Final Term Project – CS634 Data Mining

Instructed by Prof. Jason Wang

First-name:	Rahul Gautham
Last-name:	Putcha
NJIT-ID:	31524074
UCID	RP39
Email-address:	rp39@njit.edu
Option:	5

<u>Topic on focus</u>

Data Mining Using Hadoop/Spark in the Cloud

TABLE OF CONTENT

List of Tables and Figures

Chapter 1. General Introduction

With the amount of data increasing at an astounding rate, it has become difficult for people to get useful insights from the large dataset in real-time. Data mining has provided us with a vast set of algorithms that are being used in a variety of applications and have proved to be reliable from time to time.

After years of research, the field of Mathematics have evolved with thirst of automation in computer science. Automation in computer science led us with to new ways to resolve a problem using sophisticated mathematical concepts, one of which is by means of Machine Learning. Machine Learning is a process of making a computer (or a machine) learn a task by using repeated experimentation, with an associated performance for improvement included at every experiment conducted on that task.

Data mining, is a crucial part of machine learning which help a computer find (or mine) useful patterns over a large dataset conquered from various sources. Therefore, Machine learning leverages data mining and computational intelligence algorithms to improve decision making models [3].

1.1 The Classification problem

In this project manual we will be focusing one of many and also an important decision-making process, namely the Classification problem.

Classification is the problem of identifying which of a set of categories (sub-populations) an observation, (or observations) belongs to [4].

In other word, if you are given few pictures of animals and you are asked to identify what kind of animal it is from a predefined set of known animals, for example: cat, dog, cow, rabbit, giraffe or lion, we are conducting a classification and the problem is popularly known as image classification problem.

In above example, the image is called as a feature and the predefined set (for eg: cow, dog or rabbit) are called labels or targets, of classification. The features need not be an image, it can be collection of attributes with values, such as whether it is an invertebrate or vertebrate, warm or cold blooded, joint or has no legs, etc. Similarly, the label we classify need not be of one type (aka, multi-class classification), but it can be more than one (aka, multi-label classification).

For the remainder of this project, we will be focusing on solving the problem of predicting possibility of Breast-Cancer over a record of patient taken from a hospital in Wisconsin, or the *Breast Cancer Wisconsin Dataset* [1][2]. This dataset is publicly available and the details about the dataset are given in *Chapter 5: Implementation*. In the next section, we will be briefing on the problem for predicting possibility of Breast-Cancer for patients in a hospital at Wisconsin.

1.2 Problem Statement

Breast cancer is most common among women in Wisconsin regardless of race. It accounts for nearly one-third of all cancers diagnosed among women [5]. This dataset was first published and is currently publicly available at https://archive.ics.uci.edu/ml/datasets repository. Features are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. They describe characteristics of the cell nuclei present in the image [1].

Following are the list and details of attributes found within the dataset [1]:

1) ID number
2) Diagnosis (M = malignant, B = benign)
3-32) Ten real-valued features are computed for each cell nucleus:
    a) radius (mean of distances from center to points on the perimeter)
    b) texture (standard deviation of gray-scale values)
    c) perimeter
    d) area
    e) smoothness (local variation in radius lengths)
    f) compactness (perimeter^2 / area - 1.0)
    g) concavity (severity of concave portions of the contour)
    h) concave points (number of concave portions of the contour)
    i) symmetry
    j) fractal dimension ("coastline approximation" - 1)

With the details mentioned in above list, we get to know that the labels we are predicting are the outcome of diagnosis results, given in Diagnosis attribute. And the rest of the attributes are all collectively used within our features for predicting the target / label.

1.3 What will we be doing and achieving by the end of this manual?

Now that we are familiar with our dataset, lets discuss on what we will be doing for the remainder of the project, i.e., how will we carry out the job of automating prediction of a task by learning a dataset.

Firstly, in the *Chapter 2: Focusing on Algorithm*, we will be discussing on one of the widely used classification algorithm, the Random Forest (RF) Classification algorithm. We will understand the working of an RF and then shift to what we require for carrying the classification task, in *Chapter 3: Requirement Specifications*. In Chapter 3, we will discuss briefly about Apache Spark, a tool used for parallel processing, with its special built-in library for Machine Learning, the Spark MLlib library.

We will also gain hands-on for using EMR (Elastic Map Reduce), a popular platform as a service (PaaS) present of the world-renowned cloud provider, the Amazon Web Service (AWS). This saves us the burden of setting up and installing/configuring software and unknown dependencies, especially Apache Spark and Hadoop HDFS.

In *Chapter 4: Setting up the Environment*, we will be presenting step by step details on setting up the application from scratch with the application source code presented in *Chapter 5: Implementation*.

And lastly, in *Chapter 6: Running the application on Apache Spark inside the EMR (Elastic MapReduce)*, we will guide you through the process of running the application on the cloud both for training and testing over the Breast-Cancer-Wisconsin's dataset.

Therefore, by the end of the project manual we will be building a trained model for predicting the possibility of breast cancer in patients at Wisconsin and present an evaluation metric that estimate the correctness of the model.

Perquisite: *It is assumed that the user has slight familiarity with AWS services and Management Console. Explanation related to AWS services and tool are presented in scope to the Final Term Project.*

Chapter 2. Focusing on Algorithm

In this chapter we will be focusing on one of the widely used classification algorithm, the *Random Forest (RF) Classification* algorithm and give a detail on how & why Random Forest algorithm efficiently solves classification problem.

2.1 Using Random Forest for Classification

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time [6]. In ensemble learning, we combine prediction of two or more models.

In Random Forest, we do this by constructing multiple decision tree over random subsets chosen from the training set. The model then classifies the label, via voting mechanism taken over the results of all random decision tree. Hence, RFs chooses the label that is produced by majority number of decision trees. Although the making of a number of decision tree is a rather overwhelming task, the correctness of model evaluated is more accurate compared to a model trained using other classification algorithms. In the next section we will details on working of Random Forest with an example.

2.2 How does it solve the problem?

The key behind random forest problem is a decision tree. Decision trees are basic building block of Random forests. It's a classification algorithm that give the solution by traversing levels involving series of questions. Each question determines conditions facing the observation to determine the kind of answer label. By this we narrow the possible values till we are confident enough to make a single prediction similar to our daily lives.

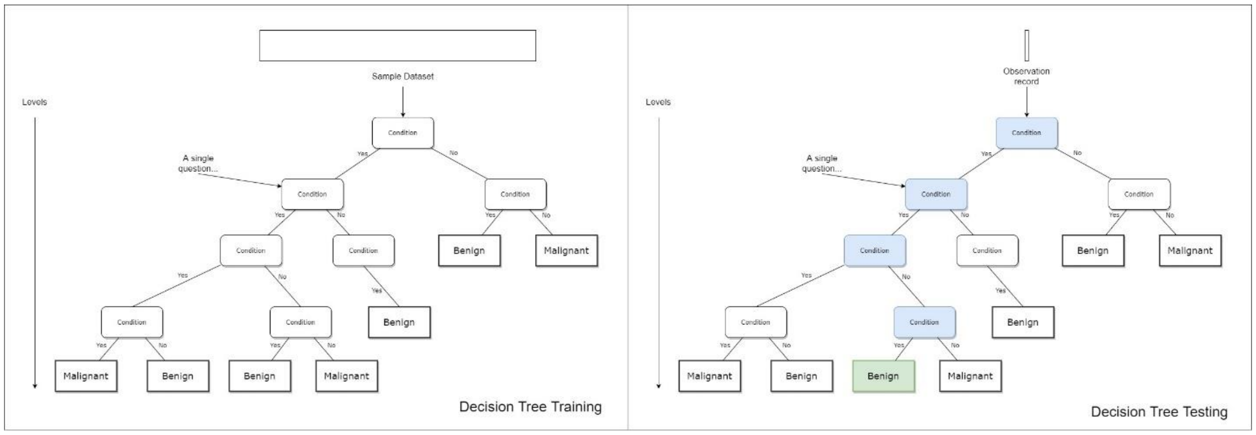


Fig 2.1 Decision Tree

As given in the figure, a tree is constructed over the entire dataset by using the entropy and information gain measure. While testing, we take the observation/sample data and run it over series of question to classify the label. Although a decision tree enforces the consideration of all possible outcomes of a decision and traces each path to a conclusion, they are unstable. This means, a small change in the data can lead to a large change in the structure of the optimal decision tree.

Hence, to resolve this problem we are using Random Forest for classification. In the below figure, we see a sample on how a random forest is generated. Random Forest trains itself by taking random subsets of training set and constructing a number of decision trees over each individual subset.

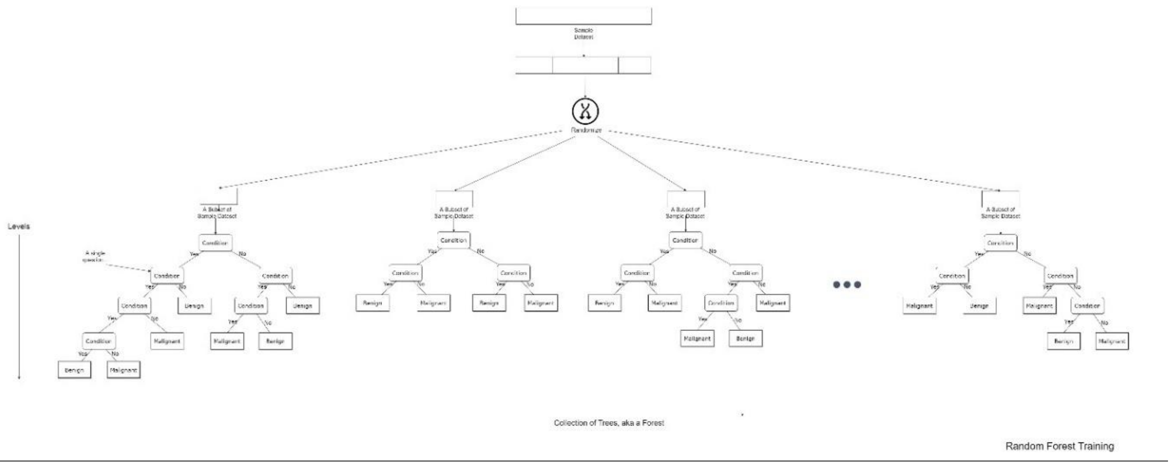


Fig 2.2 Random Forest Training

Similarly, the testing is conducted by grouping the individual resulting labels and aggregating the counts. Finally, the result is picked as whichever label that is picked by the majority number of decision trees, as shown in the figure below. Here, we see that the final result is negative or Malignant label.

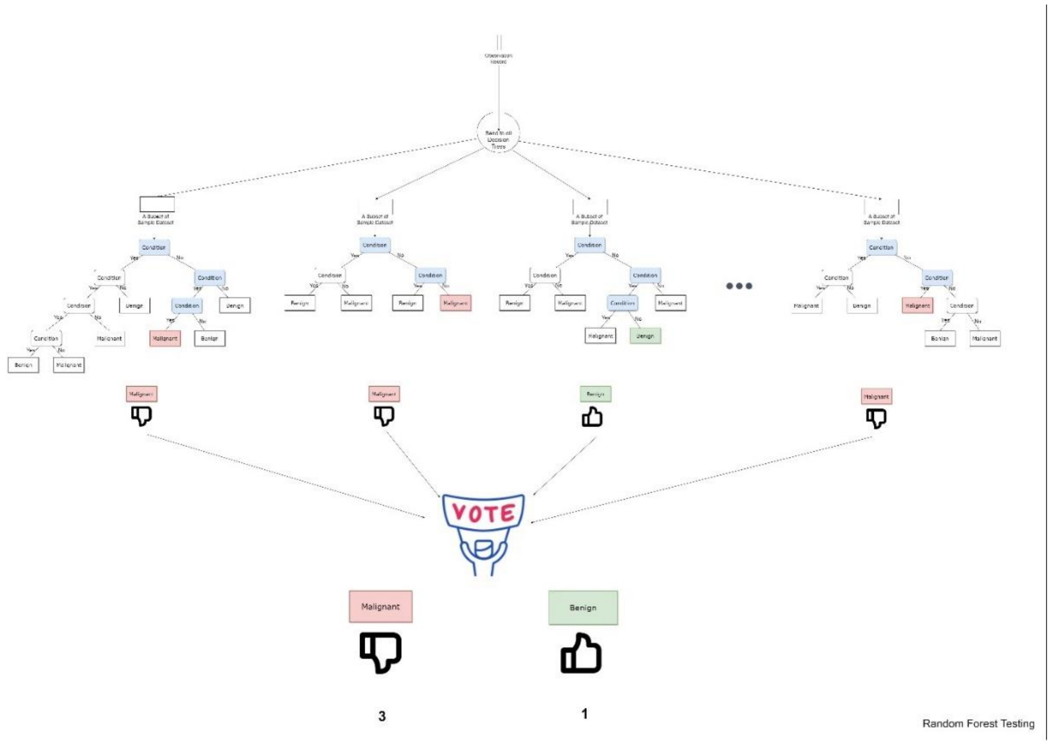


Fig 2.3 Random Forest Testing

By this we conclude Chapter 2. In the next chapter we will focus on the requirements for the application to classification problem.

Chapter 3. Requirement Specifications

This chapter presents the basic requirement to run the application for classification using Apache Spark on the cloud. This chapter also highlights all necessary task/requirements that we are going to accomplish in the later chapters.

3.1 System Requirements

In order to run the application for a general classification problem, such as Breast Cancer at Wisconsin, the following mentions the hardware and software requirements.

ReqID	Requirements	Details
Sys-1	AWS Elastic Map Reduce (EMR) • Apache Spark 2.4.7 • Hadoop 2.10 • Cluster mode launch • 4 EC2 instances used • m5.xlarge instance type	Requires AWS account. Warning: Additional cost involved!! General Linux cluster. 1 Master and 3 Slaves nodes. Spark 2.4.7 on Hadoop 2.10. For more details visit, AWS EMR page or please review Chapter 4: Setting up the Environment.
Sys-2	Programming Language: Java	Java v 1.8 (Java JDK8) (Programming Language used) *// Other Language Required: Scala v 2.12 (For Apache Spark)*
Sys-3	S3 (Simple Storage Service)	For storing dataset and final trained model for future predictions.
Sys-4	AWS CLI	AWS CLI for working with Spark on AWS resources in an efficient manner. Requires AWS programmer credential key setup.
Sys-5	SSH-Terminal App	Mobaxterm or Command Prompt with Open SSH installed.
Sys-6	Apache Maven	Apache Maven version 3.8.1
Sys-7	Apache Spark Core + Spark ML library + Spark SQL Library	spark-core_2.12 v 3.1.2, spark-mllib_2.12 v 3.1.2, spark-sql_2.12 v 3.1.2
Sys-8	Kaggle Dataset: Breast Cancer Wisconsin's	To download please visit the following link, Breast Cancer Wisconsin. Requires Kaggle account.

Tab 3.1 System Requirements

3.2 Functional Requirements

This section presents the requirements that we want our final-build of our application to fulfill. This is specific to the Final Term Project:

ReqID	Details
FR-1	Setup EMR on AWS to access any AWS resources needed for the project
FR-2	Use Apache Spark for designing ML model, trained parallel over a cluster of 4 nodes (instances).
FR-3	Connect to file storage service for storing and accessing data (for trained model and datasets)
FR-4	Take input (Command Line Argument) from user for taking dataset of choice
FR-5	Automate process for making dataset more efficient by necessary Data Preprocessing.
FR-6	Split the dataset into 2 parts. One used as a training set and another as a testing set.
FR-7	Build a model using a Data Mining algorithm (Random Forest) on any dataset (for example: Breast Cancer Wisconsin dataset)
FR-8	Evaluate model accuracy using Classification Matrix.

Tab 3.2 Functional Requirements

Chapter 4. Setting up the Environment

This chapter will give a tutorial on setting up infrastructure resources on AWS cloud. The infrastructure we will be working on is Elastic Map Reduce (EMR), which is a popular PaaS for building distributed application on cloud. We will then look into setting of S3, a cloud storage service on AWS, which we will be using for storing our dataset and final trained model for future testing and predictions.

And finally wind up with a general introduction to Apache Spark and Apache Maven which we will be using for setting our application in the next chapter.

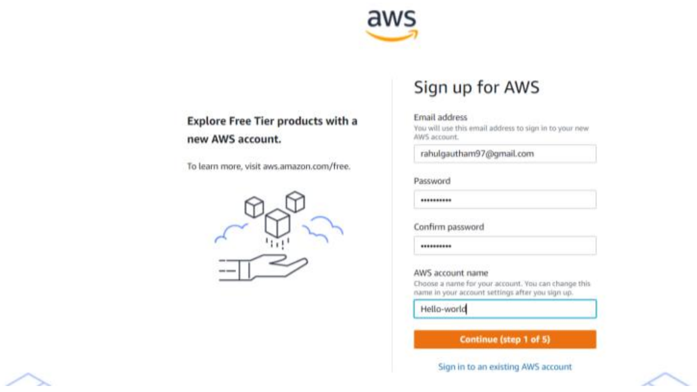
4.1 Setting up the AWS Account

In order to use AWS resources we are required to setup an AWS account. AWS give us variety ways to setup an account.

Personal Account

This involves:

- Filling up the form with user details, password credentials and unique account name.
- AWS also requires contact details for billing or promotional contacting in future.
- A personal account with a free-tier is sufficient for this project needs.
- You may also require a credit card, incase you have exceeded your free-tier limit.



Sign up for AWS

Explore Free Tier products with a new AWS account.

To learn more, visit aws.amazon.com/free.

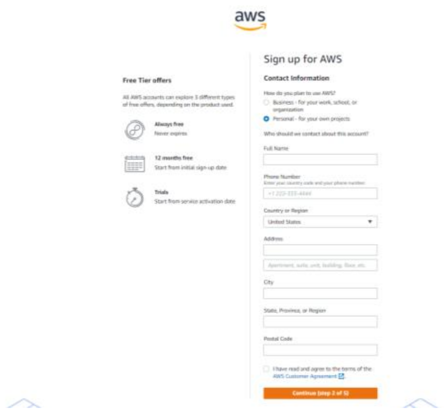
Email address
You will use this email address to sign in to your new AWS account.
rahulgautham97@gmail.com

Password
••••••••••

Confirm password
••••••••••

AWS account name
Choose a name for your account. You can change this name in your account settings after you sign up.
Hello-world

Fig 4.1 Sign-in Page (Step-1)



Sign up for AWS

Free Tier offers

All AWS accounts can explore 3 different types of free offers, depending on the product used.

Always free
Never expires

12 months free
Start from initial sign-up date

Trials
Start from service activation date

Contact Information

How do you plan to use AWS?
Business - for your work, school, or organization [✓ Personal - for your own projects]

Who should we contact about this account?

Full Name

Phone Number
Enter your country code and your phone number
+1 222-333-4444

Country or Region
United States

Address
Apartment, suite, unit, building, floor, etc.

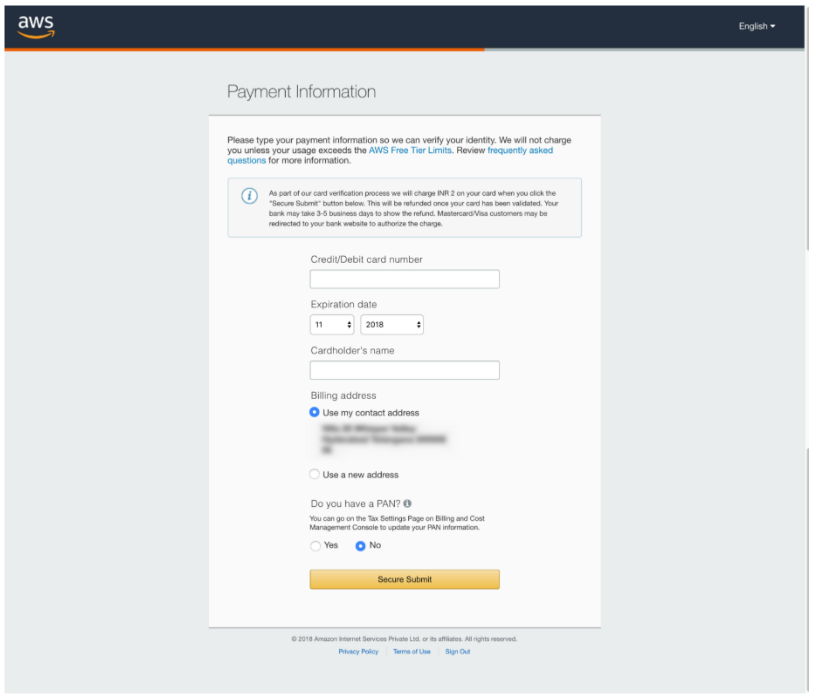
City

State, Province, or Region

Postal Code

I have read and agree to the terms of the AWS Customer Agreement

Fig 4.2 Sign-in Page (Step-2)



Payment Information

Please type your payment information so we can verify your identity. We will not charge you unless your usage exceeds the AWS Free Tier Limits. Review frequently asked questions for more information.

As part of our card verification process we will charge INR 2 on your card when you click the "Secure Submit" button below. This will be refunded once your card has been validated. Your bank may take 3-5 business days to show the refund. Mastercard/Visa customers may be redirected to your bank website to authorize the charge.

