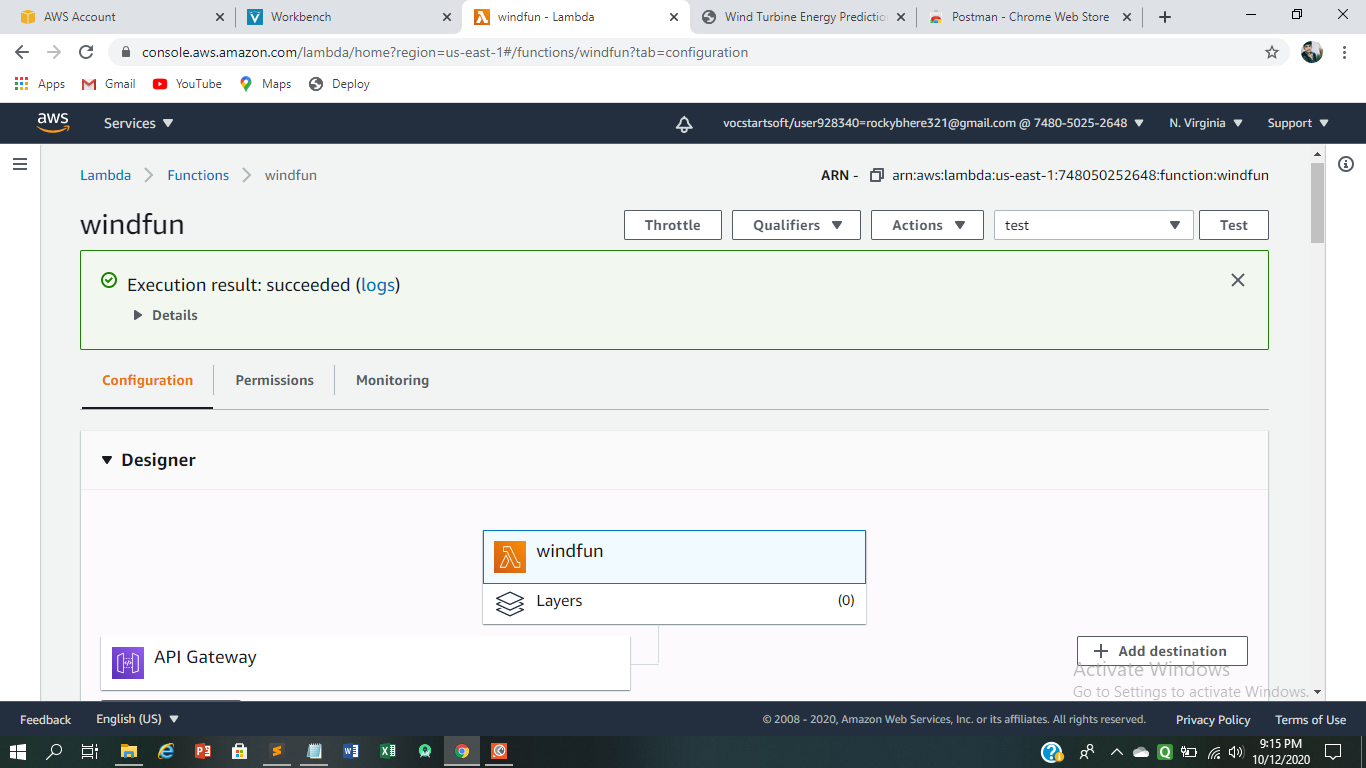
