**Spam Classifier: Rohitha Gutta**

$SPARK\_HOME/bin/pyspark/Users/rohithagutta/Desktop/SCU\_WORK/spark-2.3.2/zbin/env\_setup.sh: line 1: $: command not found

Python 2.7.10 (default, Oct 6 2017, 22:29:07)

[GCC 4.2.1 Compatible Apple LLVM 9.0.0 (clang-900.0.31)] on darwin

Type "help", "copyright", "credits" or "license" for more information.

2018-12-05 22:36:23 WARN NativeCodeLoader:62 - Unable to load native-hadoop library for your platform... using builtin-java classes where applicable

Setting default log level to "WARN".

To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).

2018-12-05 22:36:24 WARN Utils:66 - Service 'SparkUI' could not bind on port 4040. Attempting port 4041.

Welcome to

\_\_\_\_ \_\_

/ \_\_/\_\_ \_\_\_ \_\_\_\_\_/ /\_\_

\_\ \/ \_ \/ \_ `/ \_\_/ '\_/

/\_\_ / .\_\_/\\_,\_/\_/ /\_/\\_\ version 2.3.2

/\_/

Using Python version 2.7.10 (default, Oct 6 2017 22:29:07)

SparkSession available as 'spark'.

>>>

#import statements

>>>

>>> from pyspark.mllib.regression import LabeledPoint

>>> from pyspark.mllib.feature import HashingTF

>>> from pyspark.mllib.classification import LogisticRegressionWithSGD

>>>

#read spam and non spam emails

>>> spam\_email=spark.sparkContext.textFile("/Users/rohithagutta/Desktop/extra\_credit/data/spam\_email.txt")

>>> nonspam\_email=spark.sparkContext.textFile("/Users/rohithagutta/Desktop/extra\_credit/data/normal\_email.txt")

>>>

#Featurization: Create a HashingTF instance to map email text to vectors of 256 features.

>>>

>>> term\_frequency=HashingTF(numFeatures = 256)

>>>

#map each word of email to a feature by spliting each email record by space

>>> spam\_features=spam\_email.map(lambda rec:term\_frequency.transform(rec.split(" ")))

>>> nonspam\_features=nonspam\_email.map(lambda rec:term\_frequency.transform(rec.split(" ")))

>>>

#Create LabeledPoint for spam and nonspam

>>>

>>> labeled\_spam=spam\_features.map(lambda word: LabeledPoint(1, word))

>>> labeled\_nonspam=nonspam\_features.map(lambda word:LabeledPoint(0, word))

>>>

#forming training set by making union of spam and nonspam emails.

>>>

>>> trainingSet=labeled\_spam.union(labeled\_nonspam)

>>> trainingSet.cache()

UnionRDD[6] at union at NativeMethodAccessorImpl.java:0

>>>

#creating model

>>>

>>> model = LogisticRegressionWithSGD.train(trainingSet)

>>>

#read test mails

>>> test\_mails=spark.sparkContext.textFile("/Users/rohithagutta/Desktop/extra\_credit/data/10emails.txt")

>>>

>>> test\_mails\_rdd=test\_mails.map(lambda line: line.split("\t"))

>>>

#prediction for test mails

>>>

>>> mails\_prediction=test\_mails\_rdd.mapValues(lambda value: model.predict(term\_frequency.transform(value.split(" "))))

>>> mails\_prediction.collect()

[(u'e100', 1), (u'e200', 1), (u'e300', 0), (u'e400', 1), (u'e500', 0), (u'e600', 0), (u'e700', 0), (u'e800', 1), (u'e801', 0), (u'e802', 0)]

>>>

#Output

>>> output=mails\_prediction.mapValues(lambda a: "spam" if a==1 else "nospam")

>>> output.collect()

[(u'e100', 'spam'), (u'e200', 'spam'), (u'e300', 'nospam'), (u'e400', 'spam'), (u'e500', 'nospam'), (u'e600', 'nospam'), (u'e700', 'nospam'), (u'e800', 'spam'), (u'e801', 'nospam'), (u'e802', 'nospam')]

>>>

