Step-by-Step Walkthrough for CS 643 Programming Assignment 2

Cloud Computing Energy Consumption Prediction with Apache Spark and Docker

Kurudunje Deekshith Shetty - KS2378

May 6, 2025

GitHub Repository: `https://github.com/KDShetty11/Energy-Consumption-Prediction-System-in-AWS-Cloud`

Docker Hub: `https://hub.docker.com/repository/docker/kdshetty/energypred`

Contents

1 Introduction 2

2 Setting Up the AWS Cloud Environment for Parallel Training 2
- 2.1 Creating an EMR Cluster . . . 2
- 2.2 Uploading Files to EMR Master Node . . . 3
- 2.3 Copying Files to HDFS . . . 4

3 Parallel Model Training on EMR Cluster 4
- 3.1 Launching Model Training . . . 4
- 3.2 Monitoring Training Job . . . 5
- 3.3 Saving and Downloading the Trained Model . . . 5

4 Prediction Application on a Single EC2 Instance 6
- 4.1 Launching a Single EC2 Instance . . . 6
- 4.2 Pre-Configuring the EC2 Instance . . . 7
- 4.3 Uploading Files to EC2 Instance . . . 8
- 4.4 Extracting the Model . . . 8
- 4.5 Running Prediction Without Docker . . . 8

5 Building and Deploying the Docker Container 9
- 5.1 Installing Docker . . . 9
- 5.2 Building the Docker Image . . . 9
- 5.3 Running the Docker Container . . . 10
- 5.4 Pushing to Docker Hub . . . 10
- 5.5 Verifying Docker Image . . . 11

6 Repository Links 11

7 Use of ChatGPT/AI Copilots 11

8 Notes 12

1 Introduction

This document provides a comprehensive guide for completing Programming Assignment 2 in CS 643, Cloud Computing. It details the setup of an AWS cloud environment, parallel training of an energy consumption prediction model using Apache Spark on an EMR cluster, development of a prediction application on a single EC2 instance, and deployment using Docker.

Disclaimer: Although most folders or filenames read as regress or regression, the model used for this assignment is gradient-boost and not regression, the term is just a placeholder that I overlooked.

2 Setting Up the AWS Cloud Environment for Parallel Training

2.1 Creating an EMR Cluster

Step: Launch an Amazon EMR cluster with four EC2 instances for parallel model training.

- Log in to the AWS Management Console and navigate to **EMR** > **Create Cluster**.
- Configure the cluster:
  - *Release*: `emr-7.8.0` (compatible with Spark 3.5.5)
  - *Applications*: Select **Spark Initiative bundle**
  - *Cluster Name*: `Energy Consumption Prediction Parallel Training`
  - *Cluster Configuration*: Uniform Instance
  - *bootstrap actions*: load the bootstrap.sh code

```
            sudo pip3 install numpy pandas
```

  Explanation: Ensures NumPy and Pandas are available for data processing.

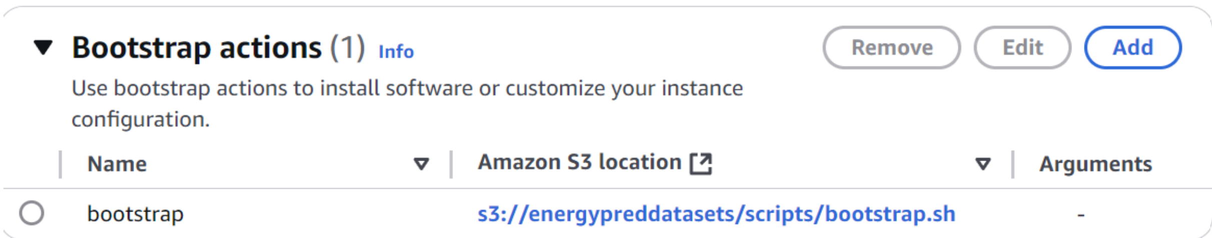


Figure 1: Python Dependency Bootstrap

- Hardware configuration:
  - *Instance Type(both primary and core*: `m5.xlarge`
  - *Number of Instances*: 4 (1 primary, 3 Core instances)
- Security: Select an EC2 key pair (vokey.ppk).
- IAM: Choose EMRDefault (add a rule to enable ssh).
- Click **Create Cluster**.

Explanation: This sets up a managed Spark cluster for distributed training on AWS EMR.

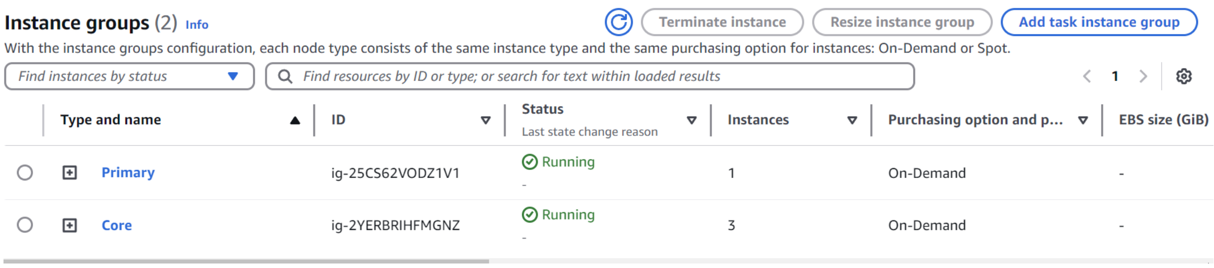


Figure 2: Screenshot of EMR Cluster Running

2.2 Uploading Files to EMR Master Node

Step: Transfer datasets and training script to the EMR master node using SFTP.

- Wait for the EMR cluster to reach the **Waiting** state.
- Copy the master node's public DNS (hadoop@ec2-3-83-174-73.compute-1.amazonaws.com).
- Open a terminal and start an SFTP session:

```
sftp -i labuser.pem hadoop@ec2-3-83-174-73.compute-1.amazonaws.com
```

- Upload files:

```
put TrainingDataset.csv
put train_model.py
```

Explanation: Transfers necessary files to the master node for storage in HDFS. Alternatively you can use winscp as shown below in the figure

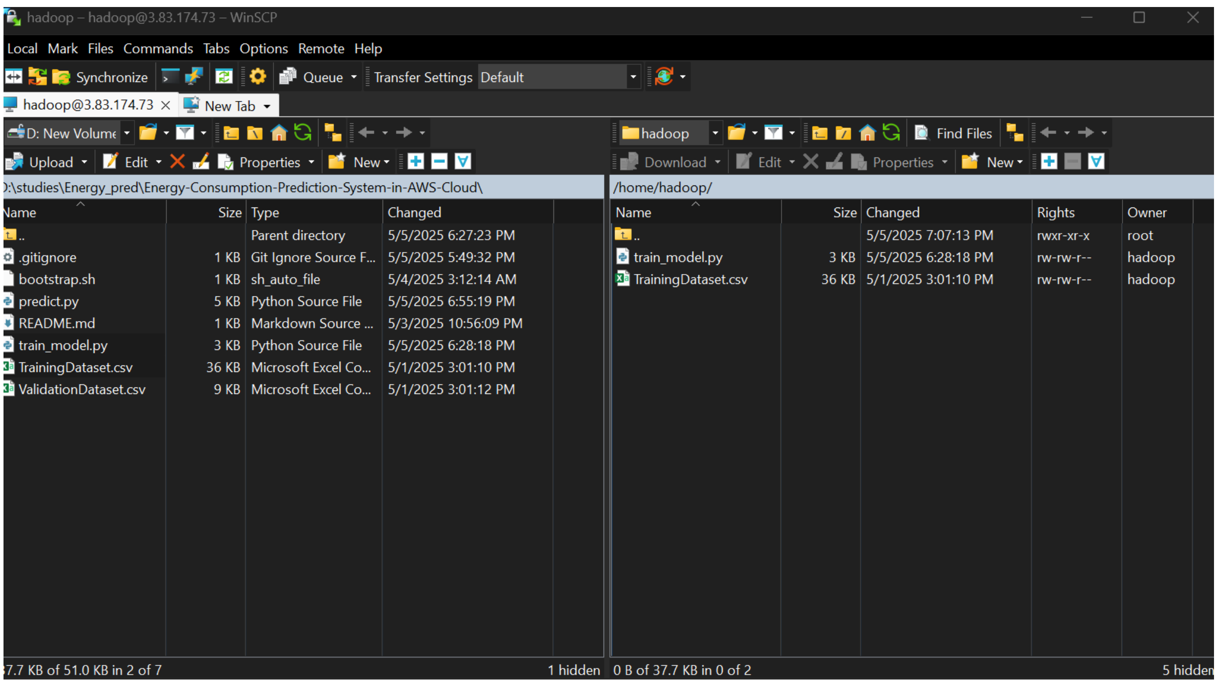


Figure 3: SFTP File Upload to EMR Master Node

2.3 Copying Files to HDFS

Step: Access the master node via SSH and move files to HDFS.

- SSH into the master node(Alternatively can use putty):

```
ssh -i labuser.pem hadoop@ec2-3-83-174-73.compute-1.amazonaws.com
```

- Copy files to HDFS:

```
hadoop fs -put TrainingDataset.csv /user/hadoop/TrainingDataset.csv
hadoop fs -put train_model.py /user/hadoop/train_model.py
```

- Verify files:

```
hdfs dfs -ls -t -R
```

Explanation: Stores datasets and script in HDFS for distributed access by Spark.

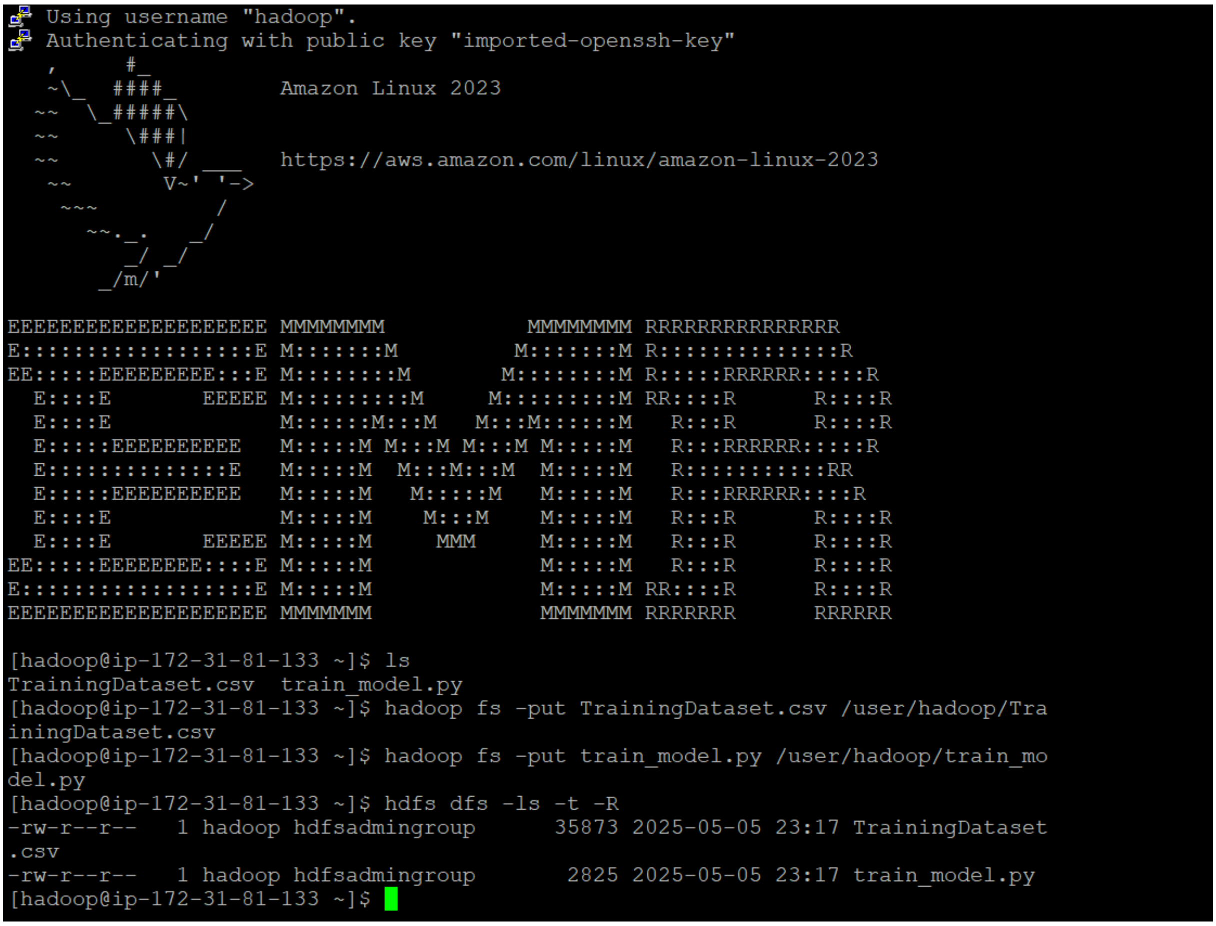


Figure 4: HDFS File Listing

3 Parallel Model Training on EMR Cluster

3.1 Launching Model Training

Step: Submit the training job to Spark.

```
spark-submit train_model.py
```

Explanation: Executes the train_model.py script, training an ML model (e.g., linear regression) using MLlib across four EC2 instances. The model is saved to HDFS in a folder (e.g., regression).

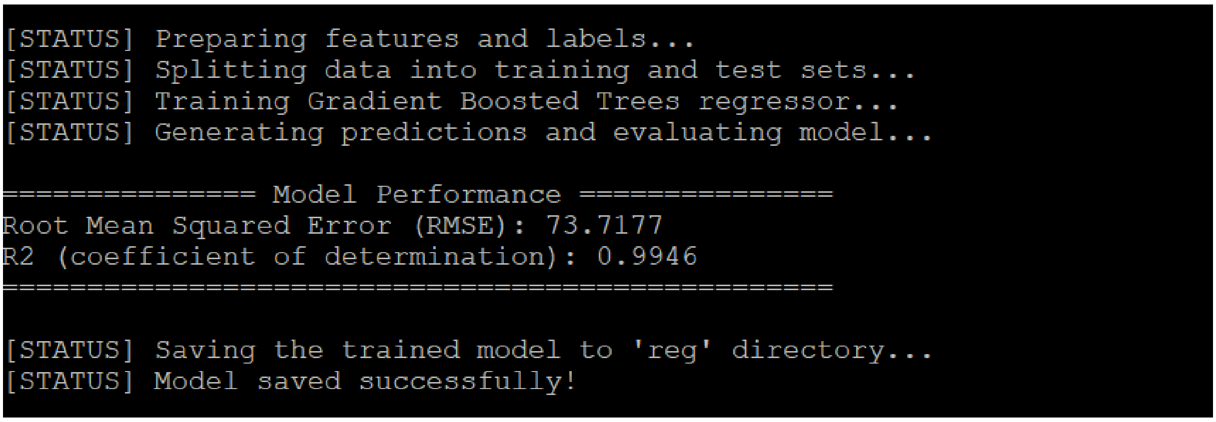


Figure 5: Spark-Submit Training Output

3.2 Monitoring Training Job

Step: Verify job execution via the Spark Web UI.

- Access the Spark Web UI through the EMR console's **Monitor** tab or at http://<master-node-dns>:808
- Confirm job completion.
- Alternatively we can use the below command to list the HDFS to verify if the trained model is saved.

```
hdfs dfs -ls -t -R
```

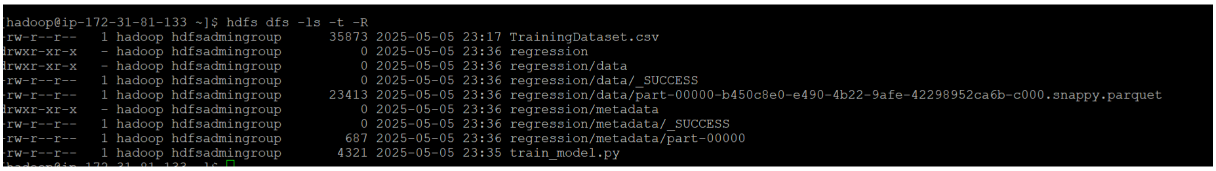


Figure 6: HDFS listing Trained Model

3.3 Saving and Downloading the Trained Model

Step: Copy and compress the trained model from HDFS.

```
hdfs dfs -copyToLocal regression /home/hadoop/regressionmod
tar -czf model.tar.gz regressionmod/
```

Explanation: Retrieves and compresses the model for transfer.

Step: Download the model to your local machine via SFTP or winscp.

```
get regressionmod/model.tar.gz
```

Explanation: Transfers the model for use in prediction.

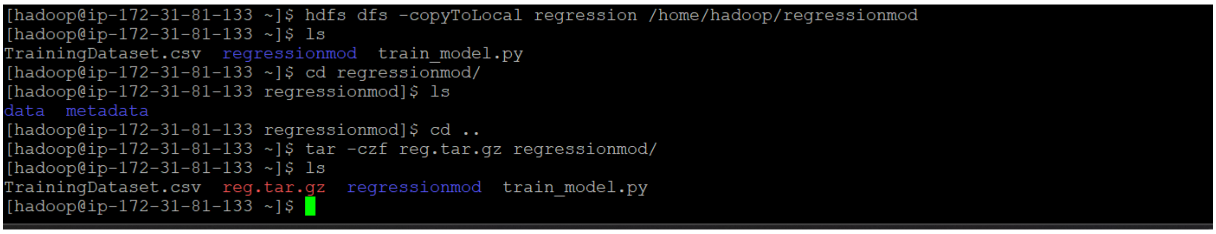


Figure 7: Model Copy and Compression

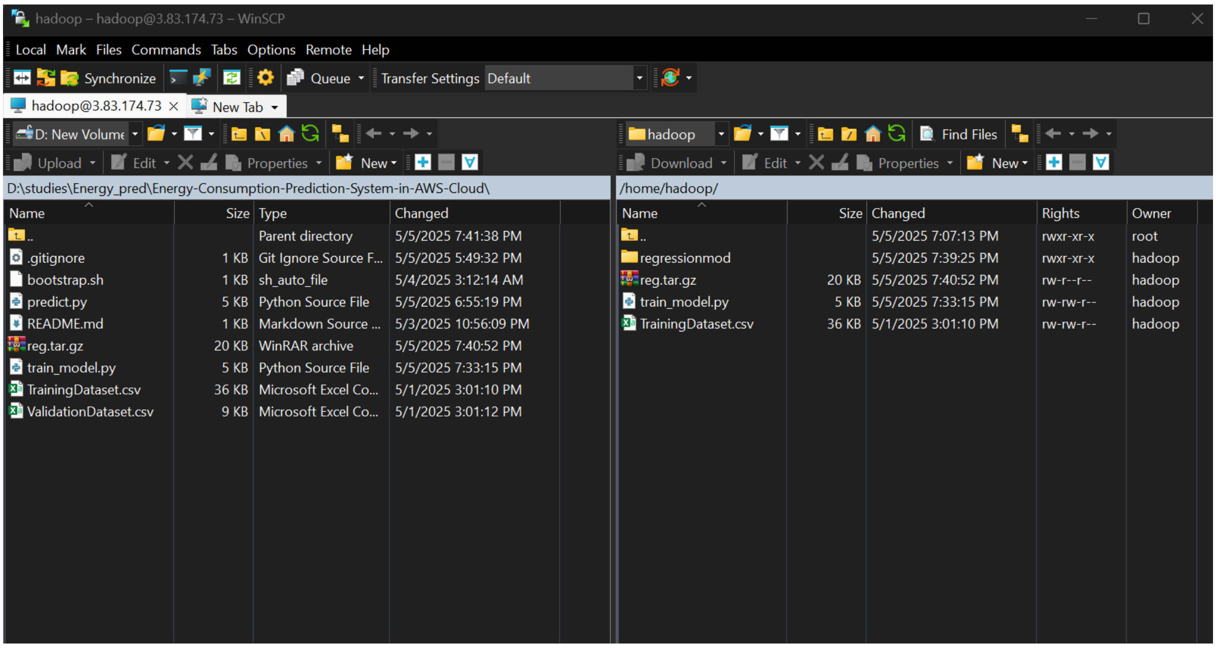


Figure 8: SFTP Model Download

4 Prediction Application on a Single EC2 Instance

4.1 Launching a Single EC2 Instance

Step: Create an EC2 instance for prediction.

- Navigate to **EC2** > **Launch Instance**.
- Select AMI: Ubuntu Server 24.04 LTS.
- Instance Type: `t2.medium`.
- Select a key pair (e.g., `vockey.pem`).
- Configure security group: Allow SSH (port 22).
- Launch the instance.

Explanation: Sets up a standalone EC2 instance for prediction.

