Big Data Parallel Programming Project Report

Activity Recognition from Single Chest-Mounted Accelerometer Data Set

Submitted By :- Chandraprakash

Content

Abstract

Activity Recognition is a rising field of research nowadays, the devices have been designed for monitoring day to day activities and to be used apersonal assistant. In this project, the dataset collects data from a wearable accelerometer mounted on the chest and the purpose of the dataset is for Activity Recognition. It provides challenges for identification and authentication of people using motion patterns . This accelerometer Data are collected from multiple sensors in controlled and uncontrolled manners from fifteen participants who are performing seven activities. This kind of movement monitoring approach based on machine learning for the recognition of working, standing, walking, climbing, talking while standing or walking help to identify and mitigate many real time challenges and give a better approach to solve from critical problems.

Introduction

It has been seen that Activity Recognition has reported on systems showing overall good recognition performance, resulting it has been considered as a potential digital technology for e- health systems. In this project we proposed a machine learning AR wearable device setup and a public domain dataset compromising. In the device, data are collected from 15 participants who all are performing 7 activities in their day to day activities and that dataset has been using for identification of challenges faced and authentication of people using motion patterns. The wearable system is easy to use, only need to start-stop the device, and comfortable to bring, reduced form which does not prevent any type of movement.

The relevant information about the device and datasets are :

- Sampling frequency of the accelerometer: 52 Hz
- Accelerometer Data are Uncalibrated
- Number of Participants: 15
- Number of Activities: 7
- Data Format: CSV

- The dataset contains the following attributes :
  Sequential numbering (SNO), x acceleration, y acceleration, z acceleration and label.
- Labels are codified by numbers
  1: Working at Computer
  2: Standing Up, Walking and Going up\down stairs
  3: Standing
  4: Walking
  5: Going Up\Down Stairs
  6: Walking and Talking with Someone
  7: Talking while Standing

Related Works

There are research and studies have been done similar to this project. In theProject "Accelerometers' Data Classification of Body Postures and Movements", they built a wearable device with the use of 4 accelerometers positioned in the waist, thigh, ankle and arm. The design of the wearable, details on the sensors used, they collected data from 4 people in in different static postures; and dynamic movements with which they trained a classifier using the AdaBoost method and decision trees . The wearable device comprised 4 tri-axial ADXL335 accelerometers connected to an ATmega328V microcontroller. All modules were of the Lilypad Arduino toolkit. The accelerometers were respectively positioned in the waist (1), left thigh (2), right ankle (3), and right arm (4). All accelerometers were calibrated prior to the data collection. The calibration consists of positioning the sensors and the performance of the reading of values to be considered as "zero". From the calibration, the read values of each axis during data collection are subtracted from the values obtained at the time of the calibration. The purpose of the calibration was to attenuate the peculiar inaccuracy issues of this type of sensor. Because of this, the sensors were calibrated on top of a flat table in the same position. Another regular type of calibration is the calibration by subject, in which the accelerometers are read and calibrated after positioned in the subjects' bodies. The calibration by subject may benefit the data collection, provided that it enables the obtainment of more homogeneous data. However, it makes the use of the wearable after completion more complex. In the recognition of activities using wearable accelerometers data obtained , a wearable device consisted of 4 accelerometers , and the data collection procedure, extraction and selection of features for the development of a classifier for human activities. The main contributions of this article are:

A comparative table of the researches in the HAR from wearable accelerometers. A wearable device for data collection of human activities. The dataset were the sample of a public domain dataset with 165,633 samples and 5 classes, In order to enable other authors to continue the research and compare the results. In future, the research will be going to include new classes in the dataset and investigate the classifier's performance with the use of accelerometers in different positions and in different quantities.

Methodology

For building a classifier for wearable accelerometers data, took the following steps to enlarge a classifier for the data achieved from 15 participants, they are performing 7 activities in their everyday activities.