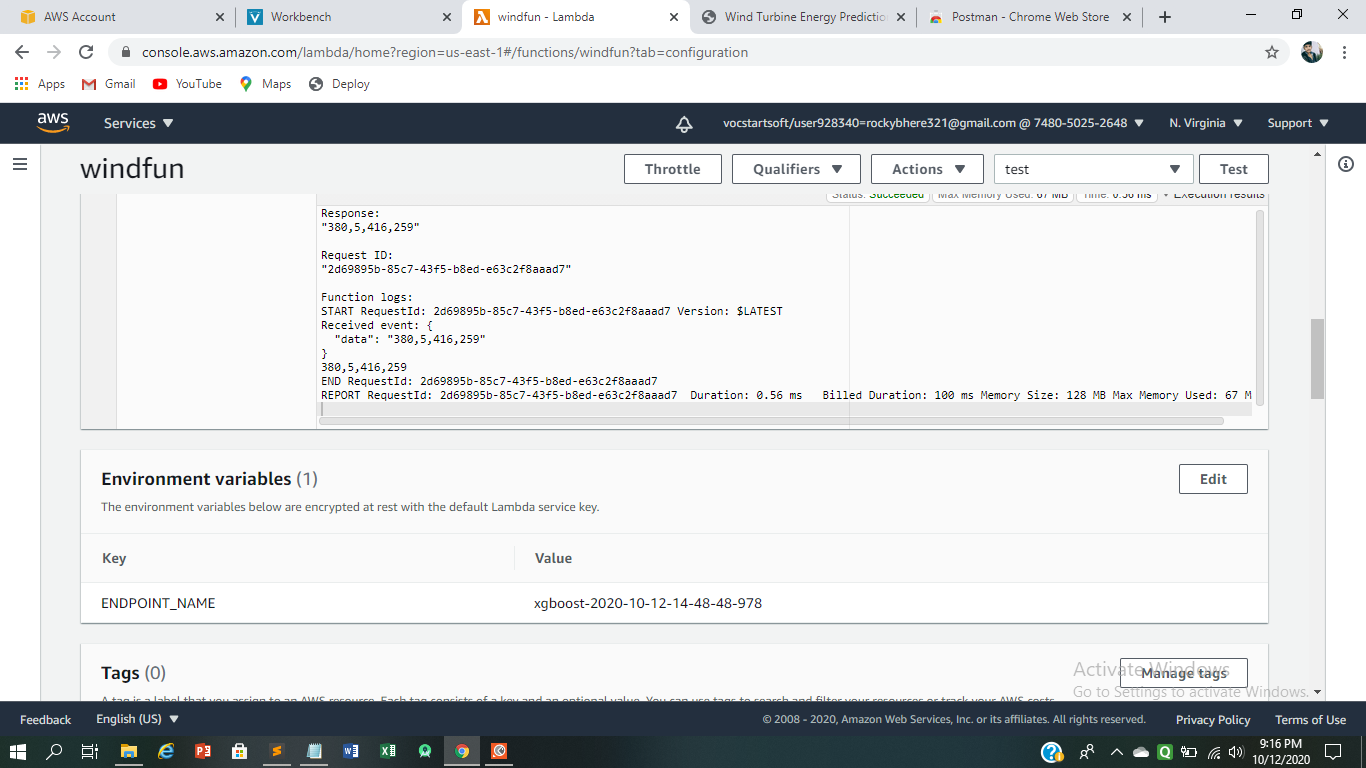
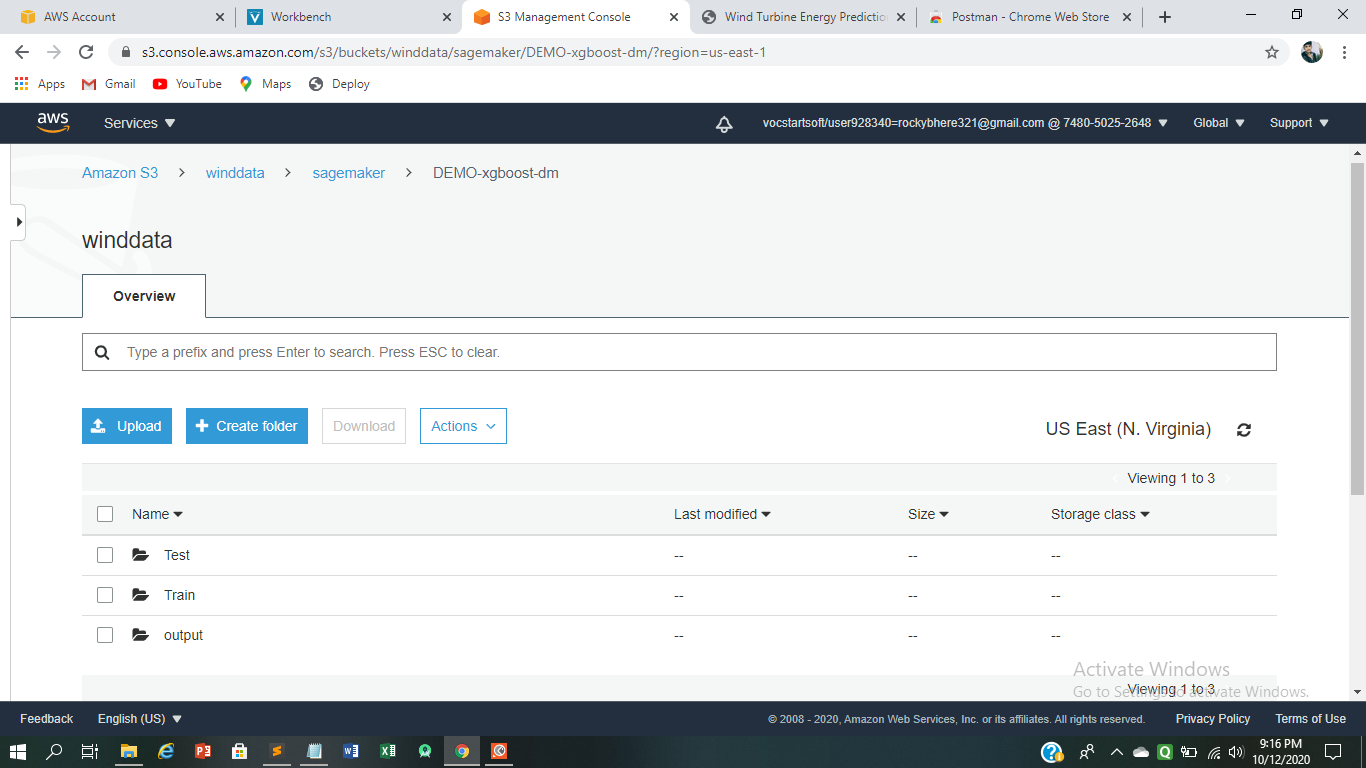