Credit/Debit card number

Expiration date: 11 / 2018

Cardholder's name

Billing address
[✓ Use my contact address]
[redacted]
Use a new address

Do you have a PAN?
You can go on the Tax Settings Page on Billing and Cost Management Console to update your PAN information.
Yes [✓ No]

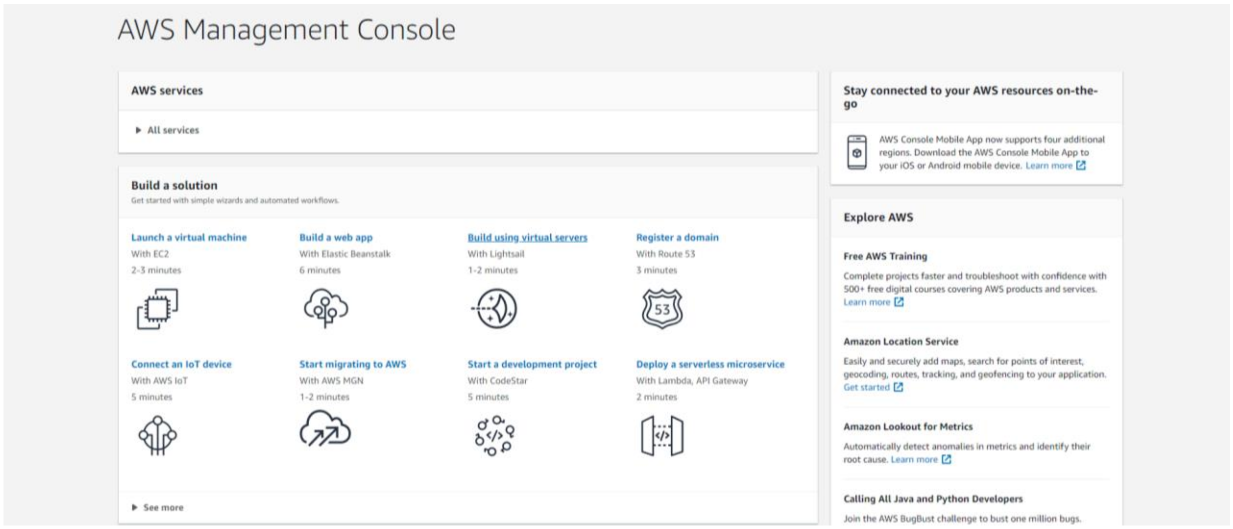
Secure Submit

© 2018 Amazon Internet Services Private Ltd. or its affiliates. All rights reserved.
Privacy Policy | Terms of Use | Sign Out

Fig 4.3 Credit-card Page

(Source: https://docs.sendwithses.com/how/aws-setups/how-to-create-a-free-aws-account)

- After verifying your contact and credit card details, you will land in to the AWS service home page, aka, the Management Console page


AWS Management Console

AWS services
▸ All services

Build a solution
Get started with simple wizards and automated workflows.

Launch a virtual machine
With EC2
2-3 minutes

Build a web app
With Elastic Beanstalk
6 minutes

Build using virtual servers
With Lightsail
1-2 minutes

Register a domain
With Route 53
3 minutes

Connect an IoT device
With AWS IoT
5 minutes

Start migrating to AWS
With AWS MGN
1-2 minutes

Start a development project
With CodeStar
5 minutes

Deploy a serverless microservice
With Lambda, API Gateway
2 minutes

▸ See more

Stay connected to your AWS resources on-the-go
AWS Console Mobile App now supports four additional regions. Download the AWS Console Mobile App to your iOS or Android mobile device. Learn more

Explore AWS

Free AWS Training
Complete projects faster and troubleshoot with confidence with 500+ free digital courses covering AWS products and services. Learn more

Amazon Location Service
Easily and securely add maps, search for points of interest, geocoding, routes, tracking, and geofencing to your application. Get started

Amazon Lookout for Metrics
Automatically detect anomalies in metrics and identify their root cause. Learn more

Calling All Java and Python Developers
Join the AWS BugBust challenge to bust one million bugs.

Fig 4.4 AWS Service Home Page (a.k.a Management Console page)

Note that the UI changes often depending on AWS satisfaction to the customer in the future, but the access to the resources may remain same to a certain extent as show in this project manual.

Student Account

AWS also offers educational account for student studying in the university. Students can access AWS cloud infrastructure without a need for any credit card details. This may require their university email id. For more details please visit, AWS Educate.

4.2 Setting up EMR (Elastic MapReduce) on AWS

Let's now focus on setting up the AWS service, Elastic MapReduce (EMR), which will be acting as the base infrastructure on which our application runs. EMR provides a simple way to setup master-slave architecture, another term for a cluster of nodes which commonly referred in distributed systems.

General Idea about a Cluster

A computer cluster is a set of computers that work together so that they can be viewed as a single system [7]. Theses cluster are connected over a local network with each having their own operating system and follow a common set of protocols with a setup that make it part of the localized network. A cluster consist of a managing entity, or a master, that makes a computer system or worker/slave node part of the network, and it accepts large workload from user. This large workload is split into chunks of sequential and parallel tasks that are later assigned to each of the worker/slave nodes.

This way we can see that any task can be executed efficiently and parallelly among a number of computers in the distributed system. For the project's sake, we are focusing on using this ability of cluster wherein we feed in the dataset to the master to create random forest. This will be done by making individual slaves in the cluster handle the task of creating the individual decision trees from a chosen sample subset of the dataset (Refer to Chapter 2 for more details).

Making the setting up and designing a program for exhibiting such behavior in network of computer systems from scratch by means of a cluster is daunting task. Although we are capable of doing the same using only Hadoop, Apache Spark provides a simple built-in library that resolves the task of setting up a cluster to assigning random decision trees training over worker nodes, via Spark ML library (We will be seeing this in later section of this chapter).

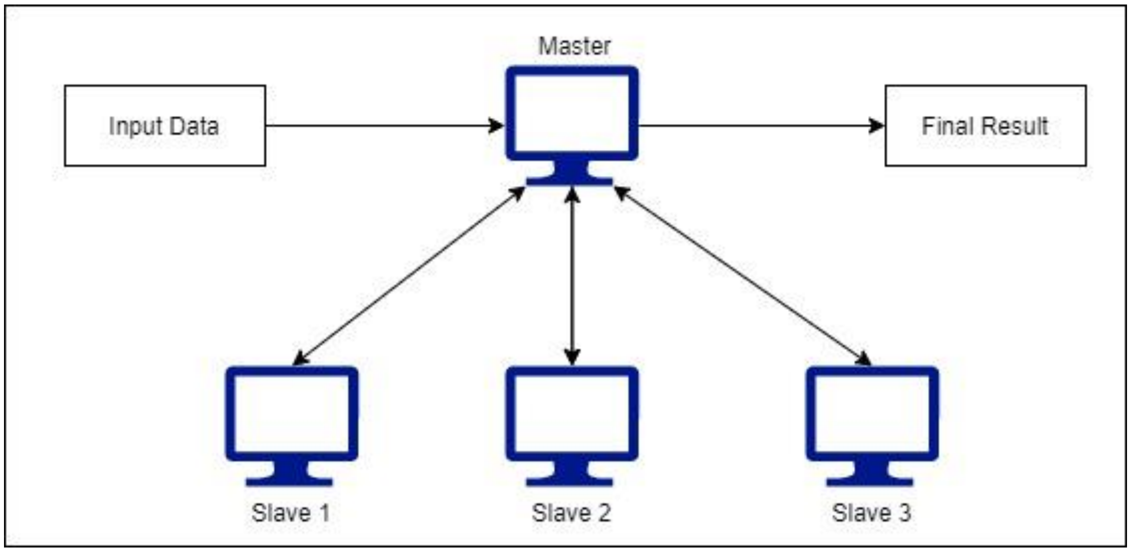


Fig 4.5 Cluster Network Architecture

About EMR and EC2(s)

Amazon EMR is the cloud big data platform for processing vast amounts of data using open-source tools such as Apache Spark, Apache Hive, Apache HBase, Apache Flink, Apache Hudi, and Presto [8]. It automatically makes the setup, operate and scaling of a big data environments on a chosen platform and tools. This setting may incur cost, but the hourly billing is nominal and we only have to pay for what we use per hour, i.e., pay for each of the EC2 instances we use per hour.

Creating a Cluster on EMR

Before we create our first EMR cluster, we would require a EC2 access key-pair for securely accessing our individual nodes using our remote terminal of choice. To do this:

- Search for EC2 and click on EC2 service

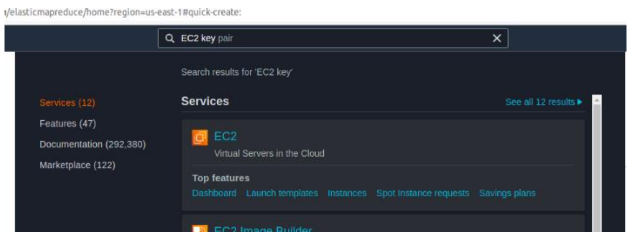
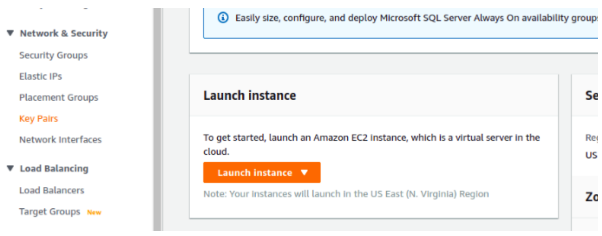


Fig 4.6 Searching for EC2 service



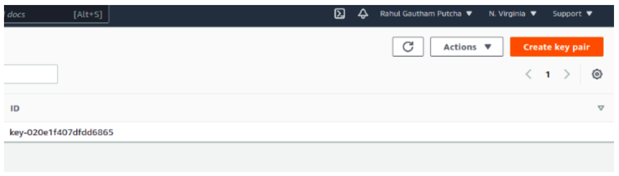
Easily size, configure, and deploy Microsoft SQL Server Always On availability groups

To get started, launch an Amazon EC2 instance, which is a virtual server in the cloud.

Note: Your instances will launch in the US East (N. Virginia) Region

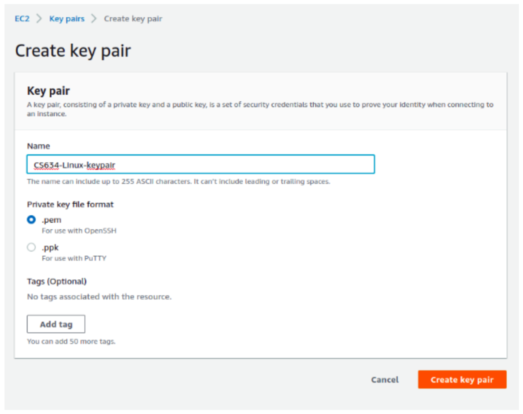
Fig 4.7 Key-Pair Tab option

- Click on Key-pair option on the left tab menu.
- Fill up the form. If you have a Linux/Mac terminal, please select the .pem and for Windows, please select .ppk option. Finally click on Create key pair button.



key-020e1f407dfdd6865

Fig 4.8 Create key pair button



A key pair, consisting of a private key and a public key, is a set of security credentials that you use to prove your identity when connecting to an instance.

Name: CS634-Linux-keypair

The name can include up to 255 ASCII characters. It can't include leading or trailing spaces.

Private key file format
[✓ .pem] For use with OpenSSH
.ppk For use with PuTTY

Tags (Optional)
No tags associated with the resource.

You can add 50 more tags.

Fig 4.9 Create your first key pair

Your key pair is created and downloaded. Please make sure to save this file, we will be using this in an upcoming section. Now, you are now ready to setup the EMR cluster. Following are the steps to setup an EMR on AWS cloud:

- On management console, use the search bar to search for EMR service and select *EMR.*
- After landing to the EMR page, click on Create Cluster.
- Fill up the form as show in, *Fig 4.8,* and then click on Create cluster.

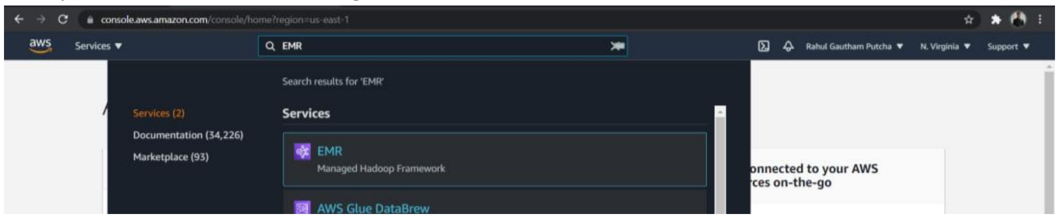
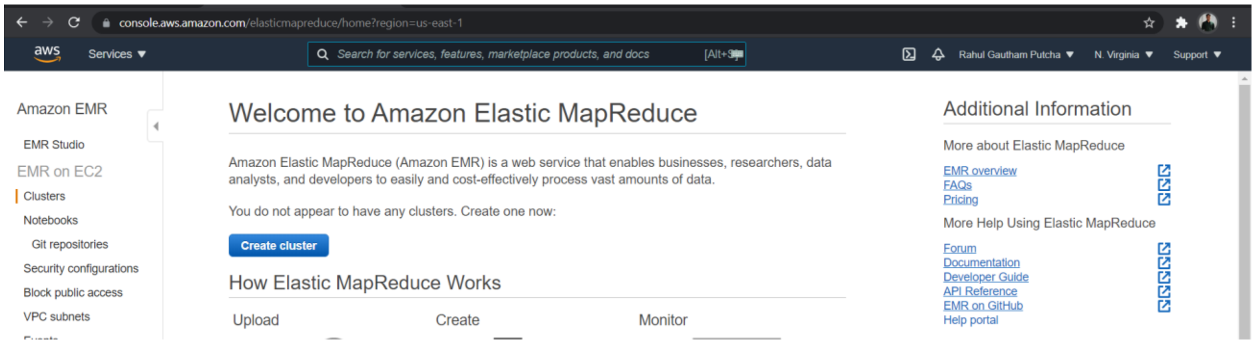


Fig 4.10 Searching for EMR service



Welcome to Amazon Elastic MapReduce

Amazon Elastic MapReduce (Amazon EMR) is a web service that enables businesses, researchers, data analysts, and developers to easily and cost-effectively process vast amounts of data.

You do not appear to have any clusters. Create one now:

How Elastic MapReduce Works

Additional Information

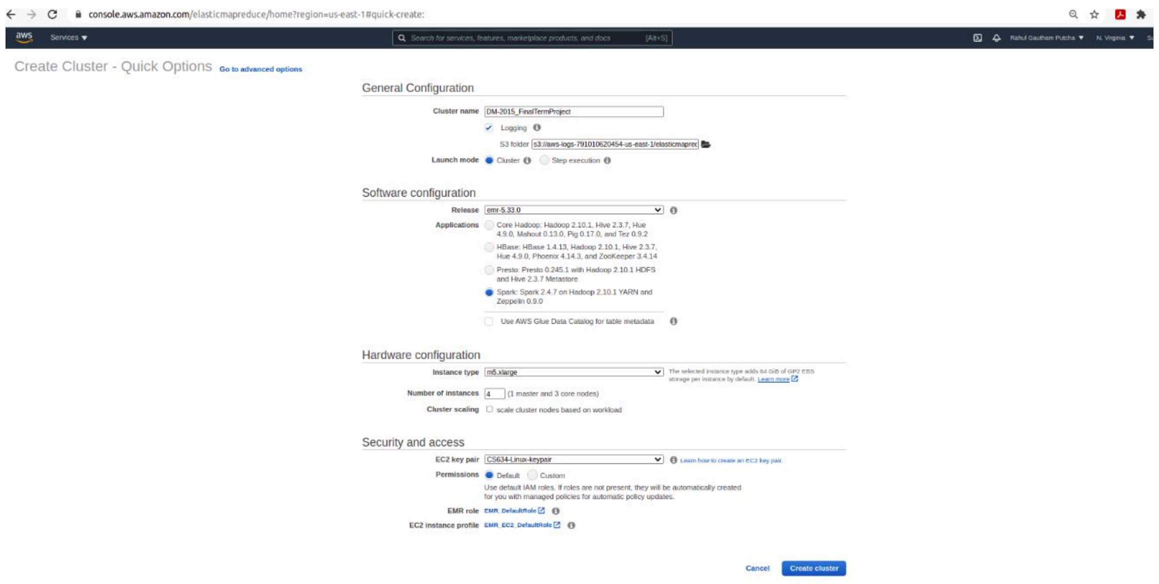
More about Elastic MapReduce

EMR overview
FAQs
Pricing

More Help Using Elastic MapReduce

Forum
Documentation
Developer Guide
API Reference
EMR on GitHub
Help portal

Fig 4.11 EMR service: Home page (For creating a cluster)


console.aws.amazon.com/elasticmapreduce/home?region=us-east-1#quick-create:
aws
Services
Search for services, features, marketplace products, and docs
Rahul Gautham Putcha
N. Virginia
Go to advanced options
Cancel
Create cluster

Create Cluster - Quick Options

General Configuration

Cluster name: DM-2015_FinalTermProject
[✓] Logging
S3 folder: s3://aws-logs-791010620454-us-east-1/elasticmaprec
Launch mode: [✓ Cluster] Step execution

Software configuration

Release: emr-5.33.0
Applications:
- Core Hadoop: Hadoop 2.10.1, Hive 2.3.7, Hue 4.9.0, Mahout 0.13.0, Pig 0.17.0, and Tez 0.9.2
- HBase: HBase 1.4.13, Hadoop 2.10.1, Hive 2.3.7, Hue 4.9.0, Phoenix 4.14.3, and ZooKeeper 3.4.14
- Presto: Presto 0.245.1 with Hadoop 2.10.1 HDFS and Hive 2.3.7 Metastore
- [✓ Spark: Spark 2.4.7 on Hadoop 2.10.1 YARN and Zeppelin 0.9.0]

Use AWS Glue Data Catalog for table metadata

Hardware configuration

Instance type: m5.xlarge — The selected instance type adds 64 GiB of GP2 EBS storage per instance by default. Learn more
Number of instances: 4 (1 master and 3 core nodes)
Cluster scaling: scale cluster nodes based on workload

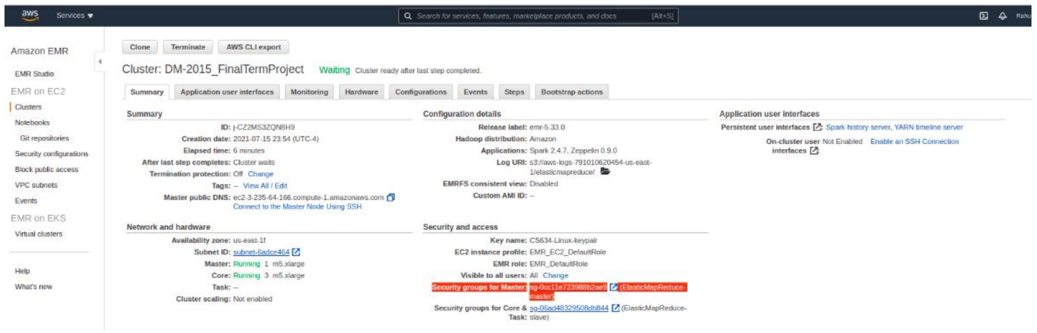
Security and access

EC2 key pair: CS634-Linux-keypair — Learn how to create an EC2 key pair.
Permissions: [✓ Default] Custom
Use default IAM roles. If roles are not present, they will be automatically created for you with managed policies for automatic policy updates.
EMR role: EMR_DefaultRole
EC2 instance profile: EMR_EC2_DefaultRole

Fig 4.12 Creating a cluster using EMR service

After this your cluster is now created and organized using EMR. We will now access the master node using the cluster we have just setup. In order to access our cluster, we first need to define a SSH rule in our EMR-master security groups. Follow steps below to setup the Security group for Master node to access it via SSH.

- After EMR cluster indicates that its running, click on Security group for Master (highlighted in the Fig below)



Cluster: DM-2015_FinalTermProject Waiting Cluster ready after last step completed.