1. Data Collection,
2. Data Pre-Processing,
3. Feature extraction,
4. Training & Testing the model
5. Evaluation of Model
6. Hyper Tuning
7. Upload in GCP Bucket

1. Data Collection: Dataset is loaded and changed into the appropriate format i.e. by using transpose if needed. When reading the file, need to set `inferSchema` to true. Moreover need to enable the `header` option to read the columns' names from the file. Discover the data to get insight on it and see the schema

na dimension on the dataset, checl the statistical summary of the attributes, breakdown the data by categorical attribute variable, find the correleation amoung disimilar attributes, if needed combine and make new attributes.

Data Load

```
project = spark.read.load("gs://dataproc-staging-europe-north1-1036817767992-g5iisz7g/BDProject.csv",format="csv", sep=",", infer
```

```
project.show(5)
```

```
+---+--------------+--------------+--------------+-----+
|SNO|x_acceleration|y_acceleration|z_acceleration|label|
+---+--------------+--------------+--------------+-----+
|  1|          2249|          2677|          2046|    1|
|  2|          2270|          2568|          2005|    1|
|  3|          2222|          2565|          2003|    1|
|  4|          2235|          2571|          2074|    1|
|  5|          2205|          2559|          2075|    1|
+---+--------------+--------------+--------------+-----+
only showing top 5 rows
```

```
project.printSchema()
```

```
root
 |-- SNO: integer (nullable = true)
 |-- x_acceleration: integer (nullable = true)
 |-- y_acceleration: integer (nullable = true)
 |-- z_acceleration: integer (nullable = true)
 |-- label: integer (nullable = true)
```

2. Data Pre-Processing : This is most important process i.e. cleaning the raw data i.e. whenever the data are gathered from different sources, it is collected in a raw format and this data is not feasible for the analysis, so certain steps are executed to convert the data into a small clean data and that can be used to train the model. Most Machine Learning algorithms cannot work with missing features, so we should take care of them

Data Cleaning

```
57]: from pyspark.sql.functions import *
from pyspark.sql.functions import when, count, col

project.select([count(when(isnan(c)|col(c).isNull(),c)).alias(c) for c in project.columns]).show()
```

```
+---+--------------+--------------+--------------+-----+
|SNO|x_acceleration|y_acceleration|z_acceleration|label|
+---+--------------+--------------+--------------+-----+
|  0|             0|             0|             0|    0|
+---+--------------+--------------+--------------+-----+
```

```
58]: ##Features Scaling using StandardScaler Estimator

from pyspark.ml.linalg import Vectors
from pyspark.ml.feature import VectorAssembler
from pyspark.ml.feature import StandardScaler

assembler = VectorAssembler(inputCols = ['SNO','x_acceleration','y_acceleration','z_acceleration','label'], outputCol="features"


featuredproject = assembler.transform(project)

scaler = StandardScaler(withMean=True, withStd=True, inputCol="features", outputCol="scaledFeatures")

# Compute summary statistics by fitting the StandardScaler
scalerModel = scaler.fit(featuredproject)

# Normalize each feature to have unit standard deviation.
scaledproject = scalerModel.transform(featuredproject)
scaledproject.select(["features", "scaledFeatures"]).show(5)
```

```
+--------------------+--------------------+
|            features|      scaledFeatures|
+--------------------+--------------------+
```

3. Feature extraction: The data collected from the accelerometers were performed a data preprocessing, with some following instructions. It was generated a 1 second time window, with 150ms overlapping. The samples were grouped and descriptive statistic was used for generating part of the

derivate features. The derivate features of acceleration in axis x, y, and z and of the samples grouped are listed. The module of acceleration of each accelerometer, defined after a statistic analysis comparing the data of 15 participants and dataset generated with all the derived features are further used.

Researching the model that will be best for the type of data: The main objective is to train the best performing model possible, using the pre-processed data.

```
|  2|          2270|          2568|          2005|    1|[2.0,2270.0,2568....|[-1.6053174482499...|(7,[1],[1.0])|
|  3|          2222|          2565|          2003|    1|[3.0,2222.0,2565....|[-1.6052573774994...|(7,[1],[1.0])|
|  4|          2235|          2571|          2074|    1|[4.0,2235.0,2571....|[-1.6051973067490...|(7,[1],[1.0])|
|  5|          2205|          2559|          2075|    1|[5.0,2205.0,2559....|[-1.6051372359985...|(7,[1],[1.0])|
+---+--------------+--------------+--------------+-----+--------------------+--------------------+-------------+
only showing top 5 rows
```

```
va2 = VectorAssembler(inputCols=["scaledFeatures", "label_info"], outputCol='final_features')

temp1 = va2.transform(newproject)

dataset = temp1.withColumn('features', temp1.final_features).select("features","label")
dataset.show(5)
```

```
+--------------------+-----+
|            features|label|
+--------------------+-----+
|(12,[0,1,2,3,4,6]...|    1|
|(12,[0,1,2,3,4,6]...|    1|
|(12,[0,1,2,3,4,6]...|    1|
|(12,[0,1,2,3,4,6]...|    1|
|(12,[0,1,2,3,4,6]...|    1|
+--------------------+-----+
only showing top 5 rows
```

4. Training and testing the model : For training a model, we initially splitted the model into 3 three sections which are 'Training data' ,'Validation data' and 'Testing data. train the classifier using train the classifier using 'training data set', So, during training the classifier only the training and/or validation set is available. The test set will only be available during testing the classifier.

Splitting Dataset to train and test Set

```
[62]: ##Splitting the data in training and Testing dataset

trainSet, testSet = dataset.randomSplit([0.8, 0.2], seed=12345)
print("Training Data Count: " + str(trainSet.count()))
print("Test Data Count: " + str(testSet.count()))

Training Data Count: 38877
Test Data Count: 9903
```

```
[63]: trainSet.groupby('label').agg({'label': 'count'}).show()

+-----+------------+
|label|count(label)|
+-----+------------+
|    1|        6089|
|    6|        4680|
|    3|        5992|
|    5|        5479|
|    4|        4765|
|    7|        6005|
|    2|        5867|
+-----+------------+
```

```
[64]: testSet.groupby('label').agg({'label': 'count'}).show()

+-----+------------+
|label|count(label)|
+-----+------------+
|    1|        1525|
|    6|        1179|
```

5. Evaluation : As model evaluation is an integral part of the model development process. It helps to find the best model that represents our data and how well the chosen model will work. For improving the model, we might tune the hyper-parameters of the model and try to improve the accuracy and also looking at the confusion matrix to try to increase the number of true positives and true negatives.

Modelling Dataset

Logistic Regression

In [65]:
```
import matplotlib.pyplot as plt
from pyspark.ml.classification import LogisticRegression
from time import *
start_time = time()
lr = LogisticRegression(featuresCol = 'features', labelCol = 'label', maxIter=10)
lrModel = lr.fit(trainSet)
trainingSummary = lrModel.summary
trainaccuracy = trainingSummary.accuracy
print("Coefficients: %s" % str(lrModel.coefficientMatrix))
print("Intercept: %s" % str(lrModel.interceptVector))
print("Training accuracy: ",trainaccuracy)
end_time = time()
elapsed_time = end_time - start_time
print("Time to train model: %.3f seconds" % elapsed_time)
```

```
Coefficients: DenseMatrix([[-3.28185377e-05, -5.98566675e-05, -4.76941990e-05,
              -3.74690978e-05, -1.63441157e-04,  0.00000000e+00,
              -4.44696919e-04, -5.51135540e-04, -6.23373474e-04,
              -5.73097144e-04, -5.69242018e-04, -4.91037556e-04],
             [-9.82424326e-01, -1.43028762e-01, -2.56584772e-02,
              -6.78713863e-02, -1.20819337e+00,  0.00000000e+00,
               5.83756685e+00, -1.21820120e+00, -2.02436164e+00,
              -7.54227476e-01, -1.45436449e+00, -5.50822677e-01],
```

6. Hyperperameter Tuning : An important task in Machine Learning is model selection or using data to find the best model or parameters for a given task. This is also called tuning. Tuning may be done for individual Estimators such as LinearRegression or for entire Pipelines, which include multiple algorithms, featurization, and other steps. Users can tune an entire Pipeline at once, rather than tuning each element in the Pipeline separately. MLlib supports model selection tools, such as CrossValidator.