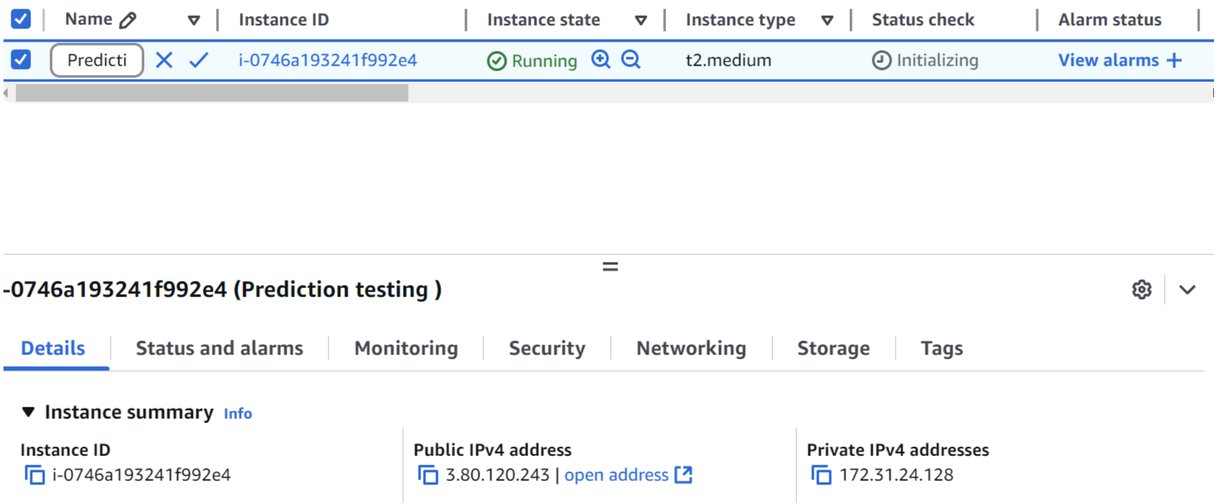


Figure 9: EC2 Instance Launch Screen

4.2 Pre-Configuring the EC2 Instance

Step: Install dependencies via SSH(can use putty).

```
ssh -i labuser.pem ubuntu@3.80.120.243
sudo apt-get update
sudo apt-get install -y python3-pip
sudo apt-get install -y python3-numpy
sudo apt-get install -y python3-pandas
sudo apt-get install -y openjdk-11-jdk
```

Explanation: Installs prerequisites for Spark.

Step: Install Apache Spark.

```
wget https://archive.apache.org/dist/spark/spark-3.5.5/spark-3.5.5-bin-hadoop3.tgz
sudo tar xvf spark-3.5.5-bin-hadoop3.tgz -C /opt
sudo chown -R ubuntu:ubuntu /opt/spark-3.5.5-bin-hadoop3
sudo ln -fs spark-3.5.5-bin-hadoop3 /opt/spark
```

Explanation: Configures Spark 3.5.5.

Step: Configure environment variables.

```
nano ~/.bash_profile
```

Add:

```
export SPARK_HOME=/opt/spark
PATH=$PATH:$SPARK_HOME/bin
export PATH
export JAVA_HOME=/usr/lib/jvm/java-1.11.0-openjdk-amd64
export PATH=$JAVA_HOME/bin:$PATH
```

Apply:

```
source ~/.bash_profile
```

Explanation: Sets up Spark and Java paths.

Step: Configure Spark logging.

```
cp $SPARK_HOME/conf/log4j2.properties.template $SPARK_HOME/conf/log4j2.
    properties
nano $SPARK_HOME/conf/log4j2.properties
```

Change `rootLogger.level = info` to `rootLogger.level = ERROR`.
Explanation: Reduces logging verbosity.

4.3 Uploading Files to EC2 Instance

Step: Upload files via SFTP.

```
sftp -i labuser.pem ubuntu@3.80.120.243
put predict.py
put ValidationDataset.csv
put model.tar.gz
```

Explanation: Transfers files for prediction. Alternatively can use winscp like the figure below.

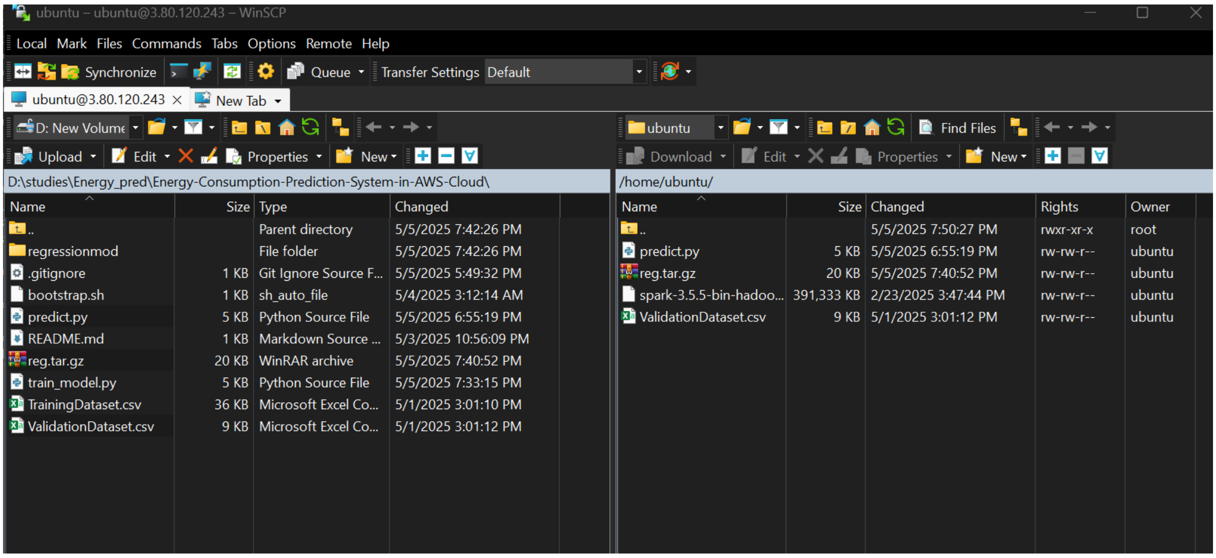


Figure 10: SFTP Upload to EC2

4.4 Extracting the Model

Step: Extract the model archive.

```
tar -xzvf model.tar.gz
```

Explanation: Uncompresses the model for prediction.

4.5 Running Prediction Without Docker

Step: Execute the prediction script.

```
spark-submit predict.py ValidationDataset.csv
```

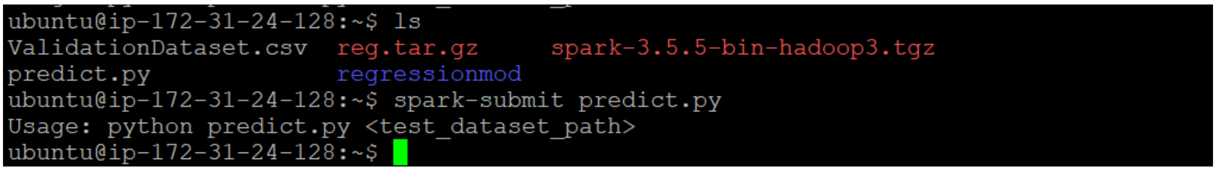


Figure 11: Model Extraction and Usecase

Explanation: Runs `predict.py`, which loads the model, predicts, and outputs RMSE.

Instructions for Running Without Docker:

1. Set up the EC2 instance (Sections 4.1–4.2).
2. Upload `predict.py`, `ValidationDataset.csv`, and `model.tar.gz`.
3. Extract the model: `tar -xzvf model.tar.gz`.
4. Run the command, replacing `ValidationDataset.csv` with the test file path if needed.

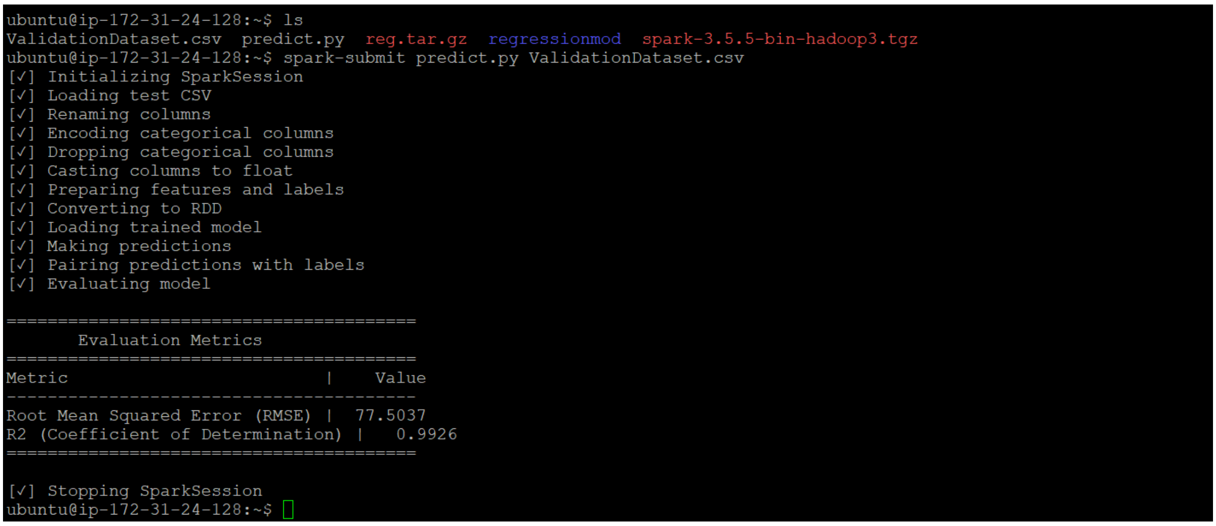


Figure 12: Prediction Output Without Docker

5 Building and Deploying the Docker Container

5.1 Installing Docker

Step: Install Docker on the EC2 instance.

```
sudo apt-get install docker.io
```

Explanation: Enables Docker container operations.

5.2 Building the Docker Image

Step: Build the Docker image.

```
sudo docker build -t energypred .
```

Explanation: Creates an image named `energypred` using the `Dockerfile`.

