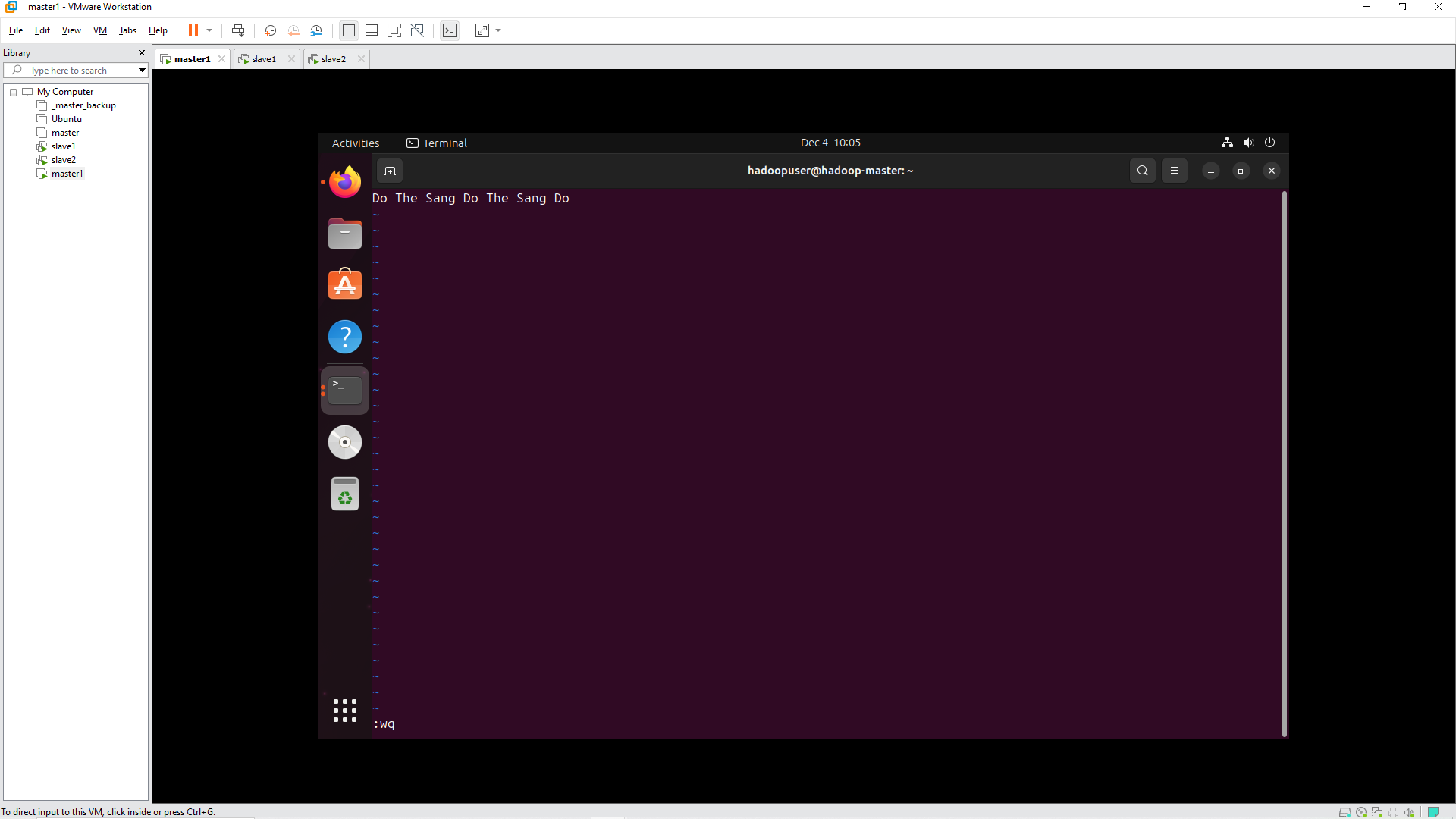
2001203004 – Đỗ Thế Sang

# Tuần 4: Xây dựng thuật toán logistic và mô hình pipeline trong mllib trên hadoop spark.

# Demo bài toán wordCount

Tạo file input.txt



Code wordCount.py

A screenshot of a computer

Description automatically generated

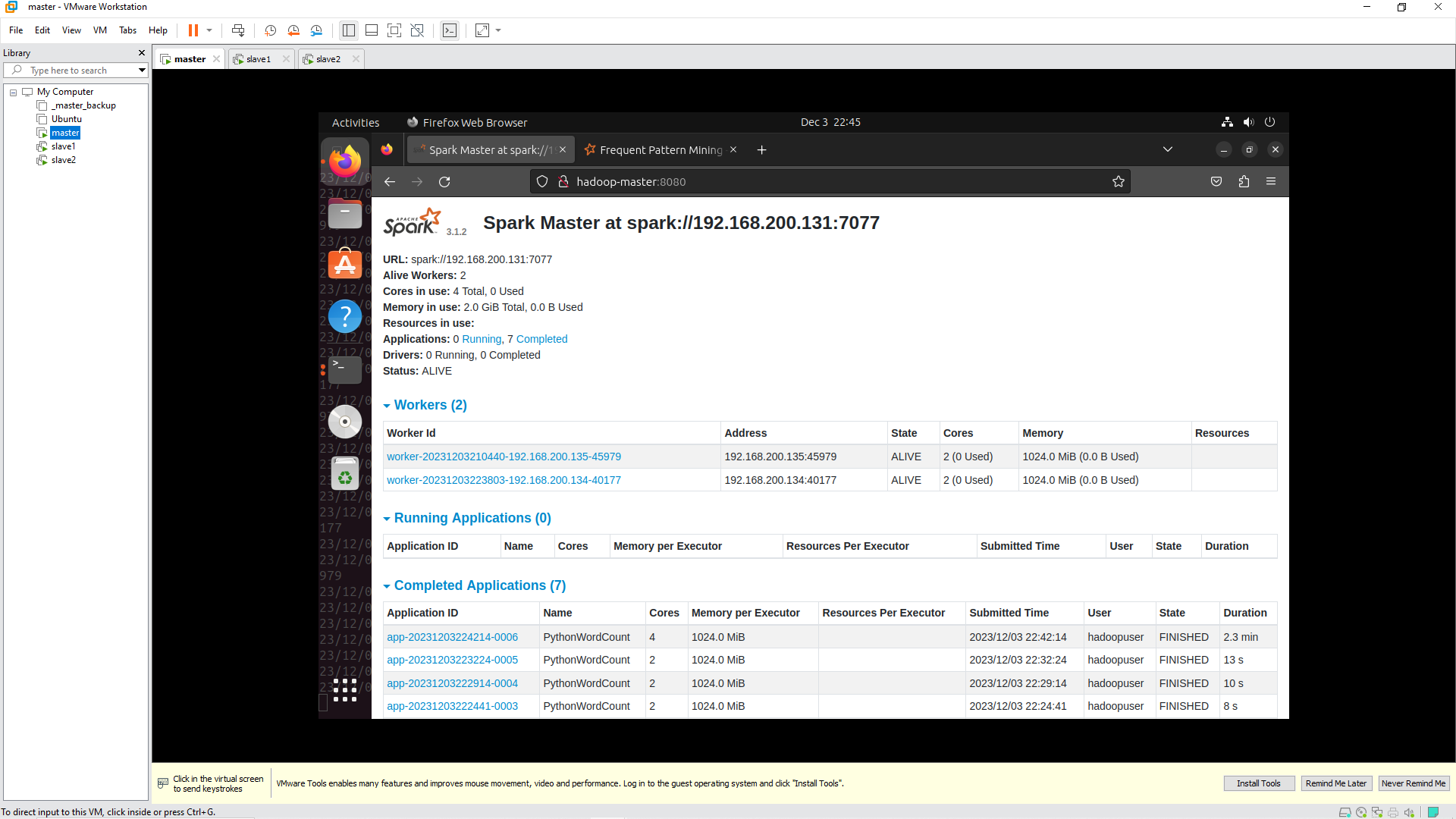
Chạy Spark trên master:

Copy file input này cho các máy master

A screenshot of a computer

Description automatically generated

Chạy Spark trên master và 2 máy slaves:



Submit file wordCount.py và chờ kết quả

A screenshot of a computer

Description automatically generated

Kết quả:

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Nếu file stderr không có ghi nhận lỗi thì code đã chạy thành công, ta trở lại terminal để kiểm tra:

A screenshot of a computer

Description automatically generated

Kết quả như mong đợi!

# Triển khai bài toán phân loại mail theo chủ đề

## Thu thập dữ liệu

A screenshot of a computer

Description automatically generated

*Tổ chức của dữ liệu train*

A screenshot of a computer

Description automatically generated

*Các file trong một chủ đề*

Ta sẽ copy thư mục trên cho các máy slave1 và slave2 để training trên mô hình cluster.