Model Tunning

```
## Hyperparameter Tunning

from pyspark.ml import Pipeline
from pyspark.ml.classification import RandomForestClassifier
from pyspark.ml.evaluation import BinaryClassificationEvaluator
from pyspark.ml.tuning import CrossValidator, ParamGridBuilder
from time import *
start_time = time()

Random_Forest_Classi = RandomForestClassifier(labelCol="label", featuresCol="features", numTrees= 5)
pipeline = Pipeline(stages=[Random_Forest_Classi])
paramGrid = ParamGridBuilder()\
                .addGrid(Random_Forest_Classi.maxDepth, [1, 2, 4, 5])\
                .addGrid(Random_Forest_Classi.minInstancesPerNode, [1, 2, 4, 5])\
                .build()

crossval = CrossValidator(estimator=pipeline,
                          estimatorParamMaps=paramGrid,
                          evaluator=BinaryClassificationEvaluator(),
                          numFolds=10)

cvModel = crossval.fit(trainSet)

predict_test = cvModel.transform(testSet)
predict_test.select('label', 'prediction', 'probability').show(5)

evaluator = BinaryClassificationEvaluator()
test_roc = evaluator.evaluate(predict_test, {evaluator.metricName: "areaUnderROC"})
mc_evaluator = MulticlassClassificationEvaluator(labelCol="label", predictionCol="prediction", metricName="accuracy")
```

7. Result upload in GCP : At the last, the final output will upload into the google cloud platform under the storage bucket . It can be access anytime and from anywhere with supported device.

Uploading output file in GCP

```
In [ ]: import random
        random1 = random.randint(0,20)
        filepath = "gs://myoutput/output" +str(random1)

        df = preductions_best.toPandas()
        csv = df.to_csv(filepath+".csv")
        print("Download file name", filepath)
```

Google Cloud Platform

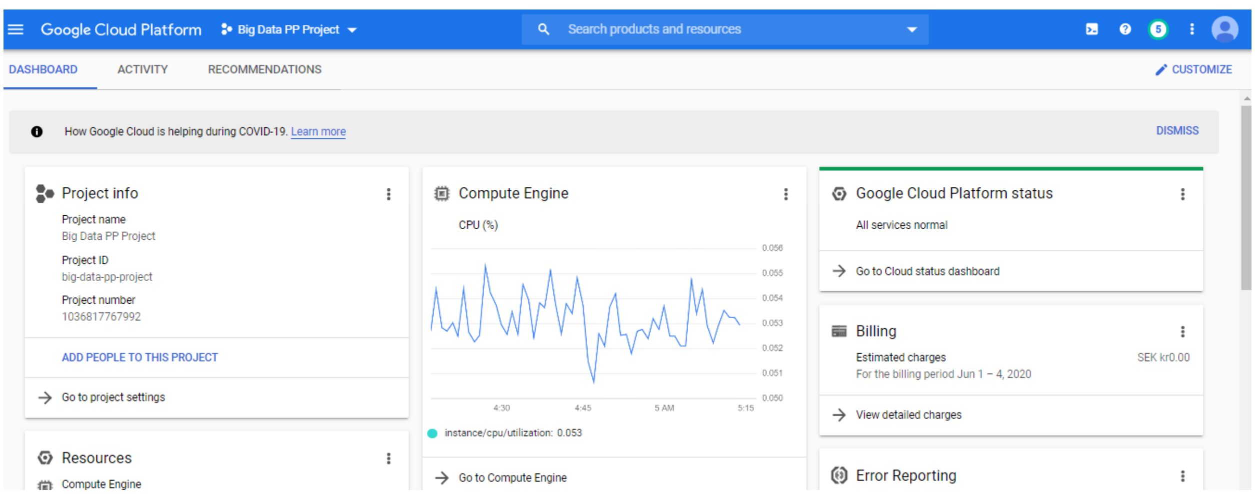
Google cloud platform is offered by google for public cloud computing services. The google platform includes a range of hosted services for storage, computing, networking, bigdata, machine learning, IOTs as well as application development tool, cloud management and security management, which run on google hardware.

Google also offers an online file storage web service called as Google Cloud Storage. It stores objects into storage buckets and each bucket are identified by unique user assigned key. Nowadays, it is very popular and demanding.

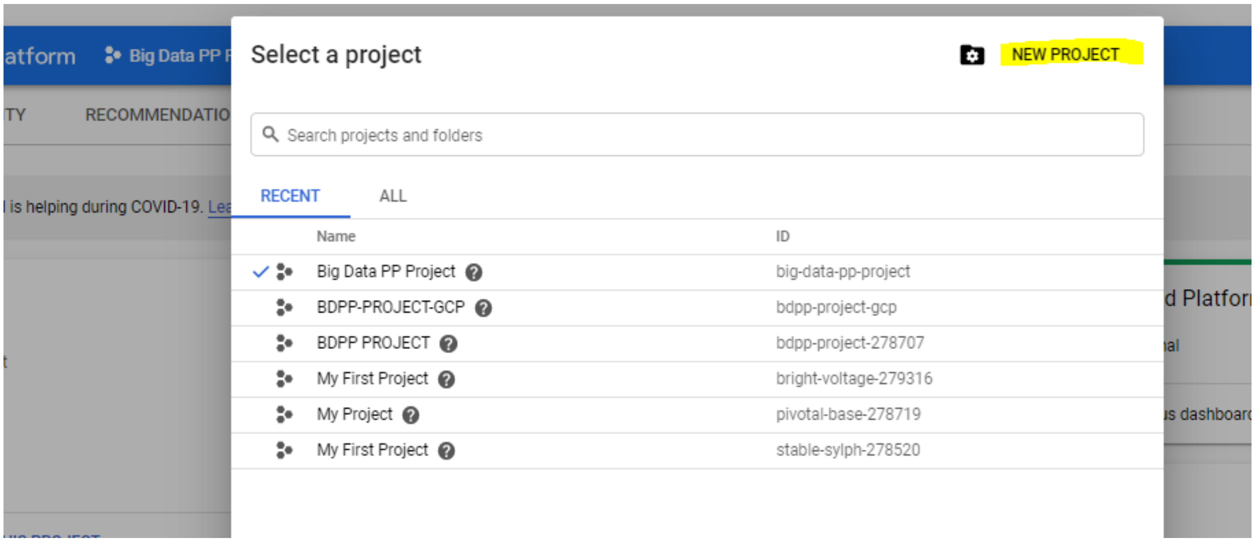
In our project also, we have used Google Cloud Platform for running the projecvt output and storing the result in google cloud storage bucket. It was great experiences working in the GCP.

The below following steps have been followed for running and storing the project output results in Google Cloud.

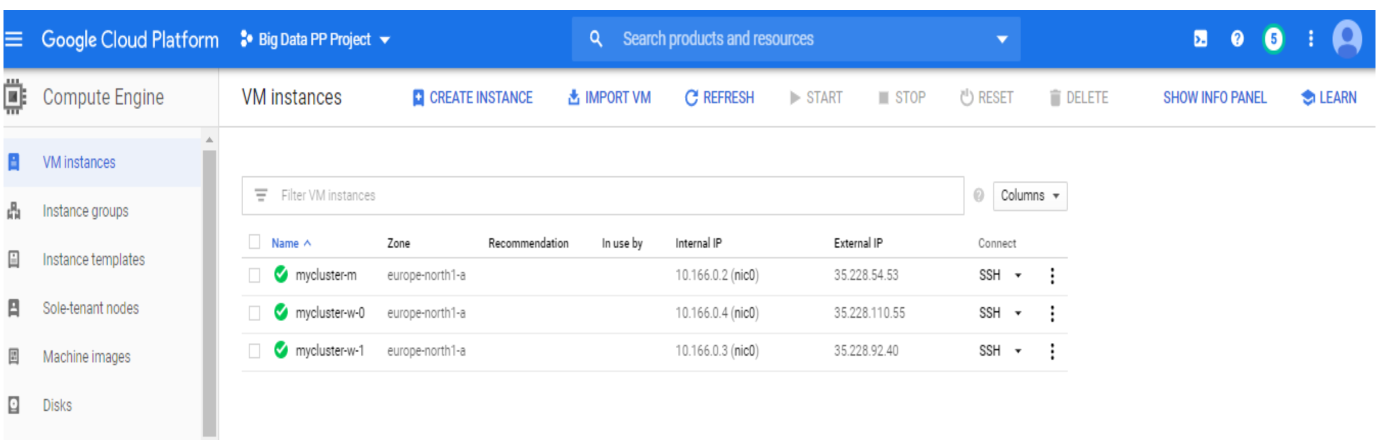
Step 1 : Created GCP account, as it is not free, so 50 dollar was provided by University for the initial run and storage. Moreover, it can also be created a free tier account with 300 dollars GCP credits to spend on it over the next 12 months. The GCP dashboard looks like as below :



Step 2: Create new Project in creation order to work in GCP project creation is required.

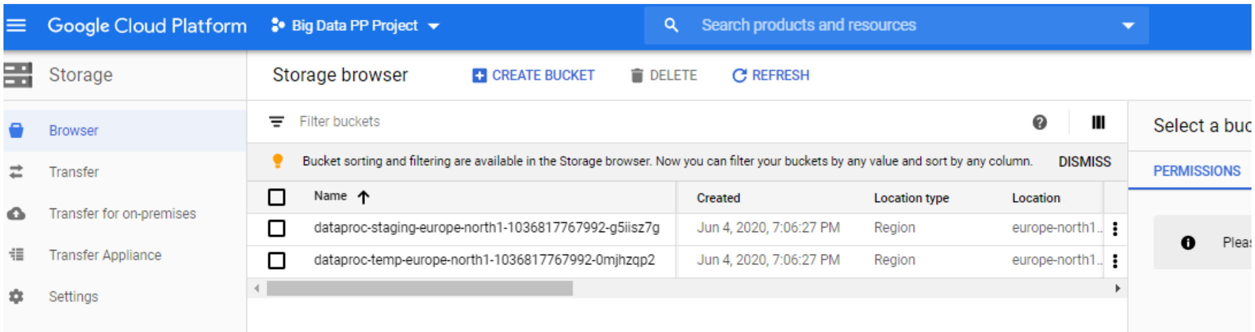


Step 3: Creating Cluster as 1 master and 2 worker with providing all the required details such as region, machine type etc, then the VM instance will be created.

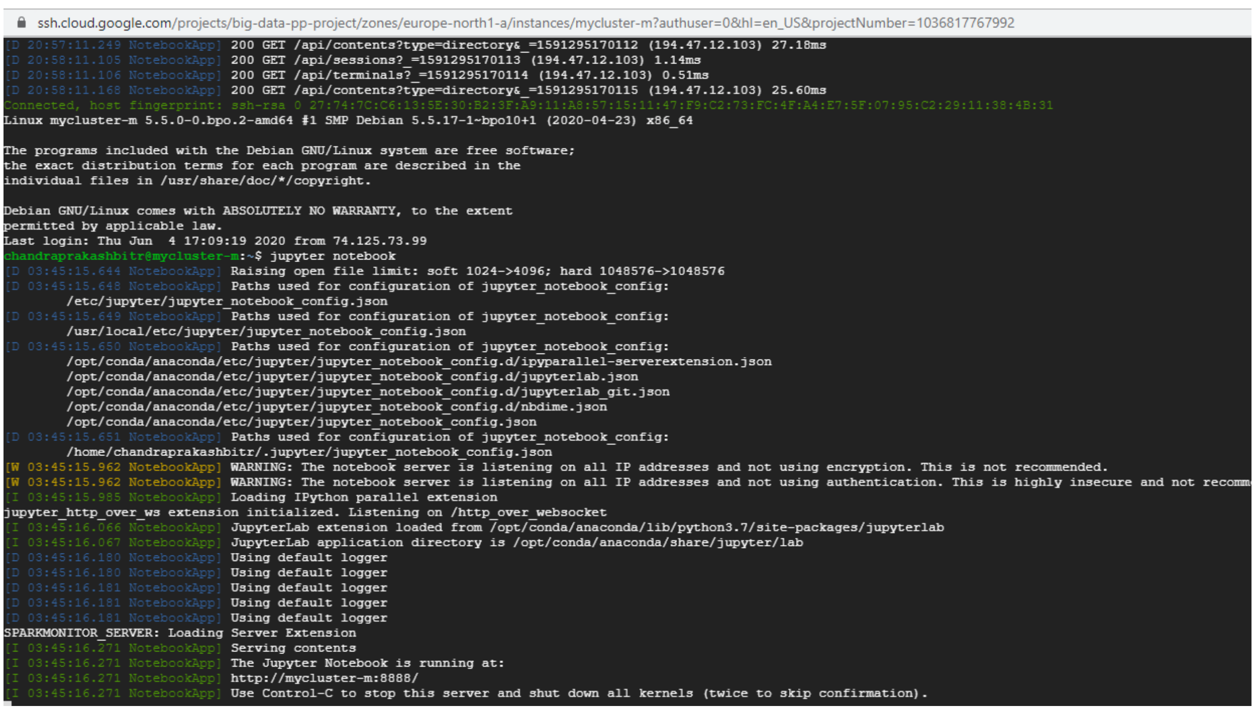


Step 4: Creating Firewall rules to allow or deny the traffic to and from VM instances based on the configuration speicified and make external IP as Static to access the Jupyter notebook.

Step 5: Creating Storage Bucket for storing the output result.

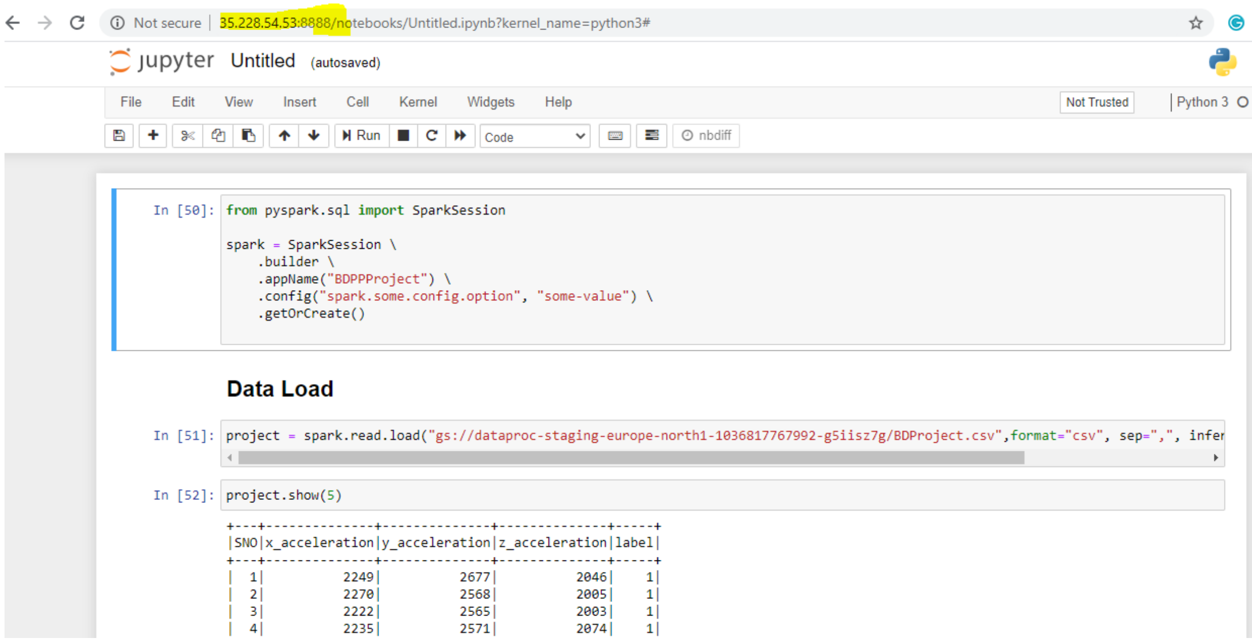


Step 6: Configure Jupyter notebook with VM server by entering the commands on SSH terminal.