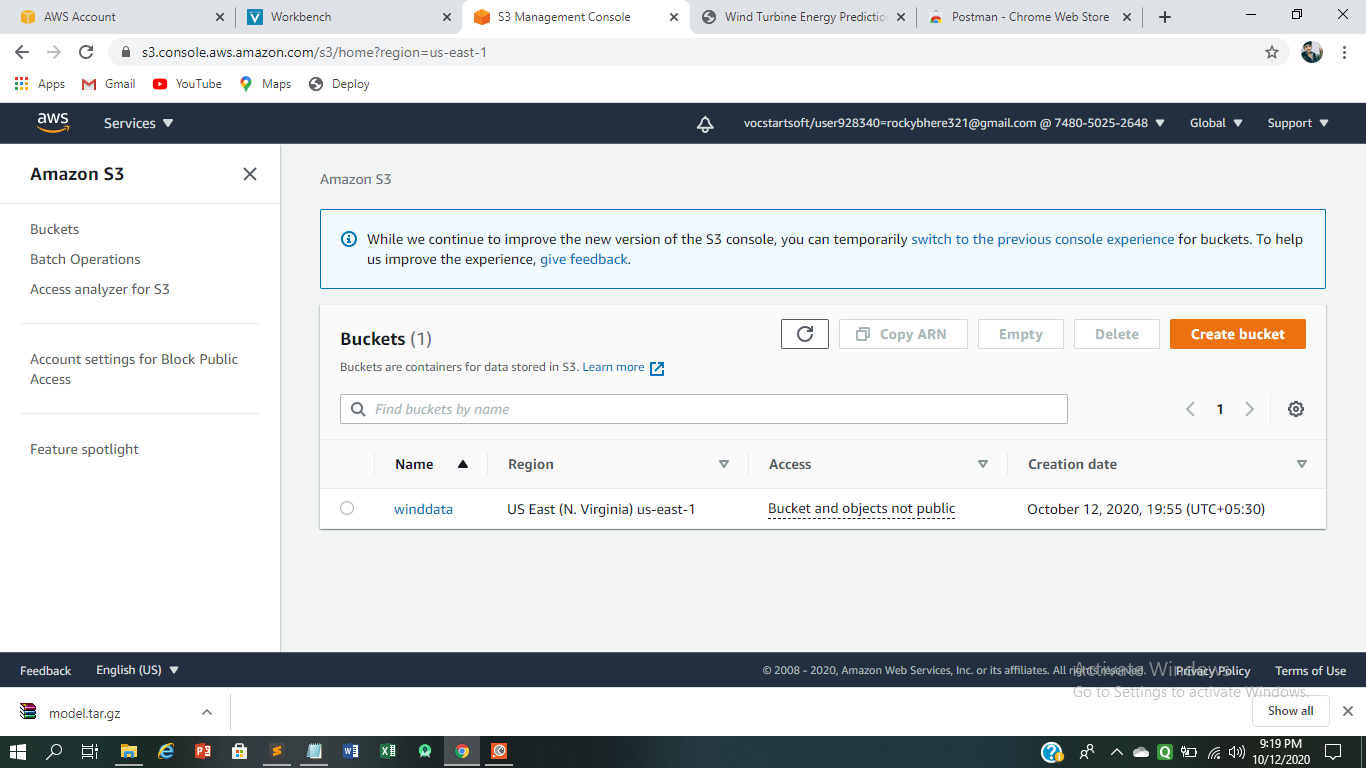
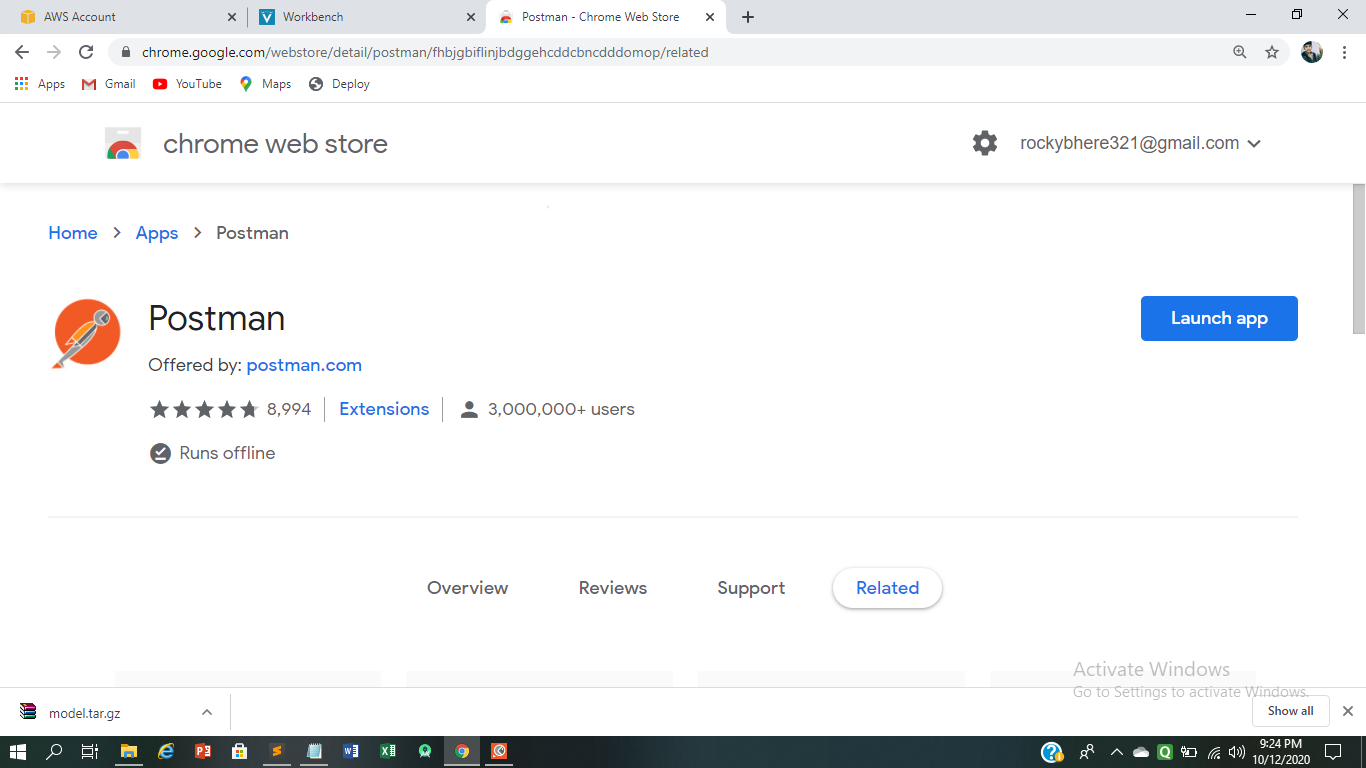
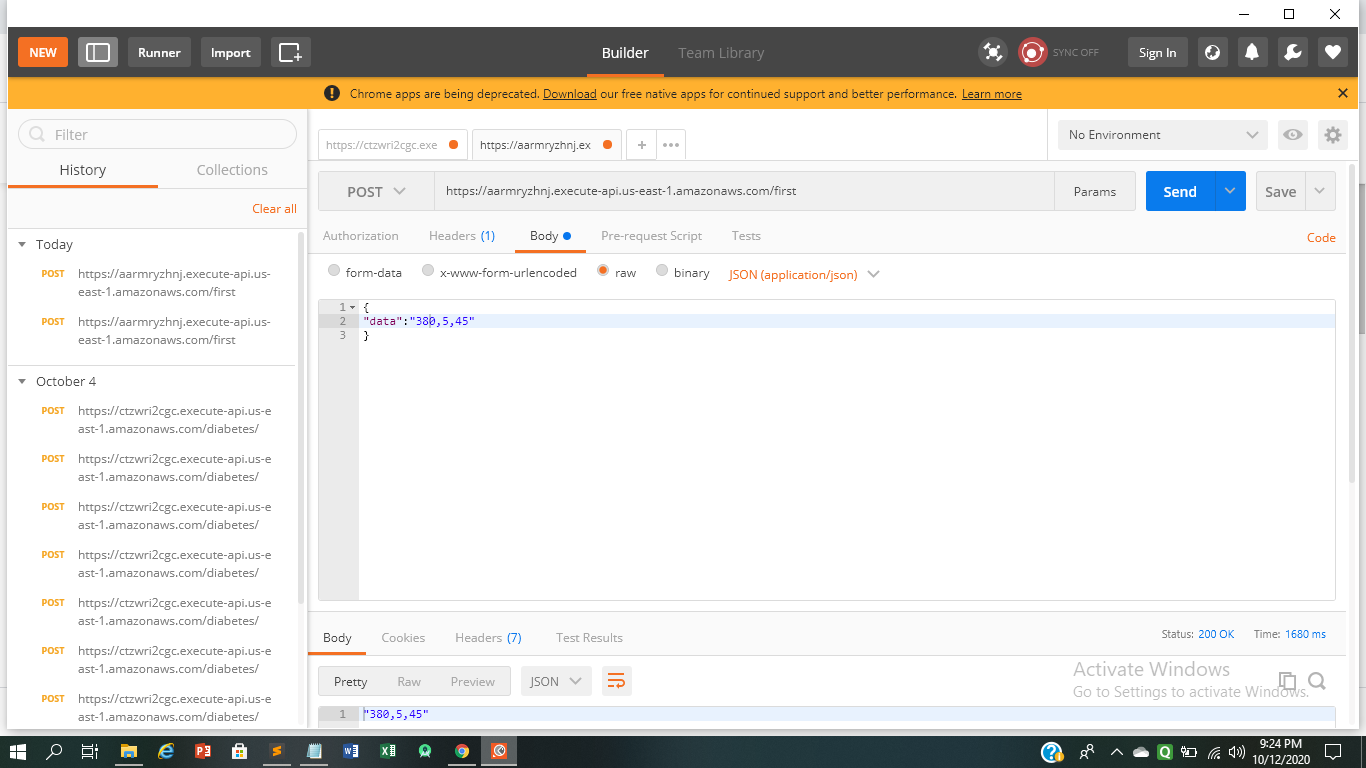
4. Creating API Gateway

5. Building Flask application

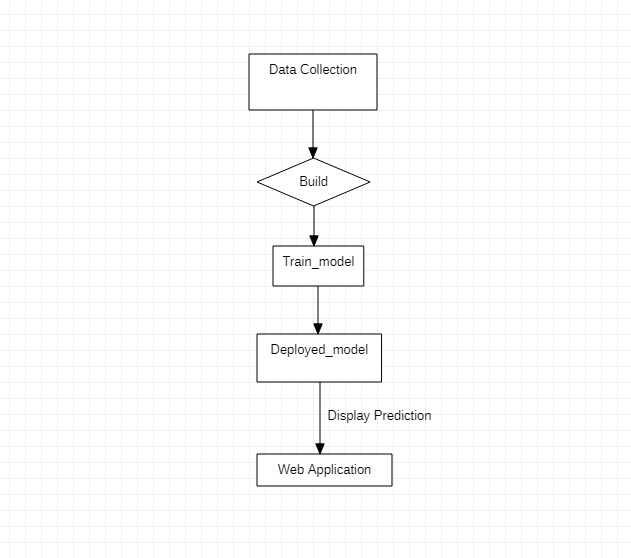
**Experimental Investigations:**

Develop an end-to-end web application that predicts the wind turbibed. The application must be built with Python-Flask or Django framework with the machine learning model trained & deployed on AWS Sagemaker. Create an API Endpoint for the model with the help of API Gateway and AWS Lambda Service.





## **Flowchart**



***Fig. 2.****Flowchart*

**Result:**

Develop an end-to-end web application that predicts the Wind Turbine based on Weather Conditions. The application must be built with Python-Flask with the machine learning model trained & deployed on AWS Sagemaker. Create an API Endpoint for the model with the help of API Gateway and AWS Lambda Service.

* We report on the correlation of the different variables for the energy output.
* Model gives a very reliable prediction of the energy output.
* We predict the energy output with accuracy up to 80%.

**Advantages & Disadvantages:**

**Advantage:**

* Easy to use and has a friendly user interface to work with.
* Reduces man power and cost efficient
* Faster Predictions on just a click of a button
* Intuitive User Interface

**Disadvantage:**

* Requires all services that handles requests and renders responses.
* Requires some complex integration of services

**Applications:**

* Several companies can use the service and deploy it,on their own servers
* This would save time and money as no three to four applications are needed
* The service can be provided to users in application along with other features
* Could be helpful even in areas with less connectivity.
* As the application is quite robust and resilient in its architecture, it allows one to easily navigate through different sections.

**Conclusion**

A Machine Learning has been developed to forecast the Power output of the wind turbine. A simple, efficient and a versatile model is built keeping in mind the diversity of data, computational complexity and overhead involved in making API calls to the model for prediction. The product can increase the accuracy of the forecasting the output from the wind turbine .Overall accurate wind power prediction reduces the financial and technical risk of uncertainty of wind power production for all electricity market participants.

**Future Scope:**

* A more generalized model can be developed to suit forecasting for different locations.
* The model can be developed to make predictions on other sources of data such as solar power, tidal power etc.
* On-device model can be developed to make much more faster predictions.

**Bibliography:**

### This video is a sample from Skillsoft's video course catalog. After watching this video, you will be able to get data into and out of an S3 bucket.

### <https://www.youtube.com/watch?v=VDdz1H18-0g>

### Data visualization using Python

<https://towardsdatascience.com/data-visualization-for-machine-learning-and-data-science-a45178970be7>

### how to build, train, and deploy a machine learning model with Amazon SageMaker? Learn how to build, train, and deploy a machine learning model with Amazon SageMaker.

<https://aws.amazon.com/getting-started/hands-on/build-train-deploy-machine-learning-model-sagemaker/>

### Build Model By Applying Training Data To Estimators And Deploy The Model

<https://aws.amazon.com/getting-started/hands-on/build-train-deploy-machine-> learning-model-sagemaker/

### You can run Python code in AWS Lambda. Lambda provides runtimes for Python that run your code to process events. Your code runs in an environment that includes the SDK for Python (Boto3), with credentials from an AWS Identity and Access Management (IAM) role that you manage.

<https://docs.aws.amazon.com/lambda/latest/dg/lambda-python.html>

### How to use AWS Lambda with Amazon API Gateway.

<https://docs.aws.amazon.com/lambda/latest/dg/services-apigateway.html>

**Appendix:**

**Source Code :**

**Predicting the Energy Output of Wind Turbine .ipynb**



**HTML CODE :**

<html>

<head>

<style>

@import url("https://fonts.googleapis.com/css?family=Fjalla+One&display=swap");

\* {

margin: 0;

padding: 0;

}

body {

background-color:black

no-repeat;

background-size: cover;

width: 100vw;

height: 100vh;

align-items: center;

justify-items: center;

}

.contact-us {

background: #f8f4e5;

padding: 50px 100px;

box-shadow: 15px 15px 1px #ffa580, 15px 15px 1px 2px black;

}

input {

display: block;

width: 100%;

font-size: 14pt;

line-height: 28pt;

font-family: "";

margin-bottom: 28pt;

border: none;

border-bottom: 5px solid black;

background: #f8f4e5;

min-width: 250px;

padding-left: 5px;

outline: none;

color: black;

}

input:focus {

border-bottom: green;

}

button {

display:block;

margin: 0 auto;

line-height: 28pt;

padding: 0 20px;

background: gray;

letter-spacing: 2px;

transition: 0.3s all ease-in-out;

outline: none;

}

button:hover {

background:green;

color:green;

border: 1px yellow;

}

::selection {

background:white;

}

input:-webkit-autofill,

input:-webkit-autofill:hover,

input:-webkit-autofill:focus {

}

</style>

<title>

Wind Turbine Energy Prediction

</title>

</head>

<body>

<div class="login-box">

<center><h2><b>Predicting the Energy Output of Wind Turbine Based on Weather Conditions </b></h2></center>

<form action="/" method="post">

<div class="contact-us">

<label>Wind Speed(m/s)</label>

<input type="number" step ="any" name="Wind Speed" required="required"/>

<br><br>

<label>Theoretical\_Power\_Curve (KWh)</label>

<input type="number" step="any" name="Theoretical\_Power" required="required"/>

<br>

<br>

<label>Wind Direction (°)</label>

<input type="number" step="any" name='Wind Direction'required="required"/>

<br><br>

<button type="button"><input type="submit" value="Predict"/></button>

<br>

<h1><b1>Final Prediction = </b>360.4564657<h1>

</form>

</form>

</div>

</body>

</html>

**Python Code :**

from flask import Flask,render\_template,redirect,url\_for,request

import requests

url = "https://aarmryzhnj.execute-api.us-east-1.amazonaws.com/first"

headers = {

'X-Amz-Content-Sha256': 'beaead3198f7da1e70d03ab969765e0821b24fc913697e929e726aeaebf0eba3',

'X-Amz-Date': '20201012T200352',

'Authorization': 'AWS4-HMAC-SHA256 Credential=ASIAXI5D2F6WWMCLBASR/20201010/us-east-1/execute-api/aws4\_request, SignedHeaders=host;x-amz-content-sha256;x-amz-date, Signature=c40374386a1e23b33a2fa7fb12075f0aa9f4ed32a5370a375c37daa025d6c9f3',

'Content-Type': 'text/plain'

}

s=""

app = Flask(\_\_name\_\_)

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

def home():

if request.method=='POST':

windspeed = str(float(request.form["Wind Speed"]))

tpower =str(float(request.form["Theoretical\_Power"]))

wind\_direction = str(float(request.form['Wind Direction']))

s=windspeed+','+tpower+','+wind\_direction

payload = "{\r\n \"data\" : \""+s+"\""+"\r\n}"

response = requests.request("POST", url, headers=headers, data = payload)

r=response.text.encode('utf8')

r=float()

return render\_template('htmlfile.html',content=r)

else:

return render\_template('htmlfile.html')

if \_\_name\_\_ == "\_\_main\_\_":

app.run(debug=True)

**Lambda Function :**

import os

import io

import boto3

import json

import csv

# grab environment variables

ENDPOINT\_NAME = "xgboost-2020-10-12-14-48-48-978"

runtime= boto3.client('runtime.sagemaker')

def lambda\_handler(event, context):

print("Received event: " + json.dumps(event, indent=2))

data = json.loads(json.dumps(event))

payload = data['data']

print(payload)

response = runtime.invoke\_endpoint(EndpointName=ENDPOINT\_NAME,

ContentType='text/csv',

Body=payload)

print(response)

result = json.loads(response['Body'].read().decode())

print(result)

return (result)

**FOR UI:**