$ scp -r /home/hadoopuser/mini\_newsgroup hadoopuser@hadoop-slave1:/home/hadoopuser/

$ scp -r /home/hadoopuser/mini\_newsgroup hadoopuser@hadoop-slave2:/home/hadoopuser/

A screenshot of a computer

Description automatically generated

## Cài đặt các thư viện cần thiết

A screenshot of a computer

Description automatically generated

## Source code

import findspark

findspark.init()

from pyspark.sql import SparkSession, SQLContext

from pyspark.ml import Pipeline

from pyspark.ml.feature import RegexTokenizer, StopWordsRemover, HashingTF

from pyspark.ml.classification import RandomForestClassifier

from pyspark.ml.evaluation import MulticlassClassificationEvaluator

import os

import matplotlib.pyplot as plt

import numpy as np

import pandas as pd

import math

base\_path = '/home/hadoopuser/mini\_newsgroups'

spark = SparkSession.builder.appName("mini\_newsgroups").getOrCreate()

sqlContext = SQLContext(spark)

data = []

topic2id = dict()

i = 0

for topic in os.listdir(base\_path):

topic2id[topic] = i

i += 1

for doc in os.listdir(os.path.join(base\_path, topic)):

doc\_path = os.path.join(base\_path, topic, doc)

with open(doc\_path, 'rb') as f:

text = f.read().decode('latin-1')

# Check if the topic is already in topic2id, if not, add it

if topic not in topic2id:

topic2id[topic] = i

i += 1

data.append([text, topic, topic2id[topic]])

df = pd.DataFrame(data, columns=['doc', 'topic', 'label'])

msk = np.random.rand(len(df)) <= 0.8

training = df[msk]

testing = df[~msk]

training\_data = sqlContext.createDataFrame(training)

testing\_data = sqlContext.createDataFrame(testing)

def plot\_topic\_distribution(data, color='b'):

topic\_dist = data.groupBy("topic").count()

fig, ax = plt.subplots(1)

x = range(len(topic\_dist.collect()))

labels = [topic[0] for topic in topic\_dist.collect()]

y = [topic[1] for topic in topic\_dist.collect()]

ax.bar(x, y, color=color)

ax.set\_xlabel("Chu De")

ax.set\_ylabel("So Luong")

ax.set\_title("Du Lieu Mail Theo Chu De")

plt.xticks(x, labels, rotation='vertical')

low = min(y)

high = max(y)

plt.ylim([math.ceil(low - 0.5 \* (high - low)), math.ceil(high + 0.01 \* (high - low))])

rects = ax.patches

for rect, y\_i in zip(rects, y):

height = rect.get\_height()

ax.text(rect.get\_x() + rect.get\_width() / 2, height, y\_i, ha='center', va='bottom')

return fig

plot\_topic\_distribution(training\_data)

plt.savefig('/home/hadoopuser/training\_plot.png')

plt.show()

plot\_topic\_distribution(testing\_data, color='r')

plt.savefig('/home/hadoopuser/testing\_plot.png')

plt.show()

rf = RandomForestClassifier(numTrees=50, maxDepth=10, labelCol="label", featuresCol="features")

tokenizer = RegexTokenizer(inputCol="doc", outputCol="words", pattern="\\s+")

remover = StopWordsRemover(inputCol=tokenizer.getOutputCol(), outputCol="words")

hashingTF = HashingTF(inputCol=remover.getOutputCol(), outputCol="features", numFeatures=150000)

pipeline = Pipeline(stages=[tokenizer, hashingTF, rf])

model = pipeline.fit(training\_data)

predictions = model.transform(testing\_data)

evaluator = MulticlassClassificationEvaluator(labelCol="label", predictionCol="prediction", metricName="accuracy")

accuracy = evaluator.evaluate(predictions)

print("Test Accuracy = {:.2%}".format(accuracy))

model.save('/home/hadoopuser/model')

spark.stop()

Trong mô hình, thay vì sử dụng Logistic Regression như thầy, em dùng một mô hình khác là Random Forest. Như vậy pipeline của em sẽ gồm tokenizer, hashingTF, và rf. Phần đọc dữ liệu cũng có thay đổi một chút để tránh xảy ra lỗi. Bài toán sẽ cho ra 2 biểu đồ và một model cho mô hình pipeline.