5.3 Running the Docker Container

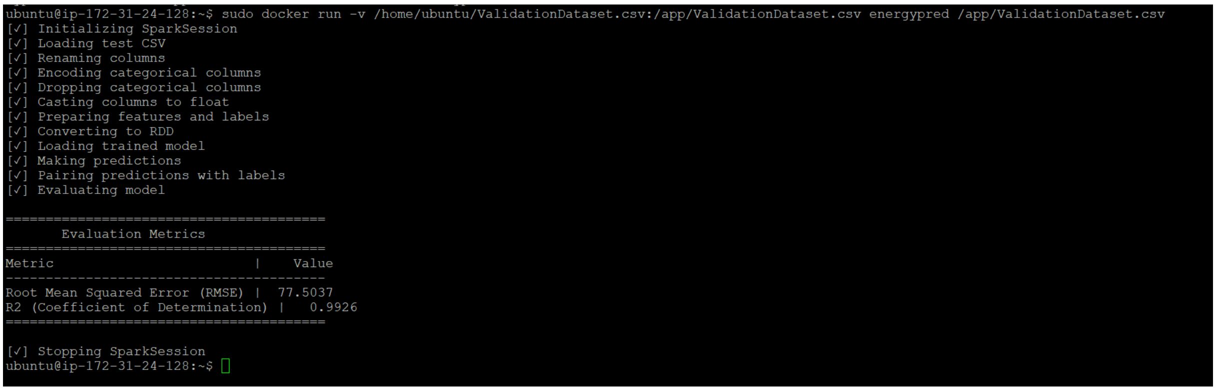
Step: Run the prediction application in a container.

```
sudo docker run -v /home/ubuntu/ValidationDataset.csv:/app/
   ValidationDataset.csv energypred /app/ValidationDataset.csv
```

Explanation: Maps the dataset to the container and runs the prediction.

Instructions for Running With Docker:

1. Install Docker (Section 5.1).
2. Pull the image: `docker pull kdshetty/energypred`.
3. Run the command, replacing paths as needed.

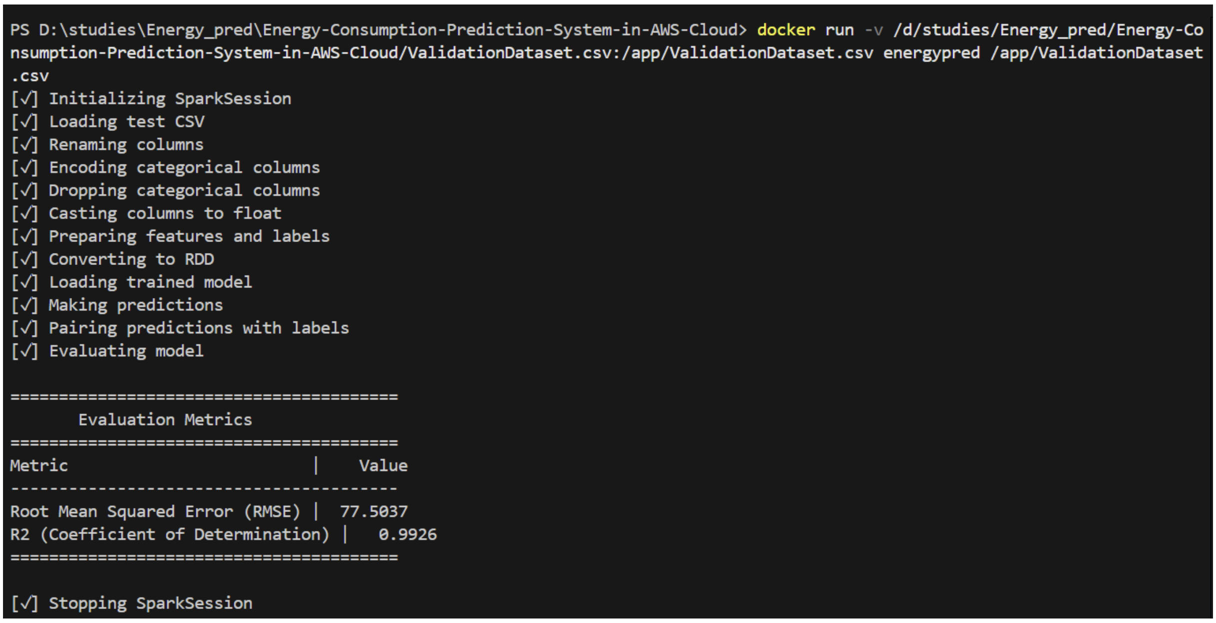


```
ubuntu@ip-172-31-24-128:~$ sudo docker run -v /home/ubuntu/ValidationDataset.csv:/app/ValidationDataset.csv energypred /app/ValidationDataset.csv
[✓] Initializing SparkSession
[✓] Loading test CSV
[✓] Renaming columns
[✓] Encoding categorical columns
[✓] Dropping categorical columns
[✓] Casting columns to float
[✓] Preparing features and labels
[✓] Converting to RDD
[✓] Loading trained model
[✓] Making predictions
[✓] Pairing predictions with labels
[✓] Evaluating model

========================================
       Evaluation Metrics
========================================
Metric                          |    Value
----------------------------------------
Root Mean Squared Error (RMSE) |  77.5037
R2 (Coefficient of Determination) |   0.9926
========================================

[✓] Stopping SparkSession
ubuntu@ip-172-31-24-128:~$
```

Figure 13: Docker Container Output



```
PS D:\studies\Energy_pred\Energy-Consumption-Prediction-System-in-AWS-Cloud> docker run -v /d/studies/Energy_pred/Energy-Consumption-Prediction-System-in-AWS-Cloud/ValidationDataset.csv:/app/ValidationDataset.csv energypred /app/ValidationDataset.csv
[✓] Initializing SparkSession
[✓] Loading test CSV
[✓] Renaming columns
[✓] Encoding categorical columns
[✓] Dropping categorical columns
[✓] Casting columns to float
[✓] Preparing features and labels
[✓] Converting to RDD
[✓] Loading trained model
[✓] Making predictions
[✓] Pairing predictions with labels
[✓] Evaluating model

========================================
       Evaluation Metrics
========================================
Metric                          |    Value
----------------------------------------
Root Mean Squared Error (RMSE) |  77.5037
R2 (Coefficient of Determination) |   0.9926
========================================

[✓] Stopping SparkSession
```

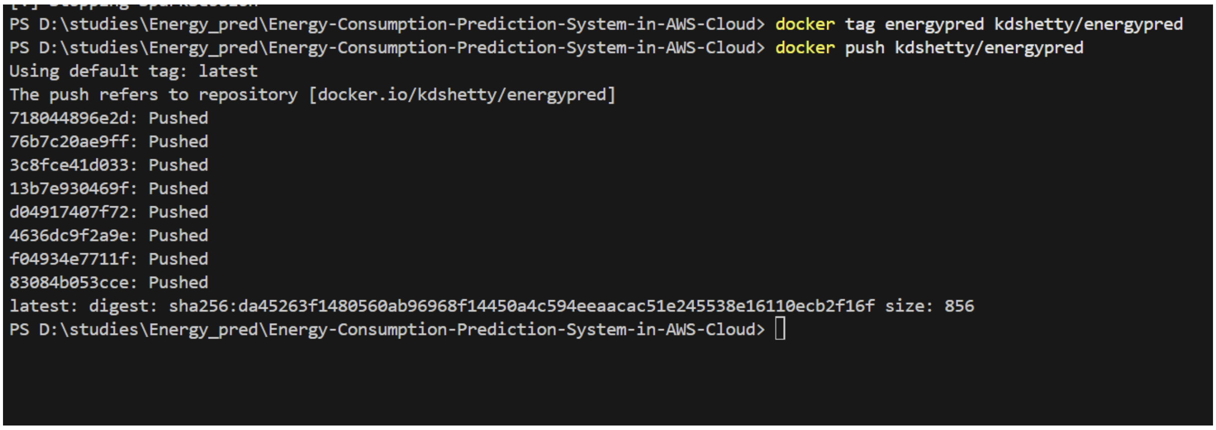
Figure 14: Docker Container Output on VSCode(Alternative)

5.4 Pushing to Docker Hub

Step: Tag and upload the image.

```
docker tag energypred kdshetty/energypred
docker push kdshetty/energypred
```

Explanation: Makes the image publicly accessible.



```
PS D:\studies\Energy_pred\Energy-Consumption-Prediction-System-in-AWS-Cloud> docker tag energypred kdshetty/energypred
PS D:\studies\Energy_pred\Energy-Consumption-Prediction-System-in-AWS-Cloud> docker push kdshetty/energypred
Using default tag: latest
The push refers to repository [docker.io/kdshetty/energypred]
718044896e2d: Pushed
76b7c20ae9ff: Pushed
3c8fce41d033: Pushed
13b7e930469f: Pushed
d04917407f72: Pushed
4636dc9f2a9e: Pushed
f04934e7711f: Pushed
83084b053cce: Pushed
latest: digest: sha256:da45263f1480560ab96968f14450a4c594eeaacac51e245538e16110ecb2f16f size: 856
PS D:\studies\Energy_pred\Energy-Consumption-Prediction-System-in-AWS-Cloud>
```

Figure 15: Docker Push to Docker Hub(Alternative on VScode)

5.5 Verifying Docker Image

Step: Pull the image to confirm availability.

```
docker pull kdshetty/energypred
```

Explanation: Ensures the image is deployable.

```
Last login: Tue May  6 03:25:34 2025 from 108.53.25.28
ubuntu@ip-172-31-24-128:~$ docker pull kdshetty/energypred
Using default tag: latest
latest: Pulling from kdshetty/energypred
13b7e930469f: Already exists
83084b053cce: Pull complete
718044896e2d: Pull complete
d04917407f72: Pull complete
4636dc9f2a9e: Pull complete
f04934e7711f: Pull complete
3c8fce41d033: Pull complete
Digest: sha256:da45263f1480560ab96968f14450a4c594eeaacac51e245538e16110ecb2f16f
Status: Downloaded newer image for kdshetty/energypred:latest
docker.io/kdshetty/energypred:latest
ubuntu@ip-172-31-24-128:~$
```

Figure 16: Docker Image Pull

---

6 Repository Links

- **GitHub Repository**: https://github.com/KDShetty11/Energy-Consumption-Prediction-Sys
- **Docker Hub Repository**: https://hub.docker.com/repository/docker/kdshetty/energypred

7 Use of ChatGPT/AI Copilots

Code Generated by ChatGPT:

- Initial `train_model.py` structure (Spark DataFrame and MLlib Gradient Boot).
- Partial `Dockerfile` (base image and dependencies).

Code Written from Scratch:

- Parameter tuning in `train_model.py` for RMSE optimization.
- `predict.py` for model loading and prediction.

Code Adapted from ChatGPT:

- Modified MLlib code for dataset-specific columns and RMSE calculation.
- Adjusted `Dockerfile` for Spark and model inclusion.

Experience with ChatGPT:

- *Usefulness*: Accelerated setup with Spark and Docker templates. MLlib examples were mostly accurate.
- *Limitations*: Generated code used outdated APIs or incorrect paths, requiring debugging. Parameter tuning advice was generic.
- *Overall*: Effective for boilerplate but required expertise to adapt.

8 Notes

- Ensure sufficient storage and permissions for EMR and EC2. Advisable to use t2.medium or better for the EC2 Instance.
- Test prediction with `ValidationDataset.csv` to verify RMSE.
- Validate Docker container on a fresh EC2 instance.