Summary

ID: j-CZ2MS3ZQN8H9
Creation date: 2021-07-15 23:54 (UTC-4)
Elapsed time: 6 minutes
After last step completes: Cluster waits
Termination protection: Off Change
Tags: -- View All / Edit
Master public DNS: ec2-3-235-64-166.compute-1.amazonaws.com
Connect to the Master Node Using SSH

Network and hardware

Availability zone: us-east-1f
Subnet ID: subnet-6adce464
Master: Running 1 m5.xlarge
Core: Running 3 m5.xlarge
Task: --
Cluster scaling: Not enabled

Configuration details

Release label: emr-5.33.0
Hadoop distribution: Amazon
Applications: Spark 2.4.7, Zeppelin 0.9.0
Log URI: s3://aws-logs-791010620454-us-east-1/elasticmapreduce/
EMRFS consistent view: Disabled
Custom AMI ID: --

Security and access

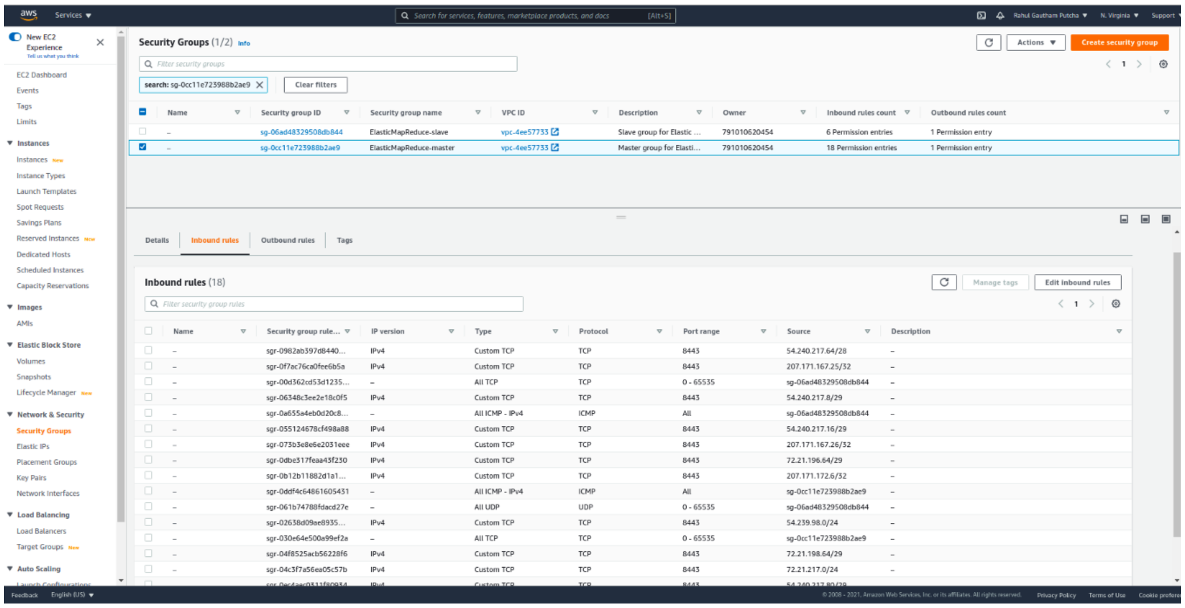
Key name: CS634-Linux-keypair
EC2 instance profile: EMR_EC2_DefaultRole
EMR role: EMR_DefaultRole
Visible to all users: All Change
Security groups for Master: sg-0cc11e723988b2ae9 (ElasticMapReduce-master)
Security groups for Core & Task: sg-06ad48329508db844 (ElasticMapReduce-slave)

Application user interfaces

Persistent user interfaces: Spark history server, YARN timeline server
On-cluster user interfaces: Not Enabled Enable an SSH Connection

Fig 4.13 EMR Post-Cluster creation and Security group for Master Highlighted

- Select *ElasticMapReduce-master ('s) security group* and proceed to *Inbound rules tab.*
- Click on *Edit Inbound rules button* to see all rule available for a EMR by default.



Security Groups (1/2)

search: sg-0cc11e723988b2ae9

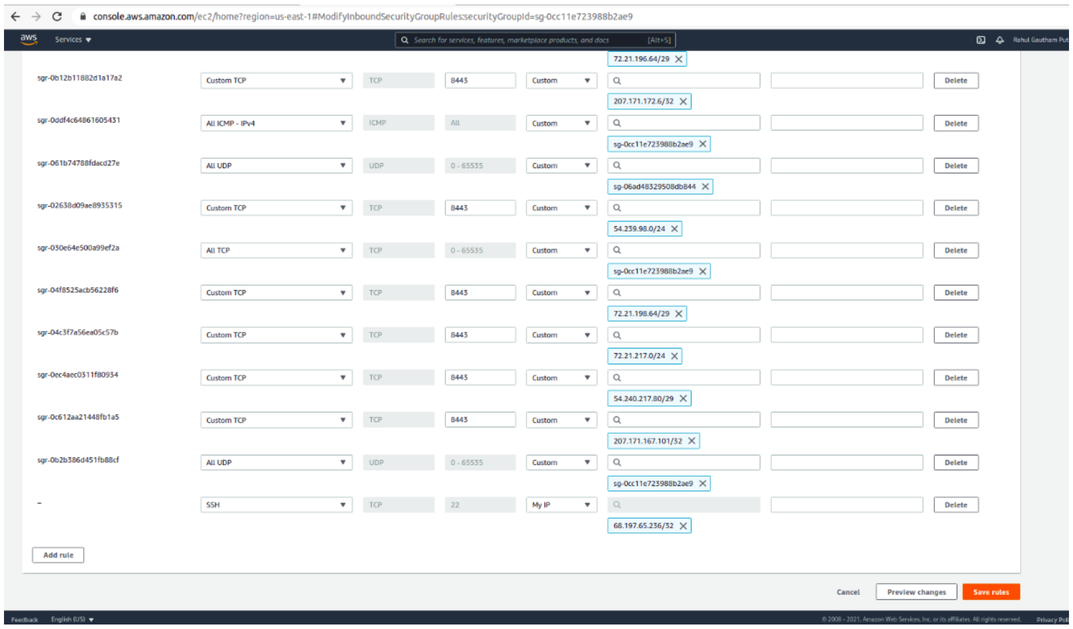
	Name	Security group ID	Security group name	VPC ID	Description	Owner	Inbound rules count	Outbound rules count
☐	–	sg-06ad48329508db844	ElasticMapReduce-slave	vpc-4ee57733	Slave group for Elastic ...	791010620454	6 Permission entries	1 Permission entry
✓	–	sg-0cc11e723988b2ae9	ElasticMapReduce-master	vpc-4ee57733	Master group for Elasti...	791010620454	18 Permission entries	1 Permission entry

Inbound rules (18)

Name	Security group rule...	IP version	Type	Protocol	Port range	Source	Description
–	sgr-0982ab397d8440...	IPv4	Custom TCP	TCP	8443	54.240.217.64/28	–
–	sgr-0f7ac76ca0fee6b5a	IPv4	Custom TCP	TCP	8443	207.171.167.25/32	–
–	sgr-00d362cd53d1235...	–	All TCP	TCP	0 - 65535	sg-06ad48329508db844	–
–	sgr-06348c3ee2e18c0f5	IPv4	Custom TCP	TCP	8443	54.240.217.8/29	–
–	sgr-0a655a4eb0d20c8...	–	All ICMP - IPv4	ICMP	All	sg-06ad48329508db844	–
–	sgr-055124678cf498a88	IPv4	Custom TCP	TCP	8443	54.240.217.16/29	–
–	sgr-073b3e8e6e2031eee	IPv4	Custom TCP	TCP	8443	207.171.167.26/32	–
–	sgr-0dbe317feaa43f230	IPv4	Custom TCP	TCP	8443	72.21.196.64/29	–
–	sgr-0b12b11882d1a1...	IPv4	Custom TCP	TCP	8443	207.171.172.6/32	–
–	sgr-0ddf4c64861605431	–	All ICMP - IPv4	ICMP	All	sg-0cc11e723988b2ae9	–
–	sgr-061b74788fdacd27e	–	All UDP	UDP	0 - 65535	sg-06ad48329508db844	–
–	sgr-02638d09ae8935...	IPv4	Custom TCP	TCP	8443	54.239.98.0/24	–
–	sgr-030e64e500a99ef2a	–	All TCP	TCP	0 - 65535	sg-0cc11e723988b2ae9	–
–	sgr-04f8525acb56228f6	IPv4	Custom TCP	TCP	8443	72.21.198.64/29	–
–	sgr-04c3f7a56ea05c57b	IPv4	Custom TCP	TCP	8443	72.21.217.0/24	–
–	sgr-0ec4aec0311f80934	IPv4	Custom TCP	TCP	8443	54.240.217.80/29	

Fig 4.14 Elastic MapReduce-master security group

- Add an SSH rule as shown below in *Fig 4.12* and click save rule.



console.aws.amazon.com/ec2/home?region=us-east-1#ModifyInboundSecurityGroupRules:securityGroupId=sg-0cc11e723988b2ae9

					72.21.196.64/29	
sgr-0b12b11882d1a17a2	Custom TCP	TCP	8443	Custom	207.171.172.6/32	
sgr-0ddf4c64861605431	All ICMP - IPv4	ICMP	All	Custom	sg-0cc11e723988b2ae9	
sgr-061b74788fdacd27e	All UDP	UDP	0 - 65535	Custom	sg-06ad48329508db844	
sgr-02638d09ae8935315	Custom TCP	TCP	8443	Custom	54.239.98.0/24	
sgr-030e64e500a99ef2a	All TCP	TCP	0 - 65535	Custom	sg-0cc11e723988b2ae9	
sgr-04f8525acb56228f6	Custom TCP	TCP	8443	Custom	72.21.198.64/29	
sgr-04c3f7a56ea05c57b	Custom TCP	TCP	8443	Custom	72.21.217.0/24	
sgr-0ec4aec0311f80934	Custom TCP	TCP	8443	Custom	54.240.217.80/29	
sgr-0c612aa21448fb1a5	Custom TCP	TCP	8443	Custom	207.171.167.101/32	
sgr-0b2b386d451fb88cf	All UDP	UDP	0 - 65535	Custom	sg-0cc11e723988b2ae9	
–	SSH	TCP	22	My IP	68.197.65.236/32	

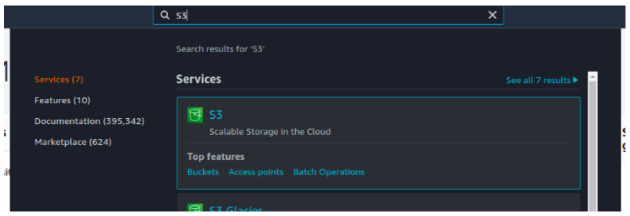
Fig 4.15 Adding a SSH rule to EMR-master

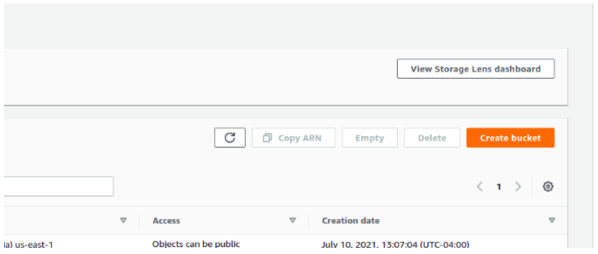
This will allow us to securely access the EMR's EC2 master node using any remote terminal of choice. The next step is to use SSH within our terminal of choice, which we will be discussing in a later section. Before that as we are now still on AWS, lets setup a file storage for holding our dataset and model files.

4.3 Setting up a S3 (Simple Storage Service) on AWS

Every Machine Learning (ML) application requires a persistent storage container for dataset file, especially a storage service that provides easy to use API for accessing containing files. AWS S3 provides a simple way to access files on EMR using the resource URI (Uniform Resource Identifier), which is associated with the file/resource we are referring.

- Using the search bar, search for S3 and select *S3 storage service*.
- In the S3 page, click on *Create bucket*.



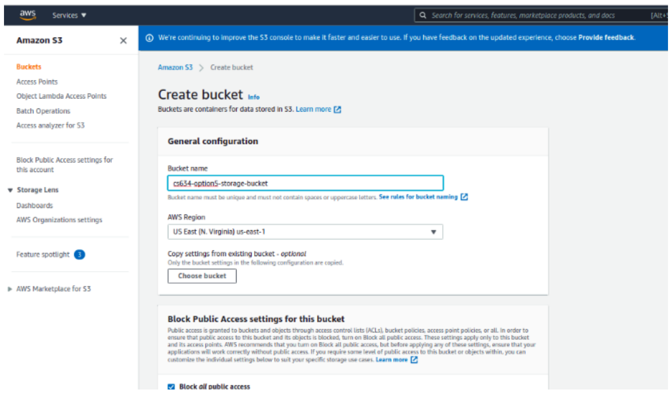


ia) us-east-1 | Objects can be public | July 10, 2021, 13:07:04 (UTC-04:00)

Fig 4.16 Searching for S3 service

Fig 4.17 Create Bucket button (Top Right-side)

- This will show us a form. Just fill in the *Bucket name,* must be a globally unique name, if needed adjust the region to your closest or most compliant region of choice.
- Scroll down and click *Create Bucket button*.


aws
Services
Search for services, features, marketplace products, and docs
[Alt+
Amazon S3
Buckets
Access Points
Object Lambda Access Points
Batch Operations
Access analyzer for S3
Block Public Access settings for this account
Storage Lens
Dashboards
AWS Organizations settings
Feature spotlight 3
AWS Marketplace for S3
Amazon S3 > Create bucket
Info
Learn more
Choose bucket

We're continuing to improve the S3 console to make it faster and easier to use. If you have feedback on the updated experience, choose **Provide feedback**.

Create bucket

Buckets are containers for data stored in S3. Learn more

General configuration

Bucket name: cs634-option5-storage-bucket

Bucket name must be unique and must not contain spaces or uppercase letters. **See rules for bucket naming**

AWS Region: US East (N. Virginia) us-east-1

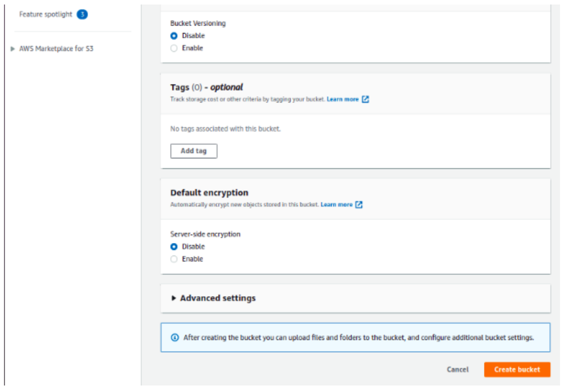
Copy settings from existing bucket - *optional*

Only the bucket settings in the following configuration are copied.

Block Public Access settings for this bucket

Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, access point policies, or all. In order to ensure that public access to this bucket and its objects is blocked, turn on Block all public access. These settings apply only to this bucket and its access points. AWS recommends that you turn on Block all public access, but before applying any of these settings, ensure that your applications will work correctly without public access. If you require some level of public access to this bucket or objects within, you can customize the individual settings below to suit your specific storage use cases. **Learn more**

☑ **Block *all* public access**



Bucket Versioning: [✓ Disable] Enable

Tags (0) - ***optional***

Track storage cost or other criteria by tagging your bucket. Learn more

No tags associated with this bucket.

Default encryption

Automatically encrypt new objects stored in this bucket. Learn more

Server-side encryption: [✓ Disable] Enable

After creating the bucket you can upload files and folders to the bucket, and configure additional bucket settings.

Fig 4.18 Filling Bucket name (must be unique)

Fig 4.19 Create Bucket button (in the form)

This will make a new bucket for us which we can use to store our datasets as we will look at it in detail, *Chapter 5: Implementation.*

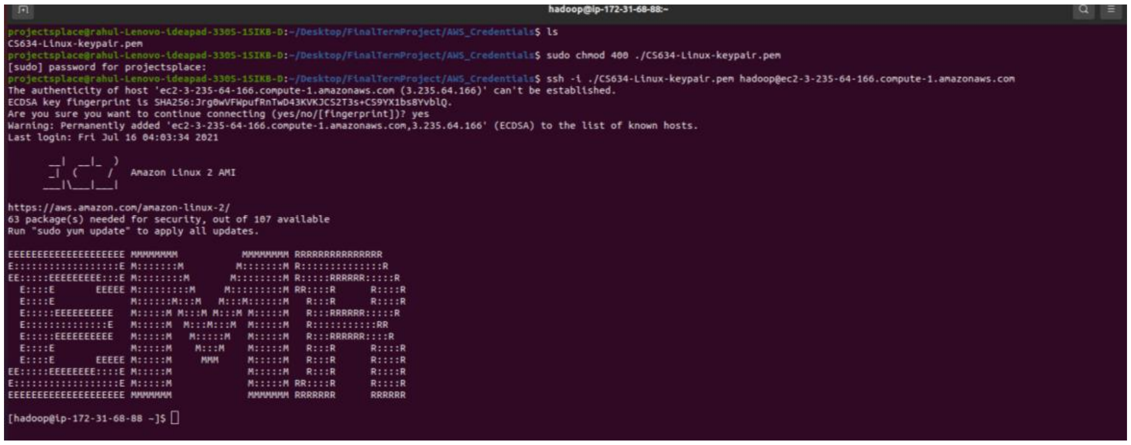
4.4 Setting up the Application Environment (inside Master node of EMR)

We are now done with setting up services inside management console. We will be focusing on using SFTP and SSH to get the application files loaded into EMR and accessing EMR using a remote terminal or Command Prompt. We will also take a brief look Apache Spark, a popular framework for working on large-scale data using clusters that exhibits parallelism in running workloads. This section also presents a tutorial for setting Apache Maven to use Apache Spark API and then go with the setup of AWS SDK/CLI credentials inside of EMR's Master EC2 node.

Using SSH to access EMR master:

We will see how one can use SSH to access EMR master EC2 node system remotely from our computer.

- Since we have already downloaded our key-pair for accessing a EC2, below image show how one can access EMR using the key pair stored in the folder from where the terminal is currently working at. For sake of convenience let name this terminal as, *EMR-master's SSH accessed terminal.*



```
projectsplace@rahul-Lenovo-ideapad-330S-15IKB-D:~/Desktop/FinalTermProject/AWS_Credentials$ ls
CS634-Linux-keypair.pem
projectsplace@rahul-Lenovo-ideapad-330S-15IKB-D:~/Desktop/FinalTermProject/AWS_Credentials$ sudo chmod 400 ./CS634-Linux-keypair.pem
[sudo] password for projectsplace:
projectsplace@rahul-Lenovo-ideapad-330S-15IKB-D:~/Desktop/FinalTermProject/AWS_Credentials$ ssh -i ./CS634-Linux-keypair.pem hadoop@ec2-3-235-64-166.compute-1.amazonaws.com
The authenticity of host 'ec2-3-235-64-166.compute-1.amazonaws.com (3.235.64.166)' can't be established.
ECDSA key fingerprint is SHA256:Jrg0wVFWpufRnTwD43KVKJCS2T3s+CS9YX1bs8YvblQ.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-3-235-64-166.compute-1.amazonaws.com,3.235.64.166' (ECDSA) to the list of known hosts.
Last login: Fri Jul 16 04:03:34 2021

       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
63 package(s) needed for security, out of 107 available
Run "sudo yum update" to apply all updates.

EEEEEEEEEEEEEEEEEEEE MMMMMMMM           MMMMMMMM RRRRRRRRRRRRRRR
E::::::::::::::::::E M:::::::M         M:::::::M R::::::::::::::R
EE:::::EEEEEEEEE:::E M::::::::M       M::::::::M R:::::RRRRRR:::::R
  E::::E       EEEEE M:::::::::M     M:::::::::M RR::::R      R::::R
  E::::E             M::::::M:::M   M:::M::::::M   R:::R      R::::R
  E:::::EEEEEEEEEE   M:::::M M:::M M:::M M:::::M   R:::RRRRRR:::::R
  E::::::::::::::E   M:::::M  M:::M:::M  M:::::M   R:::::::::::RR
  E:::::EEEEEEEEEE   M:::::M   M:::::M   M:::::M   R:::RRRRRR::::R
  E::::E             M:::::M    M:::M    M:::::M   R:::R      R::::R
  E::::E       EEEEE M:::::M     MMM     M:::::M   R:::R      R::::R
EE:::::EEEEEEEE::::E M:::::M             M:::::M   R:::R      R::::R
E::::::::::::::::::E M:::::M             M:::::M RR::::R      R::::R
EEEEEEEEEEEEEEEEEEEE MMMMMMM             MMMMMMM RRRRRRR      RRRRRR

[hadoop@ip-172-31-68-88 ~]$ 
```

Fig 4.20 Accessing EMR master using a Linux/Mac Terminal

We can also do the same inside Windows, although we may require Putty and PuttyGen tools for privately accessing our EMR master node instance. Follow the tutorial at this link for this purpose.

Using SFTP for transferring application files to EMR master:

Now let's focus on transferring the application files to EMR master node. We use SFTP command to do this. We will delve into the folder structure and files in detail in the *Chapter 5: Implementation*.

- Suppose we have the application files in a folder as shown below in *Fig 4.17* and this (project) folder is residing inside of another folder also containing our *AWS access credentials*. Having this pictured, let's proceed with using SSH within the folder containing the AWS access credential and project folder.