Lưu source code vào file mail\_clf.py:

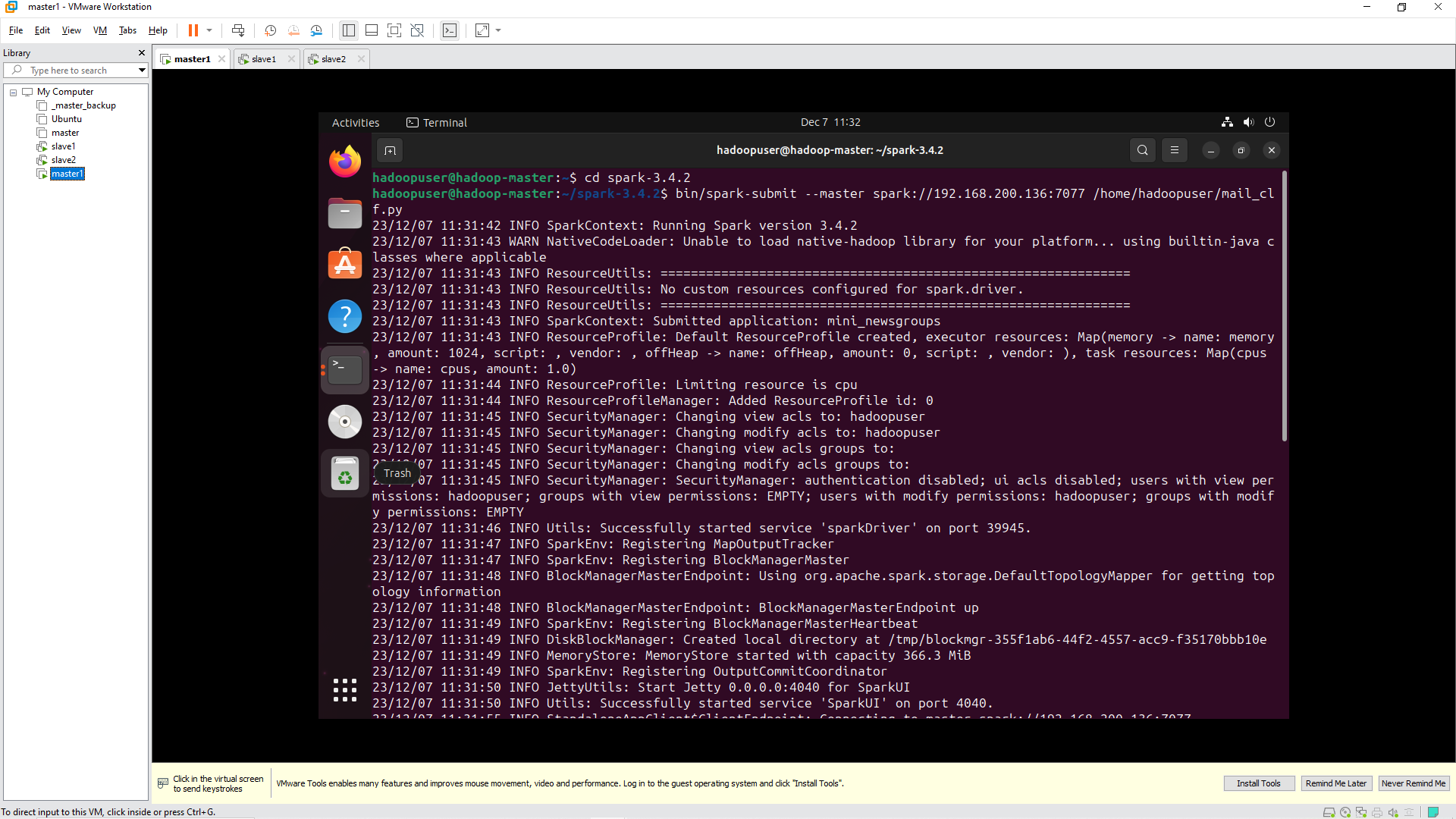
A screenshot of a computer

Description automatically generated

Chuẩn bị các máy master, slave1 và slave2

Sao đó submit code để chạy trên các cluster

$ bin/spark-submit –master spark://192.168.200.136:7077 /home/hadoopuser/mail\_clf.py



Kết quả sau khi chạy:

A screenshot of a computer

Description automatically generated

*Biểu đồ từ file /home/hadoopuser/training\_plot.png*

A screenshot of a computer

Description automatically generated

Biểu đồ của file *file /home/hadoopuser/testing\_plot.png*

Kết quả từ save model, ta được một folder chứa các giai đoạn của pipeline:

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated