CS 643: Cloud Computing (UCID: sp3254)

Programming Assignment 2: Wine Quality Prediction Model using Apache Spark and AWS

Github link: https://github.com/shivakarthik09/CS-643-Cloud-Computing-wineprediction-
Docker file: https://hub.docker.com/repository/docker/sp3254/wineprediction/general

Objective

This assignment focuses on developing parallel machine learning applications on the Amazon AWS cloud platform. The primary goal is to learn:

1. Utilizing Apache Spark to train a machine learning model in parallel across multiple EC2 instances.
2. Employing Spark's MLlib for machine learning model development and application in the cloud.
3. Leveraging Docker to containerize the ML model, simplifying deployment across environments.

Introduction

The task involves building and deploying a wine quality prediction model. This model will be trained using Spark on AWS, utilizing multiple EC2 instances for parallel processing. The implementation language will be Java on an Ubuntu Linux environment.

Methodology

Dataset Description

- **TrainingDataset.csv**: Used for training the model on multiple EC2 instances in parallel.
- **ValidationDataset.csv**: Employed for model validation and parameter tuning to optimize performance.

Model Development

- The model is trained using Spark's MLlib, starting with a basic logistic regression model, potentially exploring other models for enhanced performance.
- The model uses the training dataset for learning and the validation dataset for performance tuning.
- Classification approach considers wine scores from 1 to 10, allowing a multi-class classification model.

Instance Initial setup commands:

Download Spark
wget https://downloads.apache.org/spark/spark-3.5.3/spark-3.5.3-bin-hadoop3.tgz

Extract Spark
tar -xvzf spark-3.5.3-bin-hadoop3.tgz

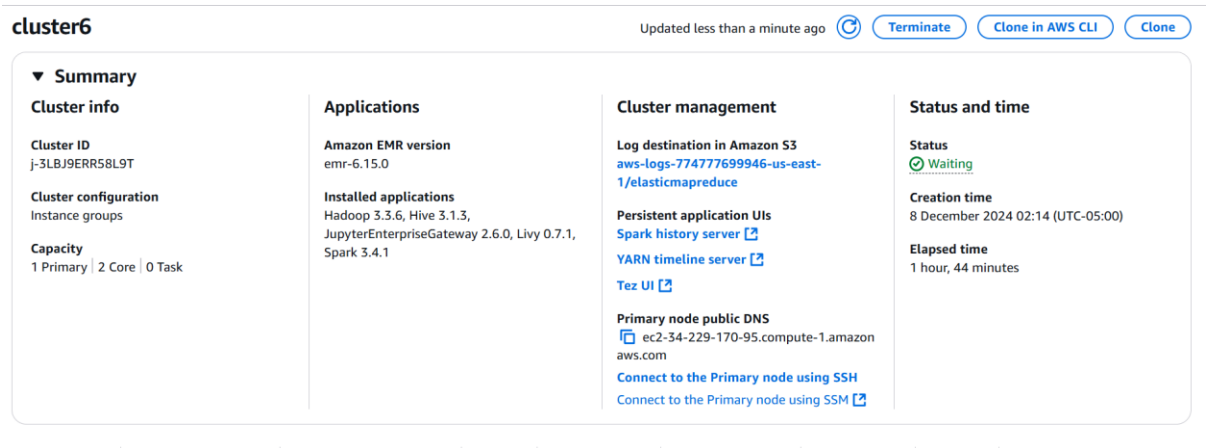
Set Up Hadoop-AWS Dependencies
wget -P /usr/local/spark/jars/ https://repo1.maven.org/maven2/org/apache/hadoop/hadoop-aws/3.3.1/hadoop-aws-3.3.1.jar
wget -P /usr/local/spark/jars/ https://repo1.maven.org/maven2/com/amazonaws/aws-java-sdk-bundle/1.11.375/aws-java-sdk-bundle-1.11.375.jar

Create Core-Site Configuration for S3 Access
nano /usr/local/spark/conf/core-site.xml

Parallel Training Implementation process:

Setting Up an EMR Cluster

1. **Initiate Cluster Creation**: Begin by accessing the Amazon EMR service. Click on "Create Cluster" to start the setup process.
2. **Configure Cluster Details**:
    - **Launch Mode**: Ensure you select "Cluster" as the launch mode.
    - **Instance Details**: Choose four instances as required for parallel processing. You'll also need to select or create an EC2 key pair. This key will be used to securely access the master node.
3. **Set Roles and Security**:
    - **Roles**: Select the default roles for service and instance to ensure proper permissions for your cluster operations.
    - **Security**: Navigate to the security settings and choose or generate a .pem file to secure communications to your cluster.
4. **Launch and Monitor the Cluster**:
    - Launch the cluster and monitor its status. Wait for it to transition from "starting" to "waiting," which indicates it is ready to receive jobs.



Preparing Your Application

5. **Package Your Java Code**:
    - Use Maven or Eclipse to package your Java application into a JAR file. This file contains all the necessary code and dependencies to run your model training.

Clean and Compile the Project
mvn clean package
Run the Project Locally
mvn exec:java -Dexec.mainClass="com.wqp.spark.WineQualityPrediction"

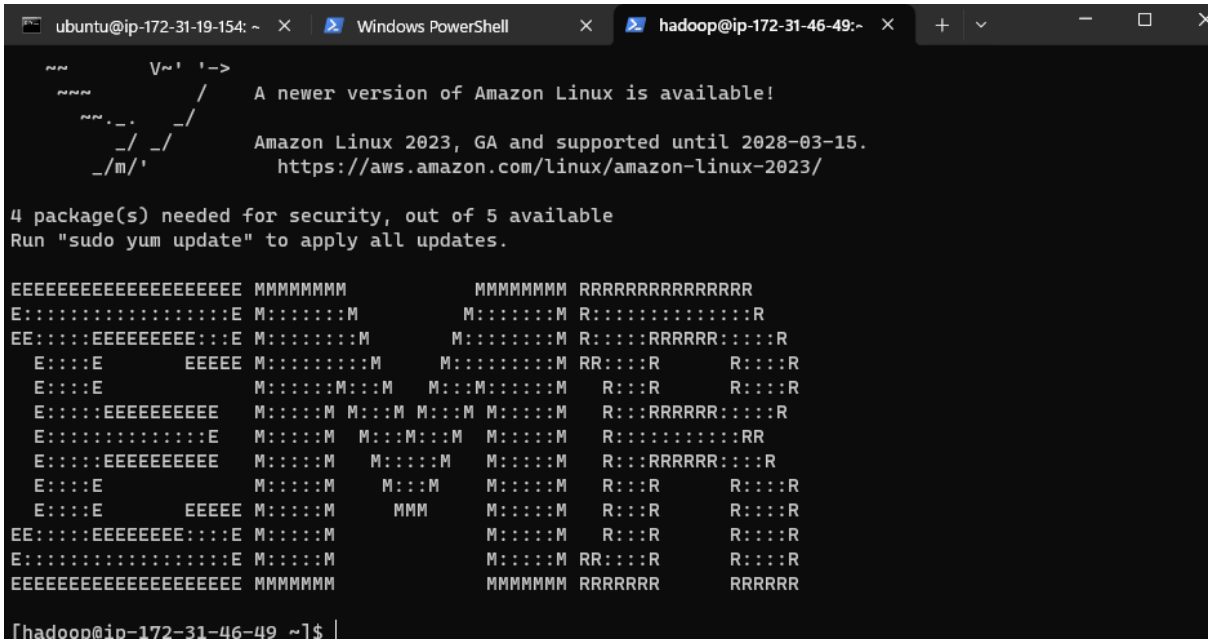
```
Results :

Tests run: 1, Failures: 0, Errors: 0, Skipped: 0

[INFO]
[INFO] --- maven-jar-plugin:2.4:jar (default-jar) @ WineQualityPredictor ---
[INFO] Building jar: /home/ubuntu/WineQualityPredictor/target/WineQualityPre
dictor-1.0-SNAPSHOT.jar
[INFO] ------------------------------------------------------------------------
---
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
---
[INFO] Total time:  5.928 s
[INFO] Finished at: 2024-12-08T22:13:32Z
[INFO] ------------------------------------------------------------------------
---
```

Deploying and Running the Model

6. **Submit Spark Job**:
    - **Access the Cluster**: Copy the DNS name of the cluster from the AWS console.
    - **Modify Security Groups**: Update the security groups for the master node to allow SSH connections on port 22 from your IP address.
    - **Run the Job**: Use the spark-submit command to run your model.jar file on the cluster. This step will execute your training model across the multiple instances.



Parallel Implemantaion using yarn on all clusters:

cmd: spark-submit --class com.wqp.spark.App --master yarn s3://s3bucket9542/WineQualityPredictor-1.0-SNAPSHOT.jar

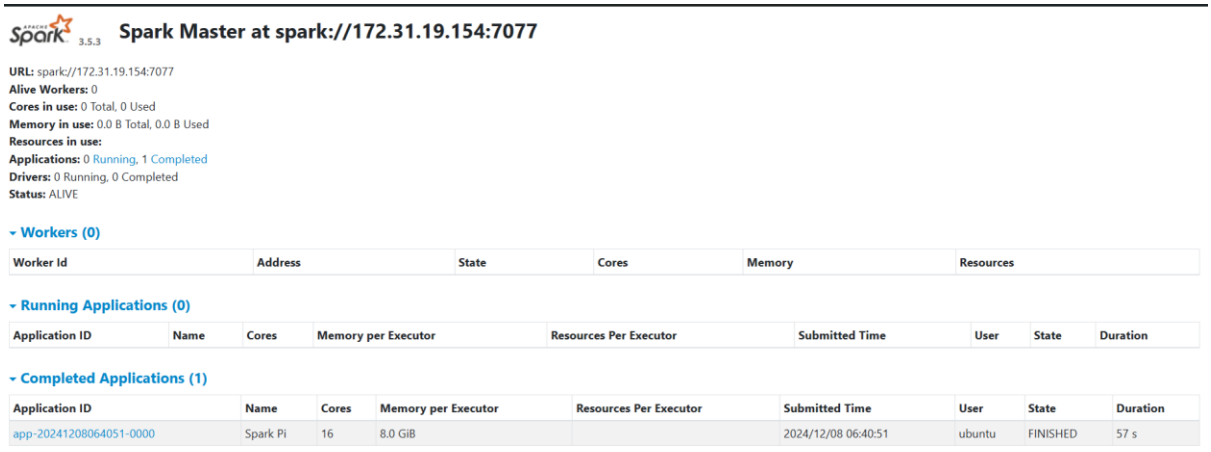
Running on Training Dataset

```
24/12/08 21:17:43 INFO NettyBlockTransferService: Server created on ip-172-3
1-45-61.ec2.internal:33401
24/12/08 21:17:43 INFO BlockManager: Using org.apache.spark.storage.RandomBl
ockReplicationPolicy for block replication policy
24/12/08 21:17:43 INFO BlockManager: external shuffle service port = 7337
24/12/08 21:17:43 INFO BlockManagerMaster: Registering BlockManager BlockMan
agerId(driver, ip-172-31-45-61.ec2.internal, 33401, None)
24/12/08 21:17:43 INFO BlockManagerMasterEndpoint: Registering block manager
 ip-172-31-45-61.ec2.internal:33401 with 1048.8 MiB RAM, BlockManagerId(driv
er, ip-172-31-45-61.ec2.internal, 33401, None)
24/12/08 21:17:43 INFO BlockManagerMaster: Registered BlockManager BlockMana
gerId(driver, ip-172-31-45-61.ec2.internal, 33401, None)
24/12/08 21:17:43 INFO BlockManager: Initialized BlockManager: BlockManagerI
d(driver, ip-172-31-45-61.ec2.internal, 33401, None)
24/12/08 21:17:44 INFO SingleEventLogFileWriter: Logging events to hdfs:/var
/log/spark/apps/local-1733692663209.inprogress
Model Accuracy: 0.8968253968253969
[hadoop@ip-172-31-45-61 ~]$
```

Validation Data:

```
24/12/08 21:17:43 INFO NettyBlockTransferService: Server created on ip-172-3
1-45-61.ec2.internal:33401
24/12/08 21:17:43 INFO BlockManager: Using org.apache.spark.storage.RandomBl
ockReplicationPolicy for block replication policy
24/12/08 21:17:43 INFO BlockManager: external shuffle service port = 7337
24/12/08 21:17:43 INFO BlockManagerMaster: Registering BlockManager BlockMan
agerId(driver, ip-172-31-45-61.ec2.internal, 33401, None)
24/12/08 21:17:43 INFO BlockManagerMasterEndpoint: Registering block manager
 ip-172-31-45-61.ec2.internal:33401 with 1048.8 MiB RAM, BlockManagerId(driv
er, ip-172-31-45-61.ec2.internal, 33401, None)
24/12/08 21:17:43 INFO BlockManagerMaster: Registered BlockManager BlockMana
gerId(driver, ip-172-31-45-61.ec2.internal, 33401, None)
24/12/08 21:17:43 INFO BlockManager: Initialized BlockManager: BlockManagerI
d(driver, ip-172-31-45-61.ec2.internal, 33401, None)
24/12/08 21:17:44 INFO SingleEventLogFileWriter: Logging events to hdfs:/var
/log/spark/apps/local-1733692663209.inprogress
Model Accuracy: 0.8968253968253969
[hadoop@ip-172-31-45-61 ~]$
```

Single Implementation using master [local*] on a single instance using maven:



Spark Master at spark://172.31.19.154:7077

URL: spark://172.31.19.154:7077
Alive Workers: 0
Cores in use: 0 Total, 0 Used
Memory in use: 0.0 B Total, 0.0 B Used
Resources in use:
Applications: 0 Running, 1 Completed
Drivers: 0 Running, 0 Completed
Status: ALIVE

Workers (0)

Worker Id	Address	State	Cores	Memory	Resources

Running Applications (0)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration

Completed Applications (1)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
app-20241208064051-0000	Spark Pi	16	8.0 GiB		2024/12/08 06:40:51	ubuntu	FINISHED	57 s

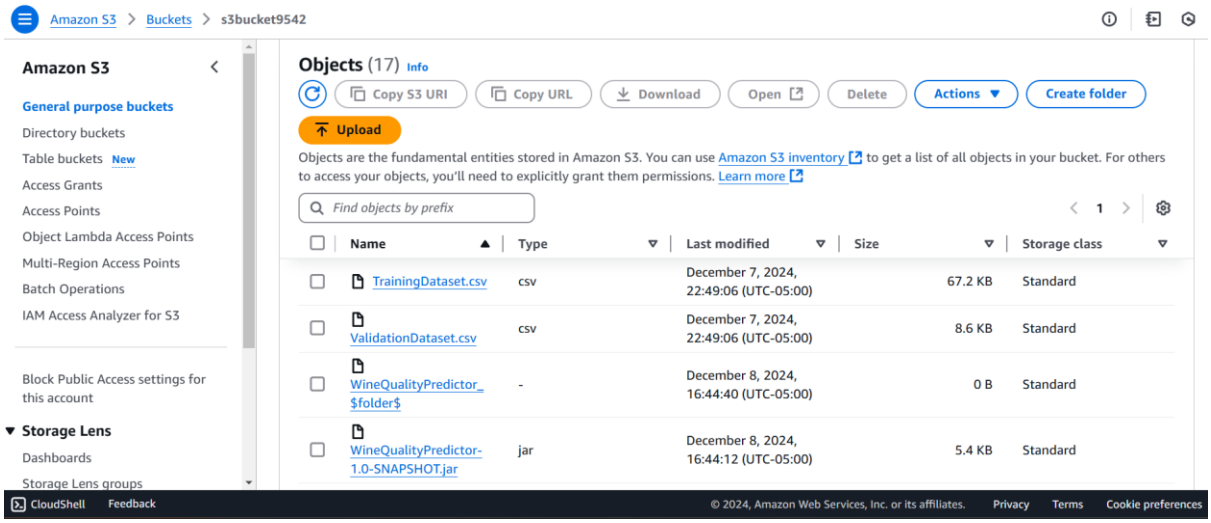
```
24/12/08 21:44:28 INFO BlockManager: external shuffle service port = 7337
24/12/08 21:44:28 INFO BlockManagerMaster: Registering BlockManager BlockMan
agerId(driver, ip-172-31-45-61.ec2.internal, 35193, None)
24/12/08 21:44:28 INFO BlockManagerMasterEndpoint: Registering block manager
 ip-172-31-45-61.ec2.internal:35193 with 1048.8 MiB RAM, BlockManagerId(driv
er, ip-172-31-45-61.ec2.internal, 35193, None)
24/12/08 21:44:28 INFO BlockManagerMaster: Registered BlockManager BlockMana
gerId(driver, ip-172-31-45-61.ec2.internal, 35193, None)
24/12/08 21:44:28 INFO BlockManager: Initialized BlockManager: BlockManagerI
d(driver, ip-172-31-45-61.ec2.internal, 35193, None)
24/12/08 21:44:29 INFO SingleEventLogFileWriter: Logging events to hdfs:/var
/log/spark/apps/local-1733694268727.inprogress
Test Error = 0.16249999999999998
Accuracy = 0.8375
F1 = 0.7634353741496599
Precision = 0.70140625
Recall = 0.8375
[hadoop@ip-172-31-45-61 ~]$
```

S3 Bucket Content:

Create an S3 Bucket for Datasets
aws s3 mb s3://s3bucket9542

Upload Datasets to S3
aws s3 cp TrainingDataset.csv s3://s3bucket9542/
aws s3 cp ValidationDataset.csv s3://s3bucket9542/

Fetch Logs from Spark Jobs
aws s3 cp s3://s3bucket9542/logs/spark-output.log




```
ubuntu@ip-172-31-19-154:~$ sudo ./aws/install
You can now run: /usr/local/bin/aws --version
ubuntu@ip-172-31-19-154:~$ aws --version
aws-cli/2.22.12 Python/3.12.6 Linux/6.8.0-1019-aws exe/x86_64.ubuntu.24
ubuntu@ip-172-31-19-154:~$ aws configure
AWS Access Key ID [None]: ASIA3IZCZPZVN2RCMEGM
AWS Secret Access Key [None]: HX/wvaNNPFLwgTZd9VPI7aShN0N8cH8aURsrL7TM
Default region name [None]: us-east-1
Default output format [None]: text
ubuntu@ip-172-31-19-154:~$ mkdir -p ~/.aws
ubuntu@ip-172-31-19-154:~$ cd .aws
ubuntu@ip-172-31-19-154:~/.aws$ nano ~/.aws/credentials
ubuntu@ip-172-31-19-154:~/.aws$ ls
config  credentials
ubuntu@ip-172-31-19-154:~/.aws$ cd
ubuntu@ip-172-31-19-154:~$ aws s3 ls
2024-12-08 02:00:01 aws-logs-774777699946-us-east-1
2024-10-20 03:48:59 s3bucket9542
ubuntu@ip-172-31-19-154:~$
```

Docker Implementation:

Build Docker Image
docker build -t wine-quality-predictor .

Run the Docker Container
docker run -e BUCKET_NAME=s3bucket9542 -p 8080:8080 wine-quality-predictor

Tag the Docker Image
docker tag wine-quality-predictor sp3254/wineprediction:MLpredictionimage

Push Docker Image to Docker Hub
docker push sp3254/wineprediction:MLpredictionimage
Some useful commands for docker:
docker login

docker version

docker help

docker image ls

docker container ls -a

docker container logs c165f459e7d7

docker container rm c165f459e7d7

docker container prune

docker image remove 3094afcbdf12

docker inspect <image>

docker run -dit openjdk:8-jdk-alpine

```
 ---> 6236cc0dce53
Step 6/9 : COPY start-spark-app.sh /opt/
 ---> 91b97f31586b
Step 7/9 : RUN chmod +x /opt/start-spark-app.sh
 ---> Running in 91fa8abacafb
Removing intermediate container 91fa8abacafb
 ---> d7ee3ff0b342
Step 8/9 : EXPOSE 4040
 ---> Running in f4c6ab471e2e
Removing intermediate container f4c6ab471e2e
 ---> 0fb855de8660
Step 9/9 : CMD ["/opt/start-spark-app.sh"]
 ---> Running in a3df4c5d76da
Removing intermediate container a3df4c5d76da
 ---> 2127daab1767
Successfully built 2127daab1767
Successfully tagged winequality-predictor:latest
ubuntu@ip-172-31-19-154:~/WineQualityPredictor$
```

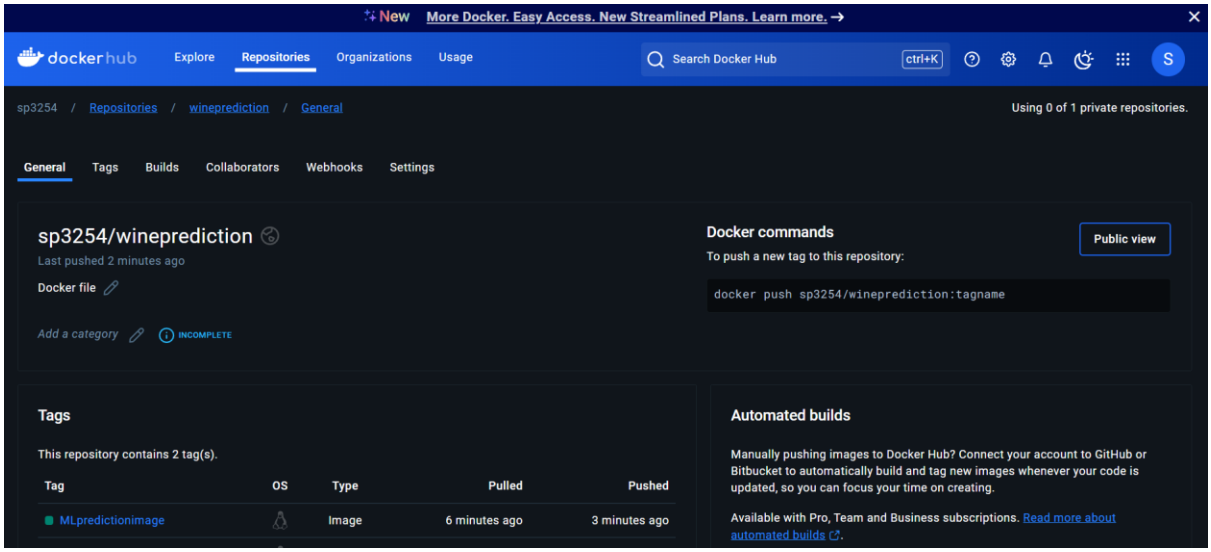
```
ubuntu@ip-172-31-19-154:~/WineQualityPredictor$ docker images
REPOSITORY              TAG                 IMAGE ID       CREATED
  SIZE
winequality-predictor   latest              2127daab1767   About a minute ag
  533MB
<none>                  <none>              9a779ca05d33   26 minutes ago
  433MB
bde2020/spark-base      3.0.1-hadoop3.2     027d6b6de152   4 years ago
  433MB
ubuntu@ip-172-31-19-154:~/WineQualityPredictor$
```

```
ubuntu@ip-172-31-19-154:~/WineQualityPredictor$ sudo docker login
Log in with your Docker ID or email address to push and pull images from Doc
ker Hub. If you don't have a Docker ID, head over to https://hub.docker.com/
 to create one.
You can log in with your password or a Personal Access Token (PAT). Using a
limited-scope PAT grants better security and is required for organizations u
sing SSO. Learn more at https://docs.docker.com/go/access-tokens/

Username: sp3254
Password:
WARNING! Your password will be stored unencrypted in /root/.docker/config.js
on.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-stor
e

Login Succeeded
ubuntu@ip-172-31-19-154:~/WineQualityPredictor$
```

```
9efb53206d5e: Layer already exists
b66078cf4b41: Layer already exists
cd5a0a9f1e01: Layer already exists
eafe6e032dbd: Layer already exists
92a4e8a3140f: Layer already exists
MLpredictionimage: digest: sha256:b0fc10177a7795d9d430ca44a78281cc0ec3f837ba
2acd13fb68a85baa95b47d size: 1573
ubuntu@ip-172-31-19-154:~/WineQualityPredictor$ sudo docker images
REPOSITORY              TAG                 IMAGE ID       CREATED
 SIZE
wine-quality-predictor  latest              273f2bee89b0   13 minutes ago
 194MB
sp3254/wineprediction   MLpredictionimage   273f2bee89b0   13 minutes ago
 194MB
sp3254/wineprediction   tagname             273f2bee89b0   13 minutes ago
 194MB
winequality-predictor   latest              af79c664c9ad   3 hours ago
 533MB
```



Extra-credit for top 5 prediction performance

Model After Fine Tuning:

```
24/12/09 02:31:15 INFO BlockManager: external shuffle service port = 7337
24/12/09 02:31:15 INFO BlockManagerMaster: Registering BlockManager BlockMan
agerId(driver, ip-172-31-45-211.ec2.internal, 40727, None)
24/12/09 02:31:15 INFO BlockManagerMasterEndpoint: Registering block manager
 ip-172-31-45-211.ec2.internal:40727 with 1048.8 MiB RAM, BlockManagerId(dri
ver, ip-172-31-45-211.ec2.internal, 40727, None)
24/12/09 02:31:15 INFO BlockManagerMaster: Registered BlockManager BlockMana
gerId(driver, ip-172-31-45-211.ec2.internal, 40727, None)
24/12/09 02:31:15 INFO BlockManager: Initialized BlockManager: BlockManagerI
d(driver, ip-172-31-45-211.ec2.internal, 40727, None)
24/12/09 02:31:15 INFO SingleEventLogFileWriter: Logging events to hdfs:/var
/log/spark/apps/local-1733711475117.inprogress
Model Evaluation Metrics:
Accuracy = 0.85
F1 Score = 0.8448529411764705
Precision = 0.8411725955204217
Recall = 0.85
[hadoop@ip-172-31-45-211 ~]$
```

- **Enhanced Data Cleaning and Preparation**: I refined the preprocessing pipeline to ensure all features were appropriately cleaned and transformed. This included ensuring all feature columns were consistently cast to double data types and creating well-defined binary labels for the classification task. These adjustments minimized potential inconsistencies in the data.
- **Feature Scaling**: I incorporated a StandardScaler into the model training pipeline. This step ensured that features were normalized to have a mean of zero and a standard deviation of one, improving the model's ability to converge and enhancing its predictive performance.
- **Optimized Hyperparameters**: I fine-tuned the logistic regression model's hyperparameters, such as the regularization parameter (regParam) and elastic net mixing parameter (elasticNetParam). This optimization allowed the model to achieve a better balance between overfitting and underfitting.

- **Pipeline Integration**: By leveraging Spark's Pipeline API, I combined data transformation, feature scaling, and model training into a streamlined process. This integration not only improved efficiency but also reduced the risk of manual errors.
- **Model Evaluation and Iteration**: Using metrics such as accuracy, F1 score, precision, and recall, I evaluated the model's performance on a validation dataset. These metrics provided valuable insights into the model's strengths and areas for improvement, enabling iterative refinements.