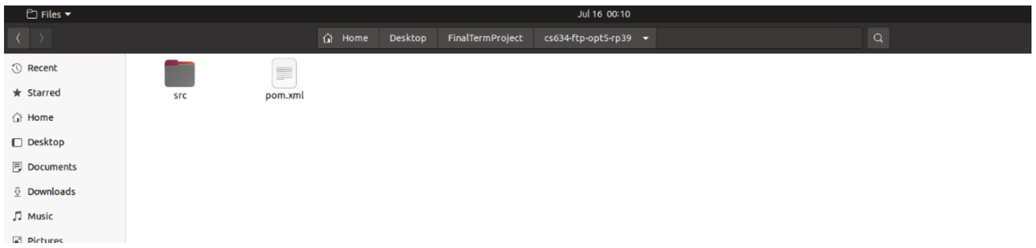


Fig 4.21 Folder containing application files

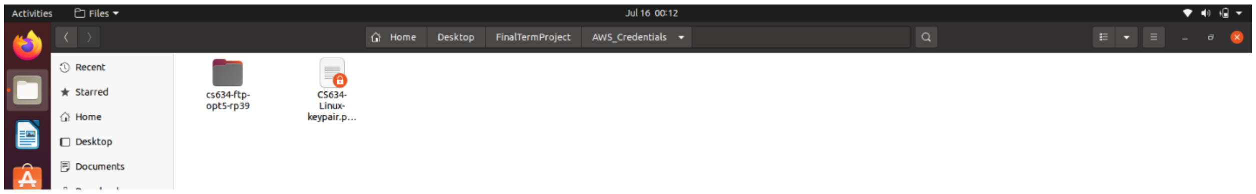
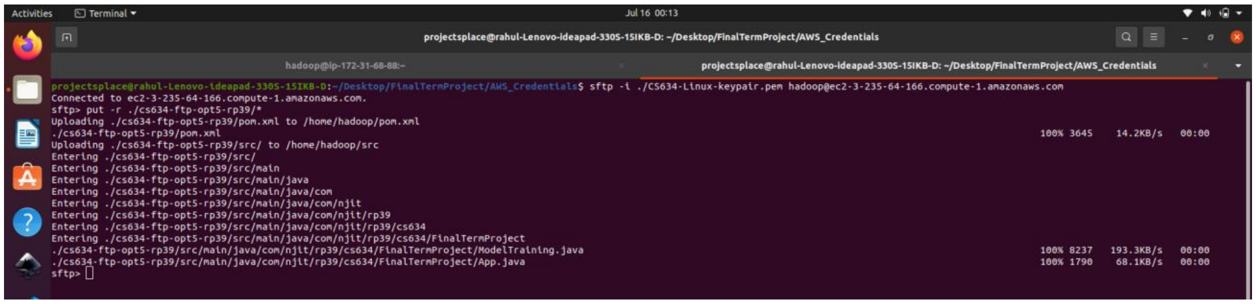


Fig 4.22 Folder containing AWS access credentials

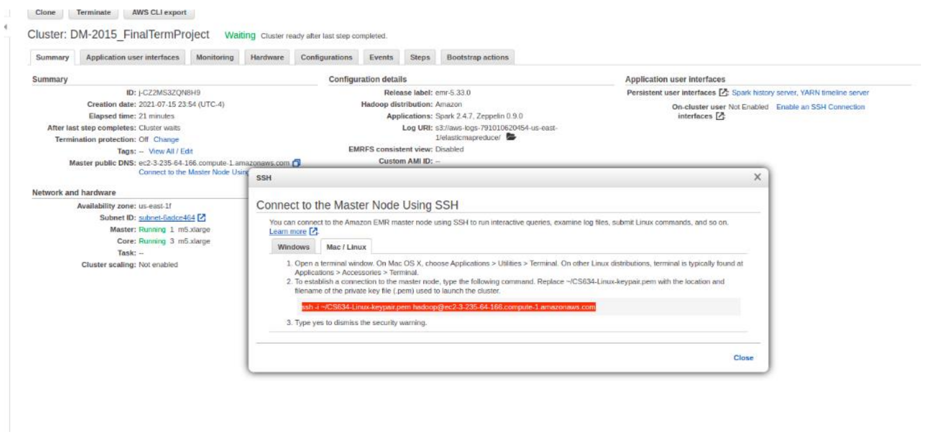

Activities
Terminal
Jul 16 00:13
projectsplace@rahul-Lenovo-ideapad-330S-15IKB-D: ~/Desktop/FinalTermProject/AWS_Credentials
hadoop@ip-172-31-68-88:~

```
projectsplace@rahul-Lenovo-ideapad-330S-15IKB-D:~/Desktop/FinalTermProject/AWS_Credentials$ sftp -i ./CS634-Linux-keypair.pem hadoop@ec2-3-235-64-166.compute-1.amazonaws.com
Connected to ec2-3-235-64-166.compute-1.amazonaws.com.
sftp> put -r ./cs634-ftp-opt5-rp39/*
Uploading ./cs634-ftp-opt5-rp39/pom.xml to /home/hadoop/pom.xml
./cs634-ftp-opt5-rp39/pom.xml                                                    100% 3645    14.2KB/s   00:00
Uploading ./cs634-ftp-opt5-rp39/src/ to /home/hadoop/src
Entering ./cs634-ftp-opt5-rp39/src/
Entering ./cs634-ftp-opt5-rp39/src/main
Entering ./cs634-ftp-opt5-rp39/src/main/java
Entering ./cs634-ftp-opt5-rp39/src/main/java/com
Entering ./cs634-ftp-opt5-rp39/src/main/java/com/njit
Entering ./cs634-ftp-opt5-rp39/src/main/java/com/njit/rp39
Entering ./cs634-ftp-opt5-rp39/src/main/java/com/njit/rp39/cs634
Entering ./cs634-ftp-opt5-rp39/src/main/java/com/njit/rp39/cs634/FinalTermProject
./cs634-ftp-opt5-rp39/src/main/java/com/njit/rp39/cs634/FinalTermProject/ModelTraining.java   100% 8237   193.3KB/s   00:00
./cs634-ftp-opt5-rp39/src/main/java/com/njit/rp39/cs634/FinalTermProject/App.java             100% 1790    68.1KB/s   00:00
sftp> 
```

Fig 4.23 Transferring file to EMR master node

- Opening the terminal in the folder similar to the one presented in *Fig 4.18*, use the commands shown in the figure above to access the SFTP. Note that we may have to change the IP-address as allotted by AWS, shown in the AWS EMR page.

If you are not able to find the IP-address, one quick way to find is to visit the EMR cluster page and click on *Connect to the Master Node Using SSH* link. This opens a dialog box, this also shows the IP-address along with the user-name of the EMR-master EC2 node, which are collectively used to access via SFTP or SSH.



Cluster: DM-2015_FinalTermProject Waiting Cluster ready after last step completed.

Summary
ID: j-CZ2MS3ZQN8H9
Creation date: 2021-07-15 23:54 (UTC-4)
Elapsed time: 21 minutes
After last step completes: Cluster waits
Termination protection: Off Change
Tags: -- View All / Edit
Master public DNS: ec2-3-235-64-166.compute-1.amazonaws.com
Connect to the Master Node Using

Network and hardware
Availability zone: us-east-1f
Subnet ID: subnet-6adce464
Master: Running 1 m5.xlarge
Core: Running 3 m5.xlarge
Task: --
Cluster scaling: Not enabled

Configuration details
Release label: emr-5.33.0
Hadoop distribution: Amazon
Applications: Spark 2.4.7, Zeppelin 0.9.0
Log URI: s3://aws-logs-791010620454-us-east-1/elasticmapreduce/
EMRFS consistent view: Disabled
Custom AMI ID: --

Application user interfaces
Persistent user interfaces: Spark history server, YARN timeline server
On-cluster user interfaces: Not Enabled Enable an SSH Connection

SSH

Connect to the Master Node Using SSH

You can connect to the Amazon EMR master node using SSH to run interactive queries, examine log files, submit Linux commands, and so on. Learn more

1. Open a terminal window. On Mac OS X, choose Applications > Utilities > Terminal. On other Linux distributions, terminal is typically found at Applications > Accessories > Terminal.
2. To establish a connection to the master node, type the following command. Replace ~/CS634-Linux-keypair.pem with the location and filename of the private key file (.pem) used to launch the cluster.

```
ssh -i ~/CS634-Linux-keypair.pem hadoop@ec2-3-235-64-166.compute-1.amazonaws.com
```

3. Type yes to dismiss the security warning.

Fig 4.24 Connect to the Master Node Using SSH

Now, in order for our instances to use any AWS resources using CLI, there are 2 ways to setup either by using Roles or by setting access credentials. Setting by Roles is out of scope for the project, therefore, let's focus on much a simpler way, i.e., by using AWS SDK/CLI access credentials. Please follow the steps below:

- Using the search bar, go to *IAM > Users tab (left tab menu) > Create a user* with at least access to EMR. (For the sake of ease for the project, you can a lot *AdministratorAccess* or role with less privilege that give access to EMR and S3 resources in AWS).

- After user is created, within the user select *Security Credentials tab > Create Access key*
- Download the csv file and proceed to the *EMR-master's SSH accessed terminal. Input the following commands.* For more details in the process, please visit this link.

```
$ mkdir .aws
$ vi ./.aws/credentials
```

Paste your access key credentials presented inside the csv to this file as shown below, and save it.

```
[default]
aws_access_key_id=AKIAIOSFODNN7EXAMPLE
aws_secret_access_key=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
```

Make sure to substitute you own credentials details for the above. We are done with the configuration.

About Apache Spark

Apache Spark is an open-source unified analytics engine for large-scale data processing. Spark provides an interface for programming entire clusters with implicit data parallelism and fault tolerance [9]. If we are to build a program using Apache Spark, the easiest way is to use Apache Maven. To install Apache Maven please follow the steps below:

- Within our *EMR-master's SSH accessed terminal,* execute following steps to install maven and setup project files:

```
# Linux-step by step setup for Apache Maven
$ cd ~
$ wget https://mirrors.ocf.berkeley.edu/apache/maven/maven-3/3.6.3/binaries/apache-maven-
3.6.3-bin.tar.gz
$ tar xvf apache-maven-3.6.3-bin.tar.gz
$ sudo mv apache-maven-3.6.3  /usr/local/apache-maven
$ export M2_HOME=/usr/local/apache-maven
$ export M2=$M2_HOME/bin
$ export PATH=$M2:$PATH
$ mvn -version
```

This will install Apache Maven inside of EMR master. The reason we are doing this is to create and organize our application files and finally let it build into a *.jar* file, getting us to producing the full-fledge application. This build is later submitted to Apache Spark, which is already preinstalled due to our choice made previously during creation of EMR cluster.

We are also going to use two popular Apache Spark's built-in libraries, namely Apache Spark ML library and Apache Spark SQL. All of the details of installation of these are already defined in Maven dependency management's pom.xml file supplied in section named *Miscellaneous* of *Chapter 6: Implementation*.

General Warning: Things to NOTE while working on AWS

While working in AWS we must always beware of cost billing associated with resources we use and also those we keep alive after requirement is fulfilled. It is always recommended to terminate, delete or clean up all resource that we have issued on AWS after the requirement and necessities are fulfilled.

Now we proceed to the next chapter which presents the details on setting up Dataset in S3 and demonstrating the code-wise implementations.

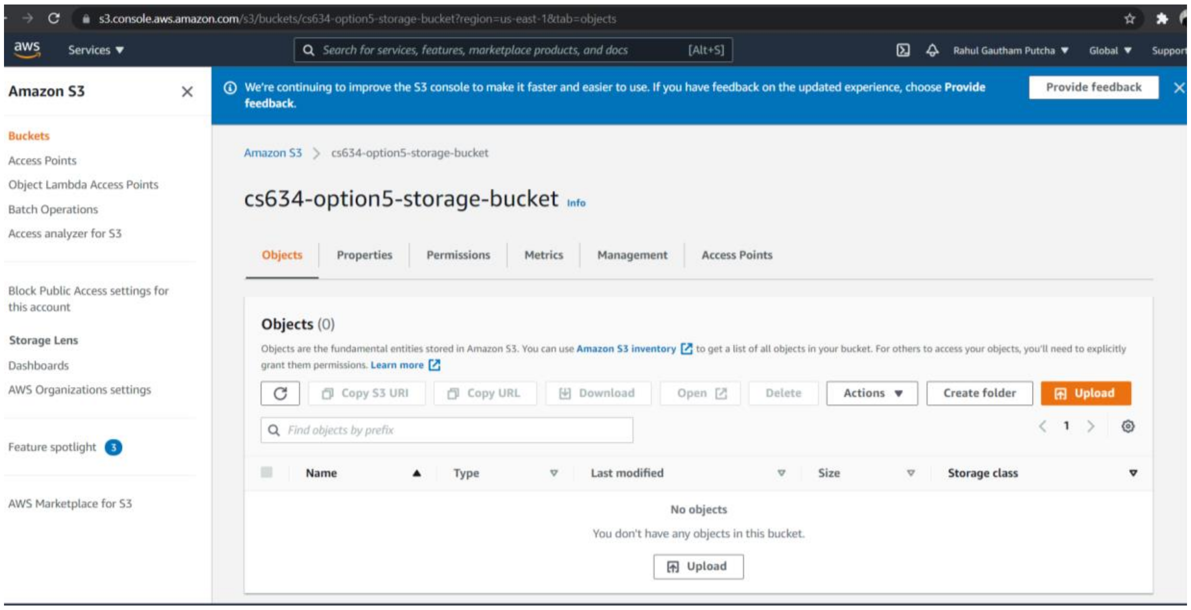
Chapter 5. Implementation

Now that we have done the setup our environment on AWS cloud, we can go into the details on how our application is implemented on *Apache Spark in Java*. First, we will be looking at how we can setup our dataset on S3 and then move on to implementation for parallel machine learning for solving Breast cancer at Wisconsin's problem.

5.1 Details on the setting up Dataset in S3

In this section we will setup our files on S3 in a systematic manner.

- Download the dataset for Breast Cancer Wisconsin from Kaggle.
- Go to S3 page in your Management Console
- Open the S3 bucket we have created in, *4.3 Setting up a S3 (Simple Storage Service) on AWS*


We're continuing to improve the S3 console to make it faster and easier to use. If you have feedback on the updated experience, choose Provide feedback.

Amazon S3 > cs634-option5-storage-bucket

cs634-option5-storage-bucket

Objects (0)

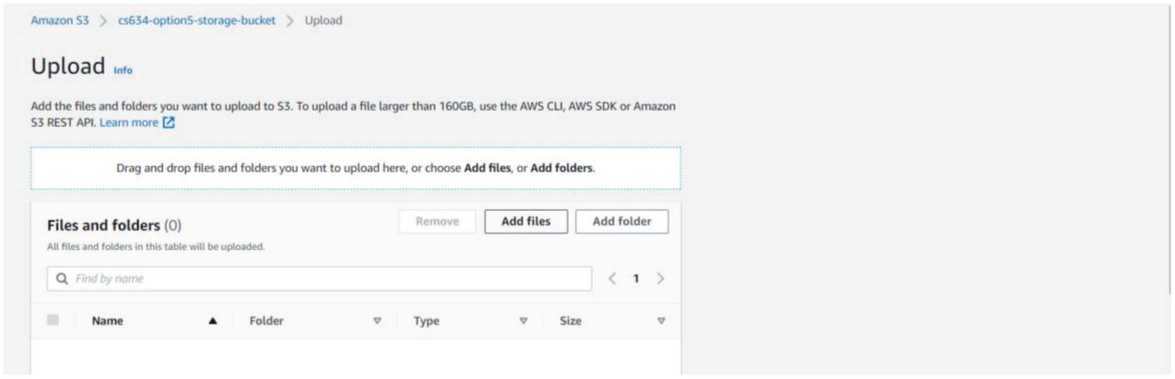
Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

No objects

You don't have any objects in this bucket.

Fig 5.1 Initial Object Listing in S3 Bucket

- Click on Upload button. An Upload form appears, Click on *Add files* and choose the dataset file.


Amazon S3 > cs634-option5-storage-bucket > Upload

Upload

Add the files and folders you want to upload to S3. To upload a file larger than 160GB, use the AWS CLI, AWS SDK or Amazon S3 REST API. Learn more

Drag and drop files and folders you want to upload here, or choose Add files, or Add folders.

Files and folders (0)

All files and folders in this table will be uploaded.

Fig 5.2 Uploading a file on S3 (1)

- Before we submit the form after choosing the dataset, the destination section shows the URI link that will be allotted to the resource we are currently submitting, as shown in *Fig 5.2.*
- Do copy this link as we are going to use it as our argument to our Random Forest Training program that's using Apache Spark.

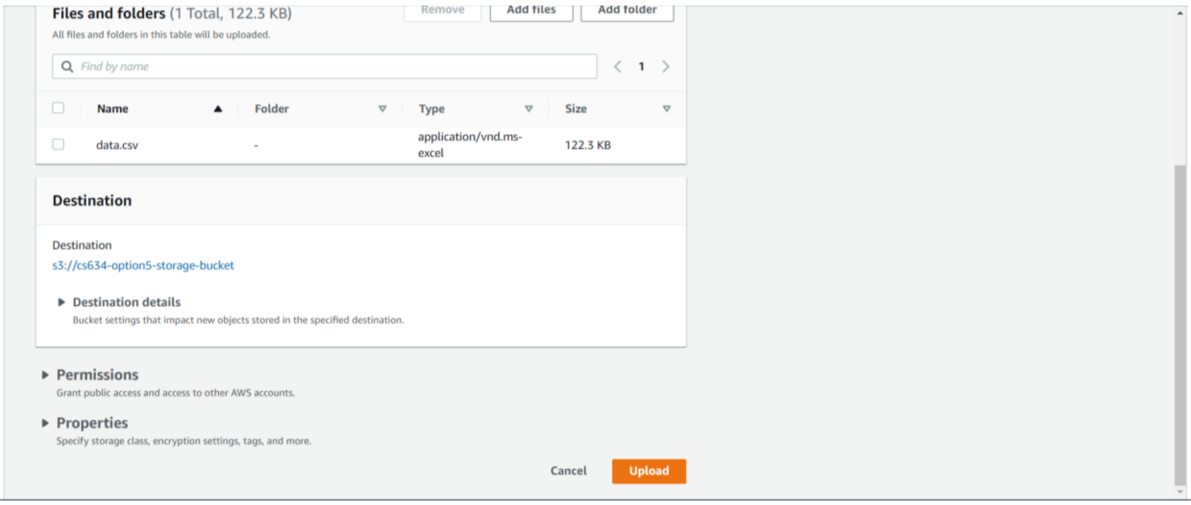
  Note: A sample link is shown below,

  s3://cs634-option5-storage-bucket

  If we want to access a specific file for this bucket, eg: data.csv file which we have recently uploaded, set the prefix and as above link and suffix as the file name as shown below.

  s3://cs634-option5-storage-bucket/data.csv

- Finally, hit the *Upload button* at the bottom of the form.


Files and folders (1 Total, 122.3 KB)

All files and folders in this table will be uploaded.

Name	Folder	Type	Size
data.csv	-	application/vnd.ms-excel	122.3 KB

Destination

Destination
s3://cs634-option5-storage-bucket

Destination details
Bucket settings that impact new objects stored in the specified destination.

Permissions
Grant public access and access to other AWS accounts.

Properties
Specify storage class, encryption settings, tags, and more.

Fig 5.3 Uploading a file on S3 (2)

5.2 Folder Structure

Below figure show the project folder structure details. All the implementation file details are presented in the later sections. Note that *src/main/java/com/njit/rp39/cs634/FinalTermProject* are all individual folders, i.e, src contains main, main contains java, java contains com folder and so on. The original reason for the structure was for maintaining the project as a package, where *com.njit.rp39.cs634.FinalTermProject* is the package our application belongs to. The screenshot below is taken from Visual Studio code.

Since we are done installing with Apache Maven on to EMR from the previous chapter, we will now move on to setting up our project file by build the application structure using files given in the next few sections, all from scratch by referring to the folder structure given below in next page.

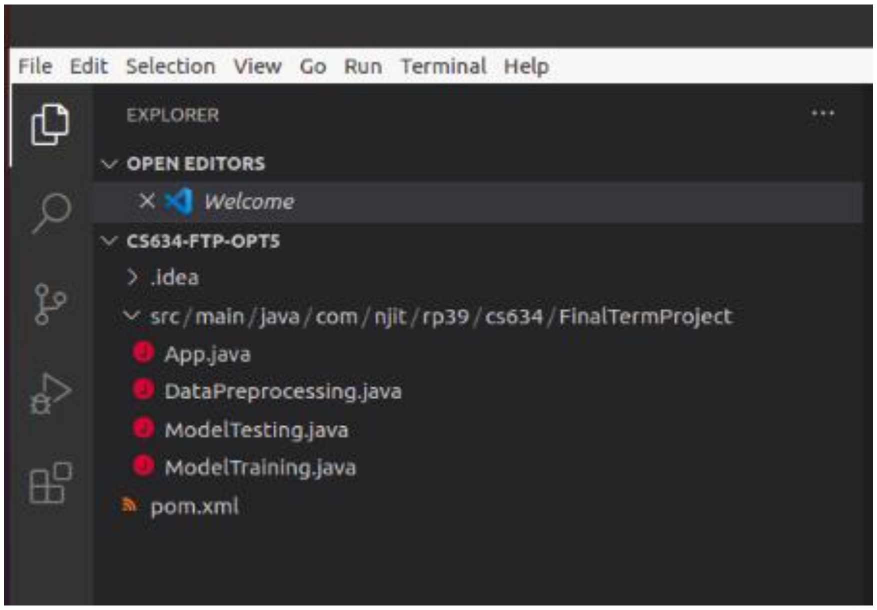


Fig 5.4 Application Folder structure

After integrating all the file in this chapter, please re-review the section 4.4 Setting up the Application Environment (inside Master node of EMR) from Chapter 4: Setting up the Environment. Below is the first piece of the code, belonging to *App.java,* which is going to be the entry point for our spark program.

App .java:

The piece of code below is the entry point for our program, whenever it runs. It gives user an option to decide whether to train/test, run on cluster/thread mode, where the save point for our model and test.csv file be, what dataset we are going to use and what is the label part of our dataset which we want to classify. All of the details above are taken as program argument (i.e., as a command line argument).

```java
package com.njit.rp39.cs634.FinalTermProject;
/**
* Final Term Project
* Author: Rahul Gautham Putcha
* Details:
* - Implement the Data Mining Algorithm on Apache Spark on AWS Elastic MapReduce EMR cluster
**/

import org.apache.log4j.Level;
import org.apache.spark.sql.SparkSession;
import org.apache.log4j.Logger;

/**
* Class: App
* Accepts: Command Line argument for
* Details: Main Application. Conducts Training/Testing
**/
public class App {

  public static void main( String[] args ) {
     SparkSession session;
     Logger.getLogger("org.apache").setLevel(Level.WARN);

     if(args.length==5) {
            if(args[1].compareTo("cluster") == 0) {
                session = SparkSession.builder()
                        .appName("Final Term Project: ")
                        .getOrCreate();
            } else {
                session = SparkSession.builder()
                        .appName("Final Term Project: ")
                        .master("local[*]")
                        .getOrCreate();
            }


            if(args[0].compareTo("train") == 0) {
                new ModelTraining(session, args[3], args[2], args[4]);
            } else if(args[0].compareTo("test") == 0) {
                new ModelTesting(session, args[2], args[3], args[4]);
            } else {
                System.out.println(
                        "[ERROR]: CLI SYNTAX ERROR\n" +
                        "[INFO]: CLI SYNTAX arg: " +
                        "{0:train / test} {1:cluster / thread} {2:save/load point location} " +
                        "{3:dataset filepath} {4:label}"
                );
            }

        } else{
            System.out.println(
                    "[ERROR]: CLI SYNTAX ERROR\n" +
                    "[INFO]: CLI SYNTAX arg: " +
                    "{0:train / test} {1:cluster / thread} {2:save/load point location} " +
                    "{3:dataset filepath} {4:label}"
            );
        }
    }
}
```

Code 5.1 Code for App.java

Data Preprocessing .java

Below code is shared between both ModelTesting.java and ModelTraining.java. This piece of code does all Data preprocessing: Data Cleaning, Data Imputation, Normalization etc., before we send it over to Training or Testing phase.

```
package com.njit.rp39.cs634.FinalTermProject;

// Import Files (Apache Spark)
import org.apache.spark.ml.feature.Imputer;
import org.apache.spark.ml.feature.Normalizer;
import org.apache.spark.ml.feature.VectorAssembler;

import org.apache.spark.sql.Column;
import org.apache.spark.sql.Dataset;
import org.apache.spark.sql.Row;
import static org.apache.spark.sql.functions.col;

/**
* Class: Data Preprocessing
* Details: Contains Common utilities for Data Preprocessing (used in Training and Testing phase)
**/
public class DataPreprocessing {
  protected Dataset<Row> feature, target;
  protected String[] getFeatureNames(Dataset<Row> data, String label){
    /**
     * Function: getFeatureNames
     * Argument:
     * Dataset<Row> data: Spark SQL dataset with a Header row
     * String label: Label name present in dataset's header row
     * Details:
     * Parse Header row from a given dataset and disregards label from features.
     * Returns: String[] list_of_features in the dataset `data`
     **/

    String[] columnName = data.schema().fieldNames();
    int j = 0, columnLength = 0;
    for (String s : columnName) {
      if (s.compareTo(label) != 0 && s.charAt(0) != '_' && s.compareTo("id")!=0) {
        columnLength++;
      }
    }
    String[] featureCol = new String[columnLength];
    for (String s : columnName) {
      if (s.compareTo(label) != 0 && s.charAt(0) != '_' && s.compareTo("id")!=0) {
        featureCol[j++] = s;
      }
    }
    return featureCol;
  }


  protected Dataset<Row> getFeatures(Dataset<Row> data, String label){
    /**
     * Function: getFeatures
     * Argument:
     * Dataset<Row> data: a Dataset with features and label
     * String label: label name present as column header in the dataset `data`
     * Details:
     * - Removes label column inside of dataset `data`
     * Returns: Dataset<Row> feature_columns (...excludes label_column)
     **/
    String[] featureCols = getFeatureNames(data, label);
    VectorAssembler vectorAssembler = new VectorAssembler();
    vectorAssembler.setInputCols(featureCols);
    vectorAssembler.setOutputCol("features");

    Column[] featureColumns =  { col("id"), col("features") };

    return vectorAssembler.transform(data).select(featureColumns);
  }

  protected Dataset<Row> cleanAndImputeData(Dataset<Row> data, String[] columnNames, String label){
    /**
     * Function: cleanAndImputeData
     * Argument:
     * Dataset<Row> data: a Dataset with features and label
     * String[] columnNames: List of columns names
     * String label: label name present as column header in the dataset `data`
     * Details:
     * - Clean dataset by removing columns with over 30% of NULL values and Impute dataset (... on features only)
     * - Hence, excludes `label` column in dataset `data`
     * Returns: Dataset<Row> cleaned_dataset
     **/

    System.out.println("[INFO]: Cleaning data.");

    // Cleaning Data
    for (String s: columnNames) {
      if( data.where(data.col(s).isNull()).count() > data.count()*(30.0f/100.0f) ){
```

```
                System.out.println("[INFO]: Dropped column "+s+" for >30% null values.");
                data = data.drop(s);
            }
        }

        //Impute Data
        columnNames = getFeatureNames(data,label);
        data = (new Imputer()
                .setInputCols(columnNames)
                .setOutputCols(columnNames)
                .setStrategy("mean")).fit(data).transform(data);

        return data;
    }
    protected void preprocessData(Dataset<Row> dataset, String label){
        /**
         * Function: preprocessData
         * Argument:
         * Dataset<Row> data: a Dataset with features and label
         * String label: label name present as column header in the dataset `data`
         * Details: (Basic)
         * * 1. Clean Null Population        : Done
         * * 2. Imputation                   : Done
         * * 3. Normalization                : Done
         * After Execution: (Result)
         * - Stores `this.feature` variable
         * - Stores `this.target` variable
         **/


        Dataset<Row> cleanedData= cleanAndImputeData(dataset, dataset.schema().fieldNames() ,label);

        // If necessary, drop other
        //for (String s:manualDroppedColumns) { cleanedData = cleanedData.drop(s); }
        //cleanedData.show(5);

        // Getting features from Dataset
        Dataset<Row> features = getFeatures(cleanedData, label);

        // Normalization
        Normalizer normalizer = new Normalizer()
                .setInputCol("features")
                .setOutputCol("normFeatures")
                .setP(1.0);
        Column[] featureColumns =  { col("id"), col("normFeatures") };
        Dataset<Row> normData = normalizer.transform(features).select(featureColumns);

        // Storing individual feature and target to class member vaiables
        Column[] targetColumns =  { col("id"), col(label) };
        feature = normData;
        target = dataset.select(targetColumns);

    }
}
```

Code 5.2 Code for DataPreprocessing.java

5.3 Code for Random Forest (RF) Classification training on Apache Spark

Below is the code used for the purpose of training our model. This code loads the dataset and splits the dataset into 2 subsets, Training and testing. The testing dataset is saved at the save point entered as argument into our App and the training dataset is used for building our model. Before we train the model using the training dataset, we preprocess the data using function inherited from DataPreprocessing class.

After the training is complete, we save the model to the same save point location where we are storing the test file.

<u>ModelTraining.java</u>

```
package com.njit.rp39.cs634.FinalTermProject;

// Import Files (Apache Spark)
import org.apache.spark.ml.Pipeline;
import org.apache.spark.ml.PipelineStage;
import org.apache.spark.ml.classification.RandomForestClassifier;
import org.apache.spark.ml.evaluation.BinaryClassificationEvaluator;
import org.apache.spark.ml.feature.StringIndexer;
import org.apache.spark.ml.param.ParamMap;
import org.apache.spark.ml.tuning.CrossValidator;
import org.apache.spark.ml.tuning.CrossValidatorModel;
import org.apache.spark.ml.tuning.ParamGridBuilder;

import org.apache.spark.sql.SparkSession;
import org.apache.spark.sql.Dataset;
import org.apache.spark.sql.Row;
import static org.apache.spark.sql.functions.col;
import org.apache.spark.sql.Column;

import java.io.IOException;


/**
* Class: Model Training
* Details: Conducts Training on Dataset using Random Forest Model using Apache Spark
**/
public class ModelTraining extends DataPreprocessing{
  String[] manualDroppedColumns;

  public ModelTraining(SparkSession session, String datasetLoc, String saveLoc, String label){
     /**
      * Function: ModelTraining (Constructor)
      * Argument:
      * SparkSession session: A spark session for working with Spark utilities
      * String datasetLoc: Location of the initial dataset (containing Training+Testing dataset)
      * String saveLoc: A Save point location for saving the test dataset (test.csv) and final trained model (rf_model.model)
      * String label: Label name belonging to the dataset present at location `datasetLoc`
      * Details:
      * - Loads dataset into spark,
      * - Splits dataset into random train & test,
      * - Conducts Data-Preprocessing, Normalization
      * - Trains model
      * * All tasks are conducted over cluster with parallelism using Apache Spark
      * Dataset requirement (Assumptions):
      * - Works on any dataset with a mandatory `id` column
      * - must have at least one feature columns (other than `id`)
      * - must have at least one label column
      **/

     // Manual Editing columns
     manualDroppedColumns = new String[] {};

     // Fetching Entire Dataset/CSV file from
     Dataset<Row> data = session.read()
           .option("header", true)
           .option("inferSchema", true)
           .csv(datasetLoc);
     data.show(5);
```

```
    // Splitting Dataset to Train-Test set Split here ...
    Dataset<Row>[] splits = data.randomSplit(new double[] {0.8, 0.2}, 12345);
    Dataset<Row> train = splits[0];
    Dataset<Row> test = splits[1];

    // Saving Testing set to use it later while Running Program with TestingModel.java
    try {
      test.write().format("csv")
            .option("inferSchema", "true")
            .option("header", "true")
            .csv(saveLoc+"/test.csv");
    }catch(Exception e){
      System.out.println("[Error]: Couldn't save the test split file to location - "+saveLoc+"/test.csv");
      System.out.println(e.getMessage());
    }

    // Data Preprocessing (Basic)
    preprocessData(train, label);

    System.out.println("Training set:\nFeatures:");
    feature.show(3);

    System.out.println("Target(Labels):");
    target.show(3);

    // Splitting Dataset to into Features and Labels: Generating Training Dataset
    Column[] colNames = {col("normFeatures"), col("diagnosis")};
    Dataset<Row> train_set = feature
      .join(target, feature.col("id").equalTo( target.col("id") ))
      .select(colNames)
      .withColumnRenamed("normFeatures","features")
      .withColumnRenamed(label,"label");
    StringIndexer stringIndexer = new StringIndexer().setInputCol("label").setOutputCol("labelIndex");

    Column[] trainCol = {col("features"), col("labelIndex")};
    train_set = stringIndexer.fit(train_set)
        .transform(train_set).select(trainCol)
        .withColumnRenamed("labelIndex", "label");

    System.out.println("Final training set:");
    train_set.show(3);

    // Start Training a model on the Training Dataset
    train(train_set, saveLoc);
  }

  private void train(Dataset<Row> train, String saveLocation){
    /**
     * Function: train
     * Argument:
     * Dataset<Row> data: a Dataset with features and label
     * String saveLocation: save point for saving the model
     * Details:
     * 1. Model Selection: Random Forest (RF) Classifier
     * 2. Model Training and Cross Validation
     * 4. Best Model chosen using areaUnderROC metric
     * 3. Save Model for future testing and predictions
     * After Execution: (Result)
     * - Best Model will be saved at save point `saveLocation`
     **/

    // Training
    RandomForestClassifier rf = new RandomForestClassifier();

    //Iterable<String> impurity = Arrays.asList(new String[]{"gini", "entropy"});
    Pipeline pipeline = new Pipeline().setStages(new PipelineStage[] {rf});
    ParamMap[] paramMap = new ParamGridBuilder()
          .addGrid(rf.maxDepth(), new int[]{1,2,5, 10, 15})
          .addGrid(rf.minInstancesPerNode(), new int[]{1, 2, 4,5,10})
          .build();

    // We now treat the Pipeline as an Estimator, wrapping it in a CrossValidator instance.
    // A CrossValidator requires an Estimator, a set of Estimator ParamMaps, and an Evaluator.
    // Note that the evaluator here is a BinaryClassificationEvaluator and its default metric
    // is areaUnderROC.
    CrossValidator cv = new CrossValidator()
          .setEstimator(pipeline)
          .setEstimatorParamMaps(paramMap)
          .setEvaluator(new BinaryClassificationEvaluator())
          .setNumFolds(10) // 10-Fold validation
          .setParallelism(4); // 4 setting in parallel

    // Run cross-validation, and choose the best set of parameters.
    CrossValidatorModel cvModel = cv.fit(train);

    try {
      cvModel.save(saveLocation+"/rf_model.model");
      System.out.println("[INFO]: Model Trained Successfully");
    }catch(IOException e){
      System.out.println("[Error]: Couldn't Write model to location - "+saveLocation+"/rf_model.model");
    }

  }
}
```

Code 5.3 Code for TrainingModel.java

5.4 Code for Testing the RF model for solving Breast-Cancer Problem

Below is the code use for testing purpose. Here we take in location of the model we have saved at end of training phase, aka, load point, hence we load the model in to our application and then make Spark use it to predict and test over the testing dataset. We also accept the testing dataset location as an individual argument separately. By end of the test predication conducted, we showcase the model performance by conducting evaluation using *F1-score*, *Accuracy score* (internally performed by means of using *Confusion Matrix*) and lastly, by using *Area under ROC* metric.

ModelTesting.java

```
package com.njit.rp39.cs634.FinalTermProject;

// Import Files (Apache Spark)
import org.apache.spark.ml.evaluation.BinaryClassificationEvaluator;
import org.apache.spark.ml.evaluation.MulticlassClassificationEvaluator;
import org.apache.spark.ml.feature.StringIndexer;
import org.apache.spark.ml.tuning.CrossValidatorModel;

import org.apache.spark.sql.SparkSession;
import org.apache.spark.sql.Dataset;
import org.apache.spark.sql.Row;
import static org.apache.spark.sql.functions.col;
import org.apache.spark.sql.Column;



/**
* Class: Model Testing
* Details: Conducts Testing on Dataset using Random Forest Model using Apache Spark
**/
public class ModelTesting extends DataPreprocessing{
  String[] manualDroppedColumns;

  public ModelTesting(SparkSession session, String loadLoc, String datasetLoc , String label){
    /**
     * Function: ModelTesting (Constructor)
     * Argument:
     * SparkSession session: A spark session for working with Spark utilities
     * String datasetLoc: Location of the test dataset (Testing dataset only)
     * String loadLoc: A Load point location holding the final trained model (rf_model.model)
     * String label: Label name belonging to the dataset present at location `datasetLoc`
     * Details:
     * - Loads test dataset into spark,
     * - Conducts Data-Preprocessing, Normalization
     * - Test and Evaluates the model
     * * All tasks are conducted over cluster with parallelism using Apache Spark
     * Dataset requirement (Assumptions):
     * - Works on any dataset with a mandatory `id` column
     * - must have at least one feature columns (other than `id`)
     * - must have at least one label column
     **/

    System.out.println("loadLoc:"+loadLoc);
    System.out.println("datasetLoc:"+datasetLoc);
    // Manual Editing columns
    manualDroppedColumns = new String[] {};

    // Fetching Entire Dataset/CSV file from
    Dataset<Row> test = session.read()
          .option("header", true)
          .option("inferSchema", true)
          .csv(datasetLoc);
    test.show(5);


    // Data Preprocessing (Basic)
    preprocessData(test, label);

    System.out.println("Testing set:\nFeatures:");
    feature.show(3);

    System.out.println("Target(Labels):");
    target.show(3);

    // Splitting Dataset to into Features and Labels: Generating Training Dataset
    Column[] colNames = {col("normFeatures"), col("diagnosis")};
    Dataset<Row> test_set = feature
          .join(target, feature.col("id").equalTo( target.col("id") ))
          .select(colNames)
          .withColumnRenamed("normFeatures","features")
          .withColumnRenamed(label,"label");
    StringIndexer stringIndexer = new StringIndexer().setInputCol("label").setOutputCol("labelIndex");

    Column[] testCol = {col("features"), col("labelIndex")};
    test_set = stringIndexer.fit(test_set)
          .transform(test_set).select(testCol)
          .withColumnRenamed("labelIndex", "label");

    System.out.println("Final Test set (After preprocessing):");
    test_set.show(3);

    // Start Testing a model on the Testing Dataset
    test(test_set, loadLoc);
  }

  private void test(Dataset<Row> test, String loadLocation){
    /**
     * Function: test
     * Argument:
     * Dataset<Row> test: a Dataset with features and label
     * String loadLocation: load point for fetching the pre-trained RF model
     * Details:
     * 1. Loads the Best Pre-Trained model
     * 2. Does Prediction on the Test dataset
     * 4. Evaluates using different metrics:
     * * - Accuracy
     * * - F1-score, and
     * * - Area under ROC metric
     **/

    // Testing
    CrossValidatorModel cvModel = CrossValidatorModel.load(loadLocation+"/rf_model.model");

    // Make predictions on test documents. cvModel uses the best model found.
    Dataset<Row> predictions = cvModel.transform(test);

    MulticlassClassificationEvaluator evaluator = new MulticlassClassificationEvaluator()
          .setLabelCol("label")
          .setPredictionCol("prediction")
          .setMetricName("accuracy");
    System.out.println("Accuracy: "+evaluator.evaluate(predictions));
    evaluator = new MulticlassClassificationEvaluator()
          .setLabelCol("label")
          .setPredictionCol("prediction")
          .setMetricName("f1");
    System.out.println("F1-score: "+evaluator.evaluate(predictions));

    System.out.println("Area under ROC metric: "+new BinaryClassificationEvaluator().evaluate(predictions));

  }
}
```

Code 5.4 Code for TestingModel.java

5.5 Miscellaneous

Below code is also a crucial part of our application as it is used to perform the build using Maven and ultimately used to create the functional & independent .jar file, which we use to submit jobs to Apache Spark. The pom.xml file showcases the dependencies that our application need, to properly handle the execution. Apache Maven use this file to install all dependencies from the popular Maven repository.

pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
 <modelVersion>4.0.0</modelVersion>

 <groupId>com.njit.rp39.cs634.FinalTermProject</groupId>
 <artifactId>cs634-ftp-opt5</artifactId>
 <version>1.0-SNAPSHOT</version>

 <name>cs634-ftp-opt5</name>
```

```xml
<properties>
  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  <maven.compiler.source>1.7</maven.compiler.source>
  <maven.compiler.target>1.7</maven.compiler.target>
</properties>

<dependencies>
  <dependency>
    <groupId>org.apache.spark</groupId>
    <artifactId>spark-core_2.12</artifactId>
    <version>3.1.0</version>
  </dependency>
  <dependency>
    <groupId>org.apache.spark</groupId>
    <artifactId>spark-sql_2.12</artifactId>
    <version>3.1.0</version>
  </dependency>
  <dependency>
    <groupId>org.apache.hadoop</groupId>
    <artifactId>hadoop-hdfs</artifactId>
    <version>3.2.1</version>
  </dependency>
  <dependency>
    <groupId>org.apache.hadoop</groupId>
    <artifactId>hadoop-core</artifactId>
    <version>1.2.1</version>
  </dependency>
  <dependency>
    <groupId>org.apache.spark</groupId>
    <artifactId>spark-mllib_2.12</artifactId>
    <version>3.1.2</version>
  </dependency>
  <dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.11</version>
    <scope>test</scope>
  </dependency>
</dependencies>

<build>
  <pluginManagement><!-- lock down plugins versions to avoid using Maven defaults (may be moved to parent pom) -->
    <plugins>
      <!-- clean lifecycle, see https://maven.apache.org/ref/current/maven-core/lifecycles.html#clean_Lifecycle -->
      <plugin>
        <artifactId>maven-clean-plugin</artifactId>
        <version>3.1.0</version>
      </plugin>
      <!-- default lifecycle, jar packaging: see https://maven.apache.org/ref/current/maven-core/default-
bindings.html#Plugin_bindings_for_jar_packaging -->
      <plugin>
        <artifactId>maven-resources-plugin</artifactId>
        <version>3.0.2</version>
      </plugin>
      <plugin>
        <artifactId>maven-compiler-plugin</artifactId>
        <version>3.8.0</version>
      </plugin>
      <plugin>
        <artifactId>maven-surefire-plugin</artifactId>
        <version>2.22.1</version>
      </plugin>
      <plugin>
        <artifactId>maven-jar-plugin</artifactId>
```

```
          <version>3.0.2</version>
        </plugin>
        <plugin>
          <artifactId>maven-install-plugin</artifactId>
          <version>2.5.2</version>
        </plugin>
        <plugin>
          <artifactId>maven-deploy-plugin</artifactId>
          <version>2.8.2</version>
        </plugin>
        <!-- site lifecycle, see https://maven.apache.org/ref/current/maven-core/lifecycles.html#site_Lifecycle -->
        <plugin>
          <artifactId>maven-site-plugin</artifactId>
          <version>3.7.1</version>
        </plugin>
        <plugin>
          <artifactId>maven-project-info-reports-plugin</artifactId>
          <version>3.0.0</version>
        </plugin>
      </plugins>
    </pluginManagement>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-compiler-plugin</artifactId>
        <configuration>
          <source>8</source>
          <target>8</target>
        </configuration>
      </plugin>
    </plugins>
  </build>
</project>
```

Code 5.5 Code for pom.xml

We now also look at the code for Random Forest Classifier that Spark uses to build the Random Forest Classifier model. As described in the *Chapter 2: Focusing on the algorithms,*

```
/*
 * Licensed to the Apache Software Foundation (ASF) under one or more
 * contributor license agreements.  See the NOTICE file distributed with
 * this work for additional information regarding copyright ownership.
 * The ASF licenses this file to You under the Apache License, Version 2.0
 * (the "License"); you may not use this file except in compliance with
 * the License.  You may obtain a copy of the License at
 *
 *    http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

class RandomForestClassifier @Since("1.4.0") (
  @Since("1.4.0") override val uid: String)
  extends ProbabilisticClassifier[Vector, RandomForestClassifier, RandomForestClassificationModel]
  with RandomForestClassifierParams with DefaultParamsWritable {
  @Since("1.4.0")
  def this() = this(Identifiable.randomUID("rfc"))
```

```
// Override parameter setters from parent trait for Java API compatibility.
// Parameters from TreeClassifierParams:
/** @group setParam */
@Since("1.4.0")
def setMaxDepth(value: Int): this.type = set(maxDepth, value)
/** @group setParam */
@Since("1.4.0")
def setMaxBins(value: Int): this.type = set(maxBins, value)
/** @group setParam */
@Since("1.4.0")
def setMinInstancesPerNode(value: Int): this.type = set(minInstancesPerNode, value)
/** @group setParam */
@Since("3.0.0")
def setMinWeightFractionPerNode(value: Double): this.type = set(minWeightFractionPerNode, value)
/** @group setParam */
@Since("1.4.0")
def setMinInfoGain(value: Double): this.type = set(minInfoGain, value)
/** @group expertSetParam */
@Since("1.4.0")
def setMaxMemoryInMB(value: Int): this.type = set(maxMemoryInMB, value)
/** @group expertSetParam */
@Since("1.4.0")
def setCacheNodeIds(value: Boolean): this.type = set(cacheNodeIds, value)
/**
 * Specifies how often to checkpoint the cached node IDs.
 * E.g. 10 means that the cache will get checkpointed every 10 iterations.
 * This is only used if cacheNodeIds is true and if the checkpoint directory is set in
 * [[org.apache.spark.SparkContext]].
 * Must be at least 1.
 * (default = 10)
 * @group setParam
 */
@Since("1.4.0")
def setCheckpointInterval(value: Int): this.type = set(checkpointInterval, value)
/** @group setParam */
@Since("1.4.0")
def setImpurity(value: String): this.type = set(impurity, value)
// Parameters from TreeEnsembleParams:
/** @group setParam */
@Since("1.4.0")
def setSubsamplingRate(value: Double): this.type = set(subsamplingRate, value)
/** @group setParam */
@Since("1.4.0")
def setSeed(value: Long): this.type = set(seed, value)
// Parameters from RandomForestParams:
/** @group setParam */
@Since("1.4.0")
def setNumTrees(value: Int): this.type = set(numTrees, value)
/** @group setParam */
@Since("3.0.0")
def setBootstrap(value: Boolean): this.type = set(bootstrap, value)
/** @group setParam */
@Since("1.4.0")
def setFeatureSubsetStrategy(value: String): this.type =
  set(featureSubsetStrategy, value)

/**
 * Sets the value of param [[weightCol]].
 * If this is not set or empty, we treat all instance weights as 1.0.
```

```
   * By default the weightCol is not set, so all instances have weight 1.0.
   *
   * @group setParam
   */
  @Since("3.0.0")
  def setWeightCol(value: String): this.type = set(weightCol, value)

  override protected def train(
      dataset: Dataset[_]): RandomForestClassificationModel = instrumented { instr =>
    instr.logPipelineStage(this)
    instr.logDataset(dataset)
    val categoricalFeatures: Map[Int, Int] =
      MetadataUtils.getCategoricalFeatures(dataset.schema($(featuresCol)))
    val numClasses: Int = getNumClasses(dataset)

    if (isDefined(thresholds)) {
      require($(thresholds).length == numClasses, this.getClass.getSimpleName +
        ".train() called with non-matching numClasses and thresholds.length." +
        s" numClasses=$numClasses, but thresholds has length ${$(thresholds).length}")
    }

    val instances = extractInstances(dataset, numClasses)
    val strategy =
      super.getOldStrategy(categoricalFeatures, numClasses, OldAlgo.Classification, getOldImpurity)
    strategy.bootstrap = $(bootstrap)

    instr.logParams(this, labelCol, featuresCol, weightCol, predictionCol, probabilityCol,
      rawPredictionCol, leafCol, impurity, numTrees, featureSubsetStrategy, maxDepth, maxBins,
      maxMemoryInMB, minInfoGain, minInstancesPerNode, minWeightFractionPerNode, seed,
      subsamplingRate, thresholds, cacheNodeIds, checkpointInterval, bootstrap)

    val trees = RandomForest
      .run(instances, strategy, getNumTrees, getFeatureSubsetStrategy, getSeed, Some(instr))
      .map(_.asInstanceOf[DecisionTreeClassificationModel])
    trees.foreach(copyValues(_))

    val numFeatures = trees.head.numFeatures
    instr.logNumClasses(numClasses)
    instr.logNumFeatures(numFeatures)
    createModel(dataset, trees, numFeatures, numClasses)
  }

  private def createModel(
      dataset: Dataset[_],
      trees: Array[DecisionTreeClassificationModel],
      numFeatures: Int,
      numClasses: Int): RandomForestClassificationModel = {
    val model = copyValues(new RandomForestClassificationModel(uid, trees, numFeatures, numClasses))
    val weightColName = if (!isDefined(weightCol)) "weightCol" else $(weightCol)

    val (summaryModel, probabilityColName, predictionColName) = model.findSummaryModel()
    val rfSummary = if (numClasses <= 2) {
      new BinaryRandomForestClassificationTrainingSummaryImpl(
        summaryModel.transform(dataset),
        probabilityColName,
        predictionColName
$(labelCol),
        weightColName,
        Array(0.0))
    } else {
      new RandomForestClassificationTrainingSummaryImpl(
        summaryModel.transform(dataset),
        predictionColName,
        $(labelCol),
        weightColName,
        Array(0.0))
    }
    model.setSummary(Some(rfSummary))
  }

  @Since("1.4.1")
  override def copy(extra: ParamMap): RandomForestClassifier = defaultCopy(extra)
}

@Since("1.4.0")
object RandomForestClassifier extends DefaultParamsReadable[RandomForestClassifier] {
  /** Accessor for supported impurity settings: entropy, gini */
  @Since("1.4.0")
  final val supportedImpurities: Array[String] = TreeClassifierParams.supportedImpurities

  /** Accessor for supported featureSubsetStrategy settings: auto, all, onethird, sqrt, log2 */
  @Since("1.4.0")
  final val supportedFeatureSubsetStrategies: Array[String] =
    TreeEnsembleParams.supportedFeatureSubsetStrategies

  @Since("2.0.0")
  override def load(path: String): RandomForestClassifier = super.load(path)
}

/**
 * <a href="http://en.wikipedia.org/wiki/Random_forest">Random Forest</a> model for classification.
 * It supports both binary and multiclass labels, as well as both continuous and categorical
 * features.
 *
 * @param _trees  Decision trees in the ensemble.
 *                Warning: These have null parents.
 */
@Since("1.4.0")
class RandomForestClassificationModel private[ml] (
    @Since("1.5.0") override val uid: String,
    private val _trees: Array[DecisionTreeClassificationModel],
    @Since("1.6.0") override val numFeatures: Int,
    @Since("1.5.0") override val numClasses: Int)
  extends ProbabilisticClassificationModel[Vector, RandomForestClassificationModel]
  with RandomForestClassifierParams with TreeEnsembleModel[DecisionTreeClassificationModel]
  with MLWritable with Serializable
  with HasTrainingSummary[RandomForestClassificationTrainingSummary] {

  require(_trees.nonEmpty, "RandomForestClassificationModel requires at least 1 tree.")

  /**
   * Construct a random forest classification model, with all trees weighted equally.
   *
   * @param trees  Component trees
   */
  private[ml] def this(
      trees: Array[DecisionTreeClassificationModel],
      numFeatures: Int,
      numClasses: Int) =
    this(Identifiable.randomUID("rfc"), trees, numFeatures, numClasses)
  @Since("1.4.0")
  override def trees: Array[DecisionTreeClassificationModel] = _trees

  // Note: We may add support for weights (based on tree performance) later on.
  private lazy val _treeWeights: Array[Double] = Array.fill[Double](_trees.length)(1.0)

  @Since("1.4.0")
  override def treeWeights: Array[Double] = _treeWeights

  /**
   * Gets summary of model on training set. An exception is thrown
   * if `hasSummary` is false.
   */
  @Since("3.1.0")
  override def summary: RandomForestClassificationTrainingSummary = super.summary

  /**
   * Gets summary of model on training set. An exception is thrown
   * if `hasSummary` is false or it is a multiclass model.
   */
  @Since("3.1.0")
  def binarySummary: BinaryRandomForestClassificationTrainingSummary = summary match {
    case b: BinaryRandomForestClassificationTrainingSummary => b
    case _ =>
      throw new RuntimeException("Cannot create a binary summary for a non-binary model" +
        s"(numClasses=${numClasses}), use summary instead.")
  }

  /**
   * Evaluates the model on a test dataset.
   *
   * @param dataset Test dataset to evaluate model on.
   */
  @Since("3.1.0")
  def evaluate(dataset: Dataset[_]): RandomForestClassificationSummary = {
    val weightColName = if (!isDefined(weightCol)) "weightCol" else $(weightCol)
    // Handle possible missing or invalid prediction columns
    val (summaryModel, probabilityColName, predictionColName) = findSummaryModel()
    if (numClasses > 2) {
      new RandomForestClassificationSummaryImpl(summaryModel.transform(dataset),
        predictionColName, $(labelCol), weightColName)
    } else {
      new BinaryRandomForestClassificationSummaryImpl(summaryModel.transform(dataset),
        probabilityColName, predictionColName, $(labelCol), weightColName)
    }
  }

  @Since("1.4.0")
  override def transformSchema(schema: StructType): StructType = {
    var outputSchema = super.transformSchema(schema)
    if ($(leafCol).nonEmpty) {
      outputSchema = SchemaUtils.updateField(outputSchema, getLeafField($(leafCol)))
    }
    outputSchema
  }
override def transform(dataset: Dataset[_]): DataFrame = {
    val outputSchema = transformSchema(dataset.schema, logging = true)

    val outputData = super.transform(dataset)
  if ($(leafCol).nonEmpty) {
    val leafUDF = udf { features: Vector => predictLeaf(features) }
    outputData.withColumn($(leafCol), leafUDF(col($(featuresCol))),
      outputSchema($(leafCol)).metadata)
  } else {
    outputData
  }
}

@Since("3.0.0")
override def predictRaw(features: Vector): Vector = {
  // TODO: When we add a generic Bagging class, handle transform there: SPARK-7128
  // Classifies using majority votes.
  // Ignore the tree weights since all are 1.0 for now.
  val votes = Array.ofDim[Double](numClasses)
  _trees.foreach { tree =>
    val classCounts = tree.rootNode.predictImpl(features).impurityStats.stats
    val total = classCounts.sum
    if (total != 0) {
      var i = 0
      while (i < numClasses) {
        votes(i) += classCounts(i) / total
        i += 1
      }
    }
  }
  Vectors.dense(votes)
}

override protected def raw2probabilityInPlace(rawPrediction: Vector): Vector = {
  rawPrediction match {
    case dv: DenseVector =>
      ProbabilisticClassificationModel.normalizeToProbabilitiesInPlace(dv)
      dv
    case sv: SparseVector =>
      throw new RuntimeException("Unexpected error in RandomForestClassificationModel:" +
        " raw2probabilityInPlace encountered SparseVector")
  }
}

@Since("1.4.0")
override def copy(extra: ParamMap): RandomForestClassificationModel = {
  copyValues(new RandomForestClassificationModel(uid, _trees, numFeatures, numClasses), extra)
    .setParent(parent)
}

@Since("1.4.0")
override def toString: String = {
  s"RandomForestClassificationModel: uid=$uid, numTrees=$getNumTrees, numClasses=$numClasses, " +
    s"numFeatures=$numFeatures"
}

/**
 * Estimate of the importance of each feature.
 *
 * Each feature's importance is the average of its importance across all trees in the ensemble
 * The importance vector is normalized to sum to 1. This method is suggested by Hastie et al.
 * (Hastie, Tibshirani, Friedman. "The Elements of Statistical Learning, 2nd Edition." 2001.)
 * and follows the implementation from scikit-learn.
 *
```

```
  * @see `DecisionTreeClassificationModel.featureImportances`
  */
 @Since("1.5.0")
 lazy val featureImportances: Vector = TreeEnsembleModel.featureImportances(trees, numFeatures)

 /** (private[ml]) Convert to a model in the old API */
 private[ml] def toOld: OldRandomForestModel = {
  new OldRandomForestModel(OldAlgo.Classification, _trees.map(_.toOld))
 }

 @Since("2.0.0")
 override def write: MLWriter =
  new RandomForestClassificationModel.RandomForestClassificationModelWriter(this)
}

@Since("2.0.0")
object RandomForestClassificationModel extends MLReadable[RandomForestClassificationModel] {

 @Since("2.0.0")
 override def read: MLReader[RandomForestClassificationModel] =
  new RandomForestClassificationModelReader

 @Since("2.0.0")
 override def load(path: String): RandomForestClassificationModel = super.load(path)

 private[RandomForestClassificationModel]
 class RandomForestClassificationModelWriter(instance: RandomForestClassificationModel)
  extends MLWriter {

  override protected def saveImpl(path: String): Unit = {
   // Note: numTrees is not currently used, but could be nice to store for fast querying.
   val extraMetadata: JObject = Map(
    "numFeatures" -> instance.numFeatures,
    "numClasses" -> instance.numClasses,
    "numTrees" -> instance.getNumTrees)
   EnsembleModelReadWrite.saveImpl(instance, path, sparkSession, extraMetadata)
  }
 }

 private class RandomForestClassificationModelReader
  extends MLReader[RandomForestClassificationModel] {

  /** Checked against metadata when loading model */
  private val className = classOf[RandomForestClassificationModel].getName
  private val treeClassName = classOf[DecisionTreeClassificationModel].getName

  override def load(path: String): RandomForestClassificationModel = {
   implicit val format = DefaultFormats
   val (metadata: Metadata, treesData: Array[(Metadata, Node)], _) =
    EnsembleModelReadWrite.loadImpl(path, sparkSession, className, treeClassName)
   val numFeatures = (metadata.metadata \ "numFeatures").extract[Int]
   val numClasses = (metadata.metadata \ "numClasses").extract[Int]
   val numTrees = (metadata.metadata \ "numTrees").extract[Int]

   val trees: Array[DecisionTreeClassificationModel] = treesData.map {
     case (treeMetadata, root) =>
     val tree =
      new DecisionTreeClassificationModel(treeMetadata.uid, root, numFeatures, numClasses)
     treeMetadata.getAndSetParams(tree)
     tree
   }
   require(numTrees == trees.length, s"RandomForestClassificationModel.load expected $numTrees" +
    s" trees based on metadata but found ${trees.length} trees.")
   val model = new RandomForestClassificationModel(metadata.uid, trees, numFeatures, numClasses)
   metadata.getAndSetParams(model)
   model
  }
 }
```

Code 5.5 Random Forest Classifier (open-source: under Apache Licence)

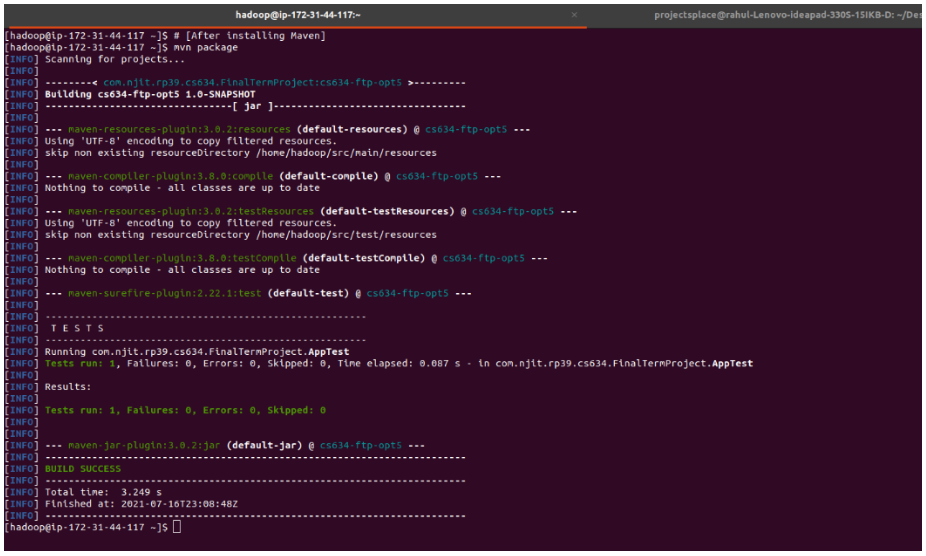
By this we end the chapter and proceed with running the application.

Chapter 6. Running the application on Apache Spark inside the EMR (Elastic MapReduce)

In the previous chapters we have done setup of necessary infrastructure on AWS cloud and together with it we made through with setting up the application implementation. Now that we are set with the implementation details, let execute the program to train the model and then test the model it on EMR using Apache Spark.

After setting up the Apache Maven and setting up the files, use the following command to build the .jar package.

```
$ # [After installing Apache Maven, execute below line in directory where pom.xml resides]
Important!!
$ mvn package
```



```
hadoop@ip-172-31-44-117:~
[hadoop@ip-172-31-44-117 ~]$ # [After installing Maven]
[hadoop@ip-172-31-44-117 ~]$ mvn package
[INFO] Scanning for projects...
[INFO]
[INFO] --------< com.njit.rp39.cs634.FinalTermProject:cs634-ftp-opt5 >---------
[INFO] Building cs634-ftp-opt5 1.0-SNAPSHOT
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- maven-resources-plugin:3.0.2:resources (default-resources) @ cs634-ftp-opt5 ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory /home/hadoop/src/main/resources
[INFO]
[INFO] --- maven-compiler-plugin:3.8.0:compile (default-compile) @ cs634-ftp-opt5 ---
[INFO] Nothing to compile - all classes are up to date
[INFO]
[INFO] --- maven-resources-plugin:3.0.2:testResources (default-testResources) @ cs634-ftp-opt5 ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory /home/hadoop/src/test/resources
[INFO]
[INFO] --- maven-compiler-plugin:3.8.0:testCompile (default-testCompile) @ cs634-ftp-opt5 ---
[INFO] Nothing to compile - all classes are up to date
[INFO]
[INFO] --- maven-surefire-plugin:2.22.1:test (default-test) @ cs634-ftp-opt5 ---
[INFO]
[INFO] -------------------------------------------------------
[INFO]  T E S T S
[INFO] -------------------------------------------------------
[INFO] Running com.njit.rp39.cs634.FinalTermProject.AppTest
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.087 s - in com.njit.rp39.cs634.FinalTermProject.AppTest
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO]
[INFO] --- maven-jar-plugin:3.0.2:jar (default-jar) @ cs634-ftp-opt5 ---
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  3.249 s
[INFO] Finished at: 2021-07-16T23:08:48Z
[INFO] ------------------------------------------------------------------------
[hadoop@ip-172-31-44-117 ~]$
```

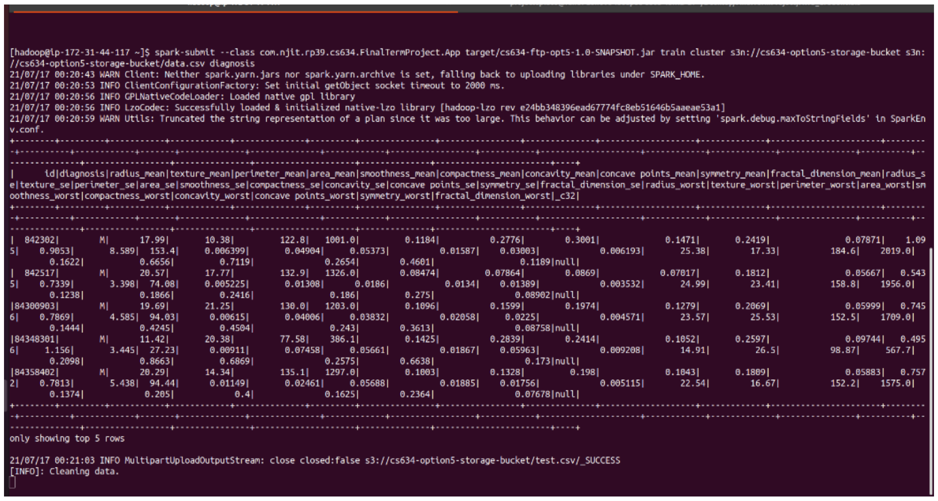
Fig 6.1 Performing a Maven build

6.1 Running application for Training the model

In order to run the program using .jar package, execute the following command.

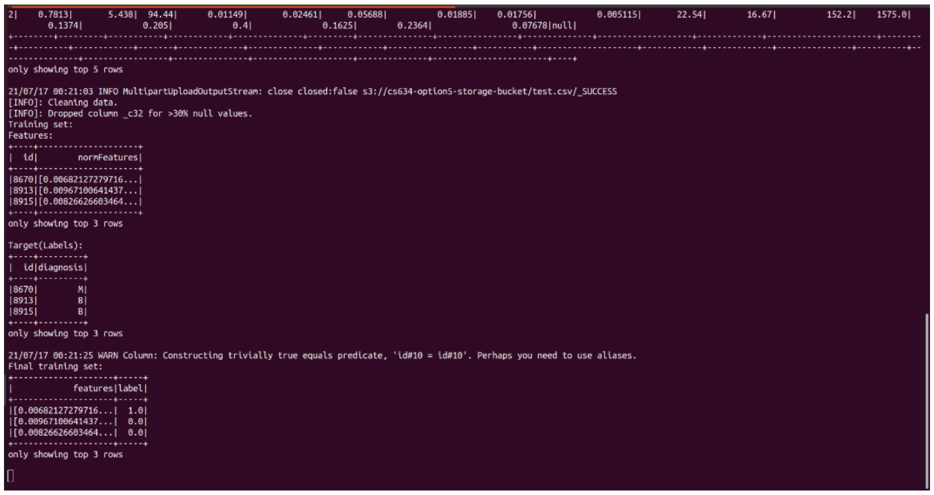
```
$ # [Make sure to put in your S3 bucket URI’s in place of mine.]

$ spark-submit --class com.njit.rp39.cs634.FinalTermProject.App target/cs634-ftp-opt5-1.0-SNAPSHOT.jar train cluster s3n://cs634-option5-storage-bucket s3n://cs634-option5-storage-bucket/data.csv diagnosis
```

This produces the following result shown in the screenshots below.



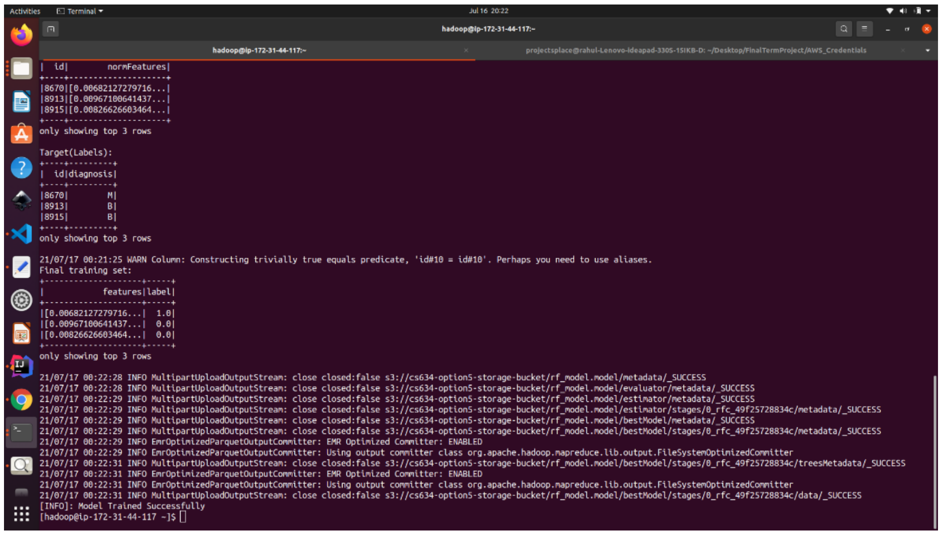
```
[hadoop@ip-172-31-44-117 ~]$ spark-submit --class com.njit.rp39.cs634.FinalTermProject.App target/cs634-ftp-opt5-1.0-SNAPSHOT.jar train cluster s3n://cs634-option5-storage-bucket s3n:
//cs634-option5-storage-bucket/data.csv diagnosis
21/07/17 00:20:43 WARN Client: Neither spark.yarn.jars nor spark.yarn.archive is set, falling back to uploading libraries under SPARK_HOME.
21/07/17 00:20:53 INFO ClientConfigurationFactory: Set initial getObject socket timeout to 2000 ms.
21/07/17 00:20:56 INFO GPLNativeCodeLoader: Loaded native gpl library
21/07/17 00:20:56 INFO LzoCodec: Successfully loaded & initialized native-lzo library [hadoop-lzo rev e24bb348396ead67774fc8eb51646b5aaeae53a1]
21/07/17 00:20:59 WARN Utils: Truncated the string representation of a plan since it was too large. This behavior can be adjusted by setting 'spark.debug.maxToStringFields' in SparkEn
v.conf.
+--------+---------+-----------+------------+--------------+---------+---------------+----------------+--------------+-------------------+-------------+----------------------+--------
-+----------+------------+-------+-------------+--------------+------------+-----------------+-----------+--------------------+------------+-------------+---------------+----------+--
-------------+-----------------+-----------------+--------------------+-------------------------+----+
|      id|diagnosis|radius_mean|texture_mean|perimeter_mean|area_mean|smoothness_mean|compactness_mean|concavity_mean|concave points_mean|symmetry_mean|fractal_dimension_mean|radius_s
e|texture_se|perimeter_se|area_se|smoothness_se|compactness_se|concavity_se|concave points_se|symmetry_se|fractal_dimension_se|radius_worst|texture_worst|perimeter_worst|area_worst|sm
oothness_worst|compactness_worst|concavity_worst|concave points_worst|symmetry_worst|fractal_dimension_worst|_c32|
+--------+---------+-----------+------------+--------------+---------+---------------+----------------+--------------+-------------------+-------------+----------------------+--------
-+----------+------------+-------+-------------+--------------+------------+-----------------+-----------+--------------------+------------+-------------+---------------+----------+--
-------------+-----------------+-----------------+--------------------+-------------------------+----+
|  842302|        M|      17.99|       10.38|         122.8|   1001.0|         0.1184|          0.2776|        0.3001|             0.1471|       0.2419|               0.07871|    1.09
5|    0.9053|       8.589|  153.4|     0.006399|       0.04904|     0.05373|          0.01587|    0.03003|            0.006193|       25.38|        17.33|          184.6|    2019.0|
        0.1622|           0.6656|         0.7119|              0.2654|        0.4601|                 0.1189|null|
|  842517|        M|      20.57|       17.77|         132.9|   1326.0|        0.08474|         0.07864|        0.0869|            0.07017|       0.1812|               0.05667|   0.543
5|    0.7339|       3.398|  74.08|     0.005225|       0.01308|      0.0186|           0.0134|    0.01389|            0.003532|       24.99|        23.41|          158.8|    1956.0|
        0.1238|           0.1866|         0.2416|               0.186|         0.275|                0.08902|null|
|84300903|        M|      19.69|       21.25|         130.0|   1203.0|         0.1096|          0.1599|        0.1974|             0.1279|       0.2069|               0.05999|   0.745
6|    0.7869|       4.585|  94.03|      0.00615|       0.04006|     0.03832|          0.02058|     0.0225|            0.004571|       23.57|        25.53|          152.5|    1709.0|
        0.1444|           0.4245|         0.4504|               0.243|        0.3613|                0.08758|null|
|84348301|        M|      11.42|       20.38|         77.58|    386.1|         0.1425|          0.2839|        0.2414|             0.1052|       0.2597|               0.09744|   0.495
6|     1.156|       3.445|  27.23|      0.00911|       0.07458|     0.05661|          0.01867|    0.05963|            0.009208|       14.91|         26.5|          98.87|     567.7|
        0.2098|           0.8663|         0.6869|              0.2575|        0.6638|                  0.173|null|
|84358402|        M|      20.29|       14.34|         135.1|   1297.0|         0.1003|          0.1328|         0.198|             0.1043|       0.1809|               0.05883|   0.757
2|    0.7813|       5.438|  94.44|      0.01149|       0.02461|     0.05688|          0.01885|    0.01756|            0.005115|       22.54|        16.67|          152.2|    1575.0|
        0.1374|            0.205|            0.4|              0.1625|        0.2364|                0.07678|null|
+--------+---------+-----------+------------+--------------+---------+---------------+----------------+--------------+-------------------+-------------+----------------------+--------
-+----------+------------+-------+-------------+--------------+------------+-----------------+-----------+--------------------+------------+-------------+---------------+----------+--
-------------+-----------------+-----------------+--------------------+-------------------------+----+
only showing top 5 rows

21/07/17 00:21:03 INFO MultipartUploadOutputStream: close closed:false s3://cs634-option5-storage-bucket/test.csv/_SUCCESS
[INFO]: Cleaning data.
```

Fig 6.2 Executing the Model Training process (1)



```
2|    0.7813|       5.438|  94.44|      0.01149|       0.02461|     0.05688|          0.01885|    0.01756|            0.005115|       22.54|        16.67|          152.2|    1575.0|
        0.1374|            0.205|            0.4|              0.1625|        0.2364|                0.07678|null|
+--------+---------+-----------+------------+--------------+---------+---------------+----------------+--------------+-------------------+-------------+----------------------+--------
-+----------+------------+-------+-------------+--------------+------------+-----------------+-----------+--------------------+------------+-------------+---------------+----------+--
-------------+-----------------+-----------------+--------------------+-------------------------+----+
only showing top 5 rows

21/07/17 00:21:03 INFO MultipartUploadOutputStream: close closed:false s3://cs634-option5-storage-bucket/test.csv/_SUCCESS
[INFO]: Cleaning data.
[INFO]: Dropped column _c32 for >30% null values.
Training set:
Features:
+----+--------------------+
|  id|        normFeatures|
+----+--------------------+
|8670|[0.00682127279716...|
|8913|[0.00967100641437...|
|8915|[0.00826626603464...|
+----+--------------------+
only showing top 3 rows

Target(Labels):
+----+---------+
|  id|diagnosis|
+----+---------+
|8670|        M|
|8913|        B|
|8915|        B|
+----+---------+
only showing top 3 rows

21/07/17 00:21:25 WARN Column: Constructing trivially true equals predicate, 'id#10 = id#10'. Perhaps you need to use aliases.
Final training set:
+--------------------+-----+
|            features|label|
+--------------------+-----+
|[0.00682127279716...|  1.0|
|[0.00967100641437...|  0.0|
|[0.00826626603464...|  0.0|
+--------------------+-----+
only showing top 3 rows
```

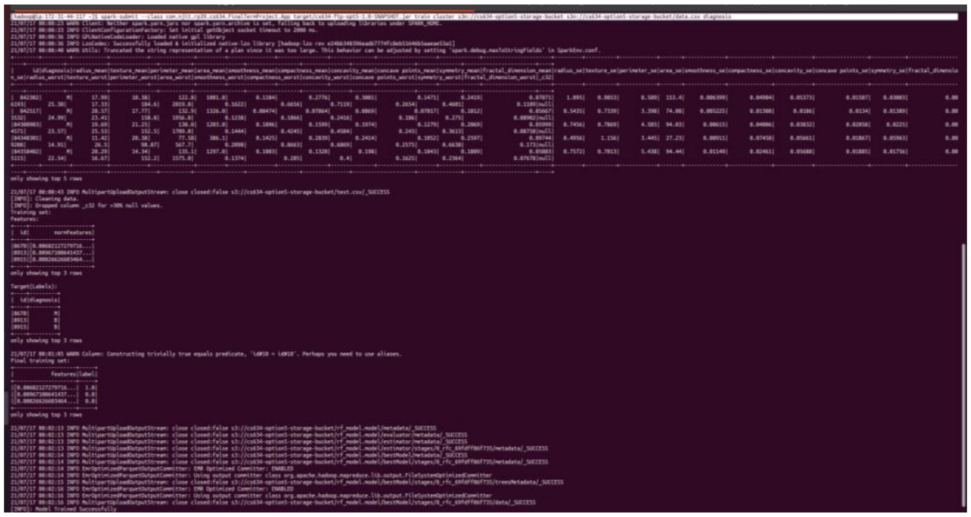
Fig 6.3 Executing the Model Training process (2)



Activities
Terminal
Jul 16 20:22
hadoop@ip-172-31-44-117:~
hadoop@ip-172-31-44-117:~
projectsplace@rahul-Lenovo-ideapad-330S-15IKB-D: ~/Desktop/FinalTermProject/AWS_Credentials

```
|  id|        normFeatures|
+----+--------------------+
|8670|[0.00682127279716...|
|8913|[0.00967100641437...|
|8915|[0.00826626603464...|
+----+--------------------+
only showing top 3 rows

Target(Labels):
+----+---------+
|  id|diagnosis|
+----+---------+
|8670|        M|
|8913|        B|
|8915|        B|
+----+---------+
only showing top 3 rows

21/07/17 00:21:25 WARN Column: Constructing trivially true equals predicate, 'id#10 = id#10'. Perhaps you need to use aliases.
Final training set:
+--------------------+-----+
|            features|label|
+--------------------+-----+
|[0.00682127279716...|  1.0|
|[0.00967100641437...|  0.0|
|[0.00826626603464...|  0.0|
+--------------------+-----+
only showing top 3 rows

21/07/17 00:22:28 INFO MultipartUploadOutputStream: close closed:false s3://cs634-option5-storage-bucket/rf_model.model/metadata/_SUCCESS
21/07/17 00:22:28 INFO MultipartUploadOutputStream: close closed:false s3://cs634-option5-storage-bucket/rf_model.model/evaluator/metadata/_SUCCESS
21/07/17 00:22:29 INFO MultipartUploadOutputStream: close closed:false s3://cs634-option5-storage-bucket/rf_model.model/estimator/metadata/_SUCCESS
21/07/17 00:22:29 INFO MultipartUploadOutputStream: close closed:false s3://cs634-option5-storage-bucket/rf_model.model/estimator/stages/0_rfc_49f25728834c/metadata/_SUCCESS
21/07/17 00:22:29 INFO MultipartUploadOutputStream: close closed:false s3://cs634-option5-storage-bucket/rf_model.model/bestModel/metadata/_SUCCESS
21/07/17 00:22:29 INFO MultipartUploadOutputStream: close closed:false s3://cs634-option5-storage-bucket/rf_model.model/bestModel/stages/0_rfc_49f25728834c/metadata/_SUCCESS
21/07/17 00:22:29 INFO EmrOptimizedParquetOutputCommitter: EMR Optimized Committer: ENABLED
21/07/17 00:22:29 INFO EmrOptimizedParquetOutputCommitter: Using output committer class org.apache.hadoop.mapreduce.lib.output.FileSystemOptimizedCommitter
21/07/17 00:22:31 INFO MultipartUploadOutputStream: close closed:false s3://cs634-option5-storage-bucket/rf_model.model/bestModel/stages/0_rfc_49f25728834c/treesMetadata/_SUCCESS
21/07/17 00:22:31 INFO EmrOptimizedParquetOutputCommitter: EMR Optimized Committer: ENABLED
21/07/17 00:22:31 INFO EmrOptimizedParquetOutputCommitter: Using output committer class org.apache.hadoop.mapreduce.lib.output.FileSystemOptimizedCommitter
21/07/17 00:22:31 INFO MultipartUploadOutputStream: close closed:false s3://cs634-option5-storage-bucket/rf_model.model/bestModel/stages/0_rfc_49f25728834c/data/_SUCCESS
[INFO]: Model Trained Successfully
[hadoop@ip-172-31-44-117 ~]$
```

Fig 6.4 Executing the Model Training process (3)



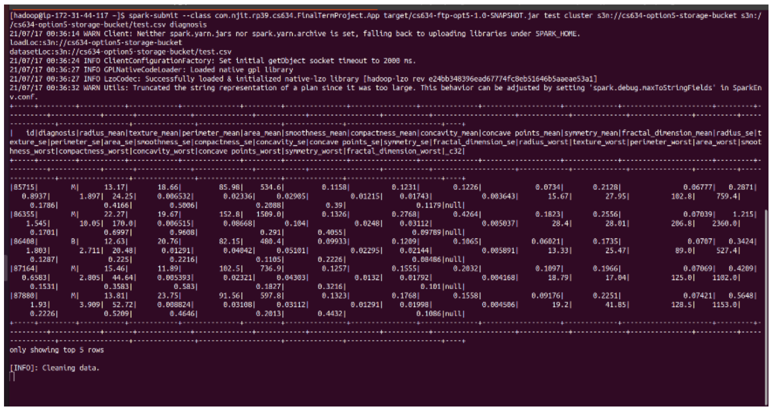
```
[hadoop@ip-172-31-44-117 ~]$ spark-submit --class com.njit.rp39.cs634.FinalTermProject.App target/cs634-ftp-opt5-1.0-SNAPSHOT.jar train cluster s3n://cs634-option5-storage-bucket s3n://cs634-option5-storage-bucket/data.csv diagnosis
21/07/17 00:00:23 WARN Client: Neither spark.yarn.jars nor spark.yarn.archive is set, falling back to uploading libraries under SPARK_HOME.
21/07/17 00:00:33 INFO ClientConfigurationFactory: Set initial getObject socket timeout to 2000 ms.
21/07/17 00:00:36 INFO GPLNativeCodeLoader: Loaded native gpl library
21/07/17 00:00:36 INFO LzoCodec: Successfully loaded & initialized native-lzo library [hadoop-lzo rev e24bb348396ead67774fc8eb51646b5aeae53a1]
21/07/17 00:00:40 WARN Utils: Truncated the string representation of a plan since it was too large. This behavior can be adjusted by setting 'spark.debug.maxToStringFields' in SparkEnv.conf.
+--------+---------+-----------+------------+--------------+---------+---------------+----------------+--------------+-------------------+-------------+----------------------+---------+----------+------------+-------+-------------+--------------+------------+-----------------+-----------+---------------
+-----------+------------+-------------+---------------+----------+----------------+-----------------+---------------+--------------------+--------------+-----------------------+----+
|      id|diagnosis|radius_mean|texture_mean|perimeter_mean|area_mean|smoothness_mean|compactness_mean|concavity_mean|concave points_mean|symmetry_mean|fractal_dimension_mean|radius_se|texture_se|perimeter_se|area_se|smoothness_se|compactness_se|concavity_se|concave points_se|symmetry_se|fractal_dimensio
n_se|radius_worst|texture_worst|perimeter_worst|area_worst|smoothness_worst|compactness_worst|concavity_worst|concave points_worst|symmetry_worst|fractal_dimension_worst|_c32|
+--------+---------+-----------+------------+--------------+---------+---------------+----------------+--------------+-------------------+-------------+----------------------+---------+----------+------------+-------+-------------+--------------+------------+-----------------+-----------+---------------
+-----------+------------+-------------+---------------+----------+----------------+-----------------+---------------+--------------------+--------------+-----------------------+----+
|  842302|        M|      17.99|       10.38|         122.8|   1001.0|         0.1184|          0.2776|        0.3001|             0.1471|       0.2419|               0.07871|    1.095|    0.9053|       8.589|  153.4|     0.006399|       0.04904|     0.05373|          0.01587|    0.03003|           0.00
6193|       25.38|        17.33|          184.6|    2019.0|          0.1622|           0.6656|         0.7119|              0.2654|        0.4601|                 0.1189|null|
|  842517|        M|      20.57|       17.77|         132.9|   1326.0|        0.08474|         0.07864|        0.0869|            0.07017|       0.1812|               0.05667|   0.5435|    0.7339|       3.398|  74.08|     0.005225|       0.01308|      0.0186|           0.0134|    0.01389|           0.00
3532|       24.99|        23.41|          158.8|    1956.0|          0.1238|           0.1866|         0.2416|               0.186|         0.275|                0.08902|null|
|84300903|        M|      19.69|       21.25|         130.0|   1203.0|         0.1096|          0.1599|        0.1974|             0.1279|       0.2069|               0.05999|   0.7456|    0.7869|       4.585|  94.03|      0.00615|       0.04006|     0.03832|          0.02058|     0.0225|           0.00
4571|       23.57|        25.53|          152.5|    1709.0|          0.1444|           0.4245|         0.4504|               0.243|        0.3613|                0.08758|null|
|84348301|        M|      11.42|       20.38|         77.58|    386.1|         0.1425|          0.2839|        0.2414|             0.1052|       0.2597|               0.09744|   0.4956|     1.156|       3.445|  27.23|      0.00911|       0.07458|     0.05661|          0.01867|    0.05963|           0.00
9208|       14.91|         26.5|          98.87|     567.7|          0.2098|           0.8663|         0.6869|              0.2575|        0.6638|                  0.173|null|
|84358402|        M|      20.29|       14.34|         135.1|   1297.0|         0.1003|          0.1328|         0.198|             0.1043|       0.1809|               0.05883|   0.7572|    0.7813|       5.438|  94.44|      0.01149|       0.02461|     0.05688|          0.01885|    0.01756|           0.00
5115|       22.54|        16.67|          152.2|    1575.0|          0.1374|            0.205|            0.4|              0.1625|        0.2364|                0.07678|null|
+--------+---------+-----------+------------+--------------+---------+---------------+----------------+--------------+-------------------+-------------+----------------------+---------+----------+------------+-------+-------------+--------------+------------+-----------------+-----------+---------------
+-----------+------------+-------------+---------------+----------+----------------+-----------------+---------------+--------------------+--------------+-----------------------+----+
only showing top 5 rows

21/07/17 00:00:43 INFO MultipartUploadOutputStream: close closed:false s3://cs634-option5-storage-bucket/test.csv/_SUCCESS
[INFO]: Cleaning data.
[INFO]: Dropped column _c32 for >30% null values.
Training set:
Features:
+----+--------------------+
|  id|        normFeatures|
+----+--------------------+
|8670|[0.00682127279716...|
|8913|[0.00967100641437...|
|8915|[0.00826626683464...|
+----+--------------------+
only showing top 3 rows

Target(Labels):
+----+---------+
|  id|diagnosis|
+----+---------+
|8670|        M|
|8913|        B|
|8915|        B|
+----+---------+
only showing top 3 rows

21/07/17 00:01:05 WARN Column: Constructing trivially true equals predicate, 'id#10 = id#10'. Perhaps you need to use aliases.
Final training set:
+--------------------+-----+
|            features|label|
+--------------------+-----+
|[0.00682127279716...|  1.0|
|[0.00967100641437...|  0.0|
|[0.00826626683464...|  0.0|
+--------------------+-----+
only showing top 3 rows

21/07/17 00:02:13 INFO MultipartUploadOutputStream: close closed:false s3://cs634-option5-storage-bucket/rf_model.model/metadata/_SUCCESS
21/07/17 00:02:13 INFO MultipartUploadOutputStream: close closed:false s3://cs634-option5-storage-bucket/rf_model.model/evaluator/metadata/_SUCCESS
21/07/17 00:02:13 INFO MultipartUploadOutputStream: close closed:false s3://cs634-option5-storage-bucket/rf_model.model/estimator/metadata/_SUCCESS
21/07/17 00:02:13 INFO MultipartUploadOutputStream: close closed:false s3://cs634-option5-storage-bucket/rf_model.model/estimator/stages/0_rfc_69fdff86f735/metadata/_SUCCESS
21/07/17 00:02:14 INFO MultipartUploadOutputStream: close closed:false s3://cs634-option5-storage-bucket/rf_model.model/bestModel/metadata/_SUCCESS
21/07/17 00:02:14 INFO MultipartUploadOutputStream: close closed:false s3://cs634-option5-storage-bucket/rf_model.model/bestModel/stages/0_rfc_69fdff86f735/metadata/_SUCCESS
21/07/17 00:02:14 INFO EmrOptimizedParquetOutputCommitter: EMR Optimized Committer: ENABLED
21/07/17 00:02:14 INFO EmrOptimizedParquetOutputCommitter: Using output committer class org.apache.hadoop.mapreduce.lib.output.FileSystemOptimizedCommitter
21/07/17 00:02:15 INFO MultipartUploadOutputStream: close closed:false s3://cs634-option5-storage-bucket/rf_model.model/bestModel/stages/0_rfc_69fdff86f735/treesMetadata/_SUCCESS
21/07/17 00:02:16 INFO EmrOptimizedParquetOutputCommitter: EMR Optimized Committer: ENABLED
21/07/17 00:02:16 INFO EmrOptimizedParquetOutputCommitter: Using output committer class org.apache.hadoop.mapreduce.lib.output.FileSystemOptimizedCommitter
21/07/17 00:02:16 INFO MultipartUploadOutputStream: close closed:false s3://cs634-option5-storage-bucket/rf_model.model/bestModel/stages/0_rfc_69fdff86f735/data/_SUCCESS
[INFO]: Model Trained Successfully
```

Fig 6.5 Complete look at the Training Process

6.2 Running application for Testing the model

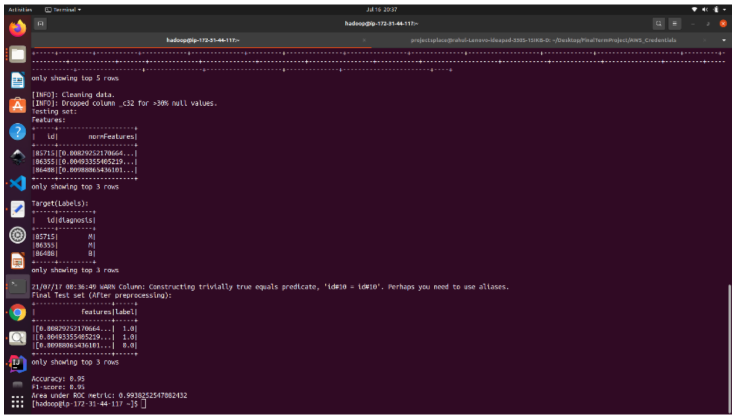
For running a test using testing dataset execute the following:

```
$ spark-submit --class com.njit.rp39.cs634.FinalTermProject.App target/cs634-ftp-opt5-1.0-
SNAPSHOT.jar test cluster s3n://cs634-option5-storage-bucket s3n://cs634-option5-storage-
bucket/test.csv diagnosis
```



```
[hadoop@ip-172-31-44-117 ~]$ spark-submit --class com.njit.rp39.cs634.FinalTermProject.App target/cs634-ftp-opt5-1.0-SNAPSHOT.jar test cluster s3n://cs634-option5-storage-bucket s3n:/
/cs634-option5-storage-bucket/test.csv diagnosis
21/07/17 00:36:14 WARN Client: Neither spark.yarn.jars nor spark.yarn.archive is set, falling back to uploading libraries under SPARK_HOME.
loadLoc:s3n://cs634-option5-storage-bucket
datasetLoc:s3n://cs634-option5-storage-bucket/test.csv
21/07/17 00:36:24 INFO ClientConfigurationFactory: Set initial getObject socket timeout to 2000 ms.
21/07/17 00:36:27 INFO GPLNativeCodeLoader: Loaded native gpl library
21/07/17 00:36:27 INFO LzoCodec: Successfully loaded & initialized native-lzo library [hadoop-lzo rev e24bb348396ead67774fc8eb51646b5aeae53a1]
21/07/17 00:36:32 WARN Utils: Truncated the string representation of a plan since it was too large. This behavior can be adjusted by setting 'spark.debug.maxToStringFields' in SparkEn
v.conf.
+-----+---------+-----------+------------+--------------+---------+---------------+----------------+--------------+-------------------+-------------+----------------------+---------+-
---------+------------+-------+-------------+--------------+------------+-----------------+-----------+--------------------+------------+-------------+---------------+----------+-----
-----------+-----------------+---------------+--------------------+--------------+-----------------------+----+
|   id|diagnosis|radius_mean|texture_mean|perimeter_mean|area_mean|smoothness_mean|compactness_mean|concavity_mean|concave points_mean|symmetry_mean|fractal_dimension_mean|radius_se|t
exture_se|perimeter_se|area_se|smoothness_se|compactness_se|concavity_se|concave points_se|symmetry_se|fractal_dimension_se|radius_worst|texture_worst|perimeter_worst|area_worst|smoot
hness_worst|compactness_worst|concavity_worst|concave points_worst|symmetry_worst|fractal_dimension_worst|_c32|
+-----+---------+-----------+------------+--------------+---------+---------------+----------------+--------------+-------------------+-------------+----------------------+---------+-
---------+------------+-------+-------------+--------------+------------+-----------------+-----------+--------------------+------------+-------------+---------------+----------+-----
-----------+-----------------+---------------+--------------------+--------------+-----------------------+----+
|85715|        M|      13.17|       18.66|         85.98|    534.6|         0.1158|          0.1231|        0.1226|             0.0734|       0.2128|               0.06777|   0.2871|
   0.8937|       1.897|  24.25|     0.006532|       0.02336|     0.02905|          0.01215|    0.01743|            0.003643|       15.67|        27.95|          102.8|     759.4|
    0.1786|           0.4166|         0.5006|              0.2088|          0.39|                 0.1179|null|
|86355|        M|      22.27|       19.67|         152.8|   1509.0|         0.1326|          0.2768|        0.4264|             0.1823|       0.2556|               0.07039|    1.215|
    1.545|       10.05|  170.0|     0.006515|       0.08668|       0.104|           0.0248|    0.03112|            0.005037|        28.4|        28.01|          206.8|    2360.0|
    0.1701|           0.6997|         0.9608|               0.291|        0.4055|                0.09789|null|
|86408|        B|      12.63|       20.76|         82.15|    480.4|        0.09933|          0.1209|        0.1065|            0.06021|       0.1735|                0.0707|   0.3424|
    1.803|       2.711|  20.48|      0.01291|       0.04042|     0.05101|          0.02295|    0.02144|            0.005891|       13.33|        25.47|           89.0|     527.4|
    0.1287|            0.225|         0.2216|              0.1105|        0.2226|                0.08486|null|
|87164|        M|      15.46|       11.89|         102.5|    736.9|         0.1257|          0.1555|        0.2032|             0.1097|       0.1966|               0.07069|   0.4209|
   0.6583|       2.805|  44.64|     0.005393|       0.02321|     0.04303|           0.0132|    0.01792|            0.004168|       18.79|        17.04|          125.0|    1102.0|
    0.1531|           0.3583|          0.583|              0.1827|        0.3216|                  0.101|null|
|87880|        M|      13.81|       23.75|         91.56|    597.8|         0.1323|          0.1768|        0.1558|            0.09176|       0.2251|               0.07421|   0.5648|
     1.93|       3.909|  52.72|     0.008824|       0.03108|     0.03112|          0.01291|    0.01998|            0.004506|        19.2|        41.85|          128.5|    1153.0|
    0.2226|           0.5209|         0.4646|              0.2013|        0.4432|                 0.1086|null|
+-----+---------+-----------+------------+--------------+---------+---------------+----------------+--------------+-------------------+-------------+----------------------+---------+-
---------+------------+-------+-------------+--------------+------------+-----------------+-----------+--------------------+------------+-------------+---------------+----------+-----
-----------+-----------------+---------------+--------------------+--------------+-----------------------+----+
only showing top 5 rows

[INFO]: Cleaning data.
```

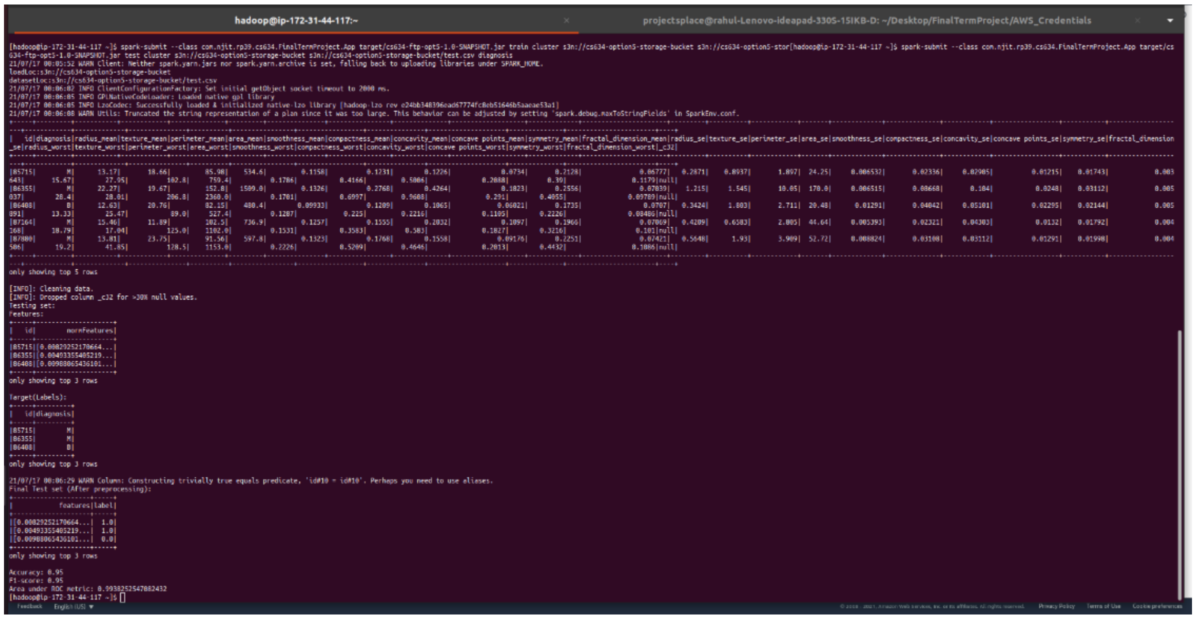
Fig 6.6 Executing the Model Testing process (1)



Activities
Terminal
Jul 16 20:37
hadoop@ip-172-31-44-117:~
hadoop@ip-172-31-44-117:~
projectsplace@rahul-Lenovo-ideapad-330S-15IKB-D: ~/Desktop/FinalTermProject/AWS_Credentials

```
only showing top 5 rows

[INFO]: Cleaning data.
[INFO]: Dropped column _c32 for >30% null values.
Testing set:
Features:
+-----+--------------------+
|   id|        normFeatures|
+-----+--------------------+
|85715|[0.00829252170664...|
|86355|[0.00493355405219...|
|86408|[0.00988065436101...|
+-----+--------------------+
only showing top 3 rows

Target(Labels):
+-----+---------+
|   id|diagnosis|
+-----+---------+
|85715|        M|
|86355|        M|
|86408|        B|
+-----+---------+
only showing top 3 rows

21/07/17 00:36:49 WARN Column: Constructing trivially true equals predicate, 'id#10 = id#10'. Perhaps you need to use aliases.
Final Test set (After preprocessing):
+--------------------+-----+
|            features|label|
+--------------------+-----+
|[0.00829252170664...|  1.0|
|[0.00493355405219...|  1.0|
|[0.00988065436101...|  0.0|
+--------------------+-----+
only showing top 3 rows

Accuracy: 0.95
F1-score: 0.95
Area under ROC metric: 0.9938252547082432
[hadoop@ip-172-31-44-117 ~]$ 
```

Fig 6.7 Executing the Model Testing process (2)


hadoop@ip-172-31-44-117:~
projectsplace@rahul-Lenovo-ideapad-330S-15IKB-D: ~/Desktop/FinalTermProject/AWS_Credentials
Feedback
English (US)
© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use
Cookie preferences

```
[hadoop@ip-172-31-44-117 ~]$ spark-submit --class com.njit.rp39.cs634.FinalTermProject.App target/cs634-ftp-opt5-1.0-SNAPSHOT.jar train cluster s3n://cs634-option5-storage-bucket s3n://cs634-option5-stor[hadoop@ip-172-31-44-117 ~]$ spark-submit --class com.njit.rp39.cs634.FinalTermProject.App target/cs
634-ftp-opt5-1.0-SNAPSHOT.jar test cluster s3n://cs634-option5-storage-bucket s3n://cs634-option5-storage-bucket/test.csv diagnosis
21/07/17 00:05:52 WARN Client: Neither spark.yarn.jars nor spark.yarn.archive is set, falling back to uploading libraries under SPARK_HOME.
loadLoc:s3n://cs634-option5-storage-bucket
datasetLoc:s3n://cs634-option5-storage-bucket/test.csv
21/07/17 00:06:02 INFO ClientConfigurationFactory: Set initial getObject socket timeout to 2000 ms.
21/07/17 00:06:05 INFO GPLNativeCodeLoader: Loaded native gpl library
21/07/17 00:06:05 INFO LzoCodec: Successfully loaded & initialized native-lzo library [hadoop-lzo rev e24bb348396ead67774fc8eb51646b5aaeae53a1]
21/07/17 00:06:08 WARN Utils: Truncated the string representation of a plan since it was too large. This behavior can be adjusted by setting 'spark.debug.maxToStringFields' in SparkEnv.conf.
+-----+---------+-----------+------------+--------------+---------+---------------+----------------+--------------+-------------------+-------------+----------------------+---------+----------+------------+-------+-------------+--------------+------------+-----------------+-----------+-----------------
---+---------+-----------+------------+--------------+---------+---------------+----------------+--------------+-------------------+-------------+----------------------+----+
|   id|diagnosis|radius_mean|texture_mean|perimeter_mean|area_mean|smoothness_mean|compactness_mean|concavity_mean|concave points_mean|symmetry_mean|fractal_dimension_mean|radius_se|texture_se|perimeter_se|area_se|smoothness_se|compactness_se|concavity_se|concave points_se|symmetry_se|fractal_dimension
_se|radius_worst|texture_worst|perimeter_worst|area_worst|smoothness_worst|compactness_worst|concavity_worst|concave points_worst|symmetry_worst|fractal_dimension_worst|_c32|
+-----+---------+-----------+------------+--------------+---------+---------------+----------------+--------------+-------------------+-------------+----------------------+---------+----------+------------+-------+-------------+--------------+------------+-----------------+-----------+-----------------
---+---------+-----------+------------+--------------+---------+---------------+----------------+--------------+-------------------+-------------+----------------------+----+
|85715|        M|      13.17|       18.66|         85.98|    534.6|         0.1158|          0.1231|        0.1226|             0.0734|       0.2128|               0.06777|   0.2871|    0.8937|       1.897|  24.25|     0.006532|       0.02336|     0.02905|          0.01215|    0.01743|            0.003
643|      15.67|        27.95|          102.8|     759.4|          0.1786|           0.4166|         0.5006|              0.2088|          0.39|                 0.1179|null|
|86355|        M|      22.27|       19.67|         152.8|   1509.0|         0.1326|          0.2768|        0.4264|             0.1823|       0.2556|               0.07039|    1.215|     1.545|       10.05|  170.0|     0.006515|       0.08668|       0.104|           0.0248|    0.03112|            0.005
037|       28.4|        28.01|          206.8|    2360.0|          0.1701|           0.6997|         0.9608|               0.291|        0.4055|                0.09789|null|
|86408|        B|      12.63|       20.76|         82.15|    480.4|        0.09933|          0.1209|        0.1065|            0.06021|       0.1735|                0.0707|   0.3424|     1.803|       2.711|  20.48|      0.01291|       0.04042|     0.05101|          0.02295|    0.02144|            0.005
891|      13.33|        25.47|           89.0|     527.4|          0.1287|            0.225|         0.2216|              0.1105|        0.2226|                0.08486|null|
|87164|        M|      15.46|       11.89|         102.5|    736.9|         0.1257|          0.1555|        0.2032|             0.1097|       0.1966|               0.07069|   0.4209|    0.6583|       2.805|  44.64|     0.005393|       0.02321|     0.04303|           0.0132|    0.01792|            0.004
168|      18.79|        17.04|          125.0|    1102.0|          0.1531|           0.3583|          0.583|              0.1827|        0.3216|                  0.101|null|
|87880|        M|      13.81|       23.75|         91.56|    597.8|         0.1323|          0.1768|        0.1558|            0.09176|       0.2251|               0.07421|   0.5648|      1.93|       3.909|  52.72|     0.008824|       0.03108|     0.03112|          0.01291|    0.01998|            0.004
506|       19.2|        41.85|          128.5|    1153.0|          0.2226|           0.5209|         0.4646|              0.2013|        0.4432|                 0.1086|null|
+-----+---------+-----------+------------+--------------+---------+---------------+----------------+--------------+-------------------+-------------+----------------------+---------+----------+------------+-------+-------------+--------------+------------+-----------------+-----------+-----------------
---+---------+-----------+------------+--------------+---------+---------------+----------------+--------------+-------------------+-------------+----------------------+----+
only showing top 5 rows

[INFO]: Cleaning data.
[INFO]: Dropped column _c32 for >30% null values.
Testing set:
Features:
+-----+--------------------+
|   id|        normFeatures|
+-----+--------------------+
|85715|[0.00829252170664...|
|86355|[0.00493355405219...|
|86408|[0.00988065436101...|
+-----+--------------------+
only showing top 3 rows

Target(Labels):
+-----+---------+
|   id|diagnosis|
+-----+---------+
|85715|        M|
|86355|        M|
|86408|        B|
+-----+---------+
only showing top 3 rows

21/07/17 00:06:29 WARN Column: Constructing trivially true equals predicate, 'id#10 = id#10'. Perhaps you need to use aliases.
Final Test set (After preprocessing):
+--------------------+-----+
|            features|label|
+--------------------+-----+
|[0.00829252170664...|  1.0|
|[0.00493355405219...|  1.0|
|[0.00988065436101...|  0.0|
+--------------------+-----+
only showing top 3 rows

Accuracy: 0.95
F1-score: 0.95
Area under ROC metric: 0.9938252547082432
[hadoop@ip-172-31-44-117 ~]$ 
```

Fig 6.9 Complete look at the Testing Process

By this we complete the project execution and our model is trained and tested.

Warning: Clean-up!! Make sure to delete the resources and turn them in to the AWS. This may save us the trouble if getting any unexpected cost incurred by running the unused resources. Keep whatever is needed. Thank you.

Chapter 7. Conclusion

We have finally reached the conclusion for this project document. We are now familiar with using Apache Spark and working with it. Also, we have leveraged the power of Cloud resources, specifically on AWS. We now know how to work on AWS Cloud environment and have worked on one of its most widely used infrastructure and service for working with Data mining and Big-data analytics problems, namely the Elastic MapReduce (EMR).

We also worked on another AWS service named the S3, or the Simple Storage Service, and used it as a base for storing our big-data files in the backend, that would later work with our application program for solving a general classification problem, such as predicting Breast-Cancer within patient records.

We have also seen the advantages of using this approach for solving big-data problems, namely working on large-dataset in an efficient manner, performance of training model on large application, using flexibility of the cloud resources and ease in deploying model for testing.

Apache Spark is an open-source unified analytics engine for large-scale data processing [9]. Spark is the outcome of years of endeavored researches carried on large-scale data processing with the aim of making the data mining process and algorithms efficient with the evolving computing infrastructures.

Along with it came the rise of Cloud Computing field, AWS being the pioneer at cloud computing [10], introducing a wide variety of resources for developers and data scientist to leverage the full potential and creativity of designing a needful application. It is now ideal to say that the field of data mining is just at its tip of the ice and will be blooming to become a large one in the near future.

By this we may now conclude the Final Term Project.

References

[1] Original Dataset: Raw form
https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Diagnostic%29

[2] Dataset source: https://www.kaggle.com/uciml/breast-cancer-wisconsin-data
[3] https://www.tdktech.com/tech-talks/data-mining-and-machine-learning/
[4] https://en.wikipedia.org/wiki/Statistical_classification
[5] Details on Breast-Cancer-Wisconsin: https://www.dhs.wisconsin.gov/publications/p01573a.pdf
[6] https://en.wikipedia.org/wiki/Random_forest
[7] https://en.wikipedia.org/wiki/Computer_cluster
[8] https://aws.amazon.com/emr/
[9] https://en.wikipedia.org/wiki/Apache_Spark
[10] Amazon Web Services: The Pioneer in Cloud Computing, casecent.re/p/133541
[11] Spark MLlib: https://spark.apache.org/docs/latest/ml-guide.html
[12] Spark SQL: https://spark.apache.org/docs/latest/sql-programming-guide.html
[11] Data Mining: Concepts and Techniques, Han et al., Elsevier, 2011, ISBN 978-0-12-381479-1
[12] Introduction to Data Mining, Tan et al., Pearson, 2019, ISBN-13: 978-0-13-312890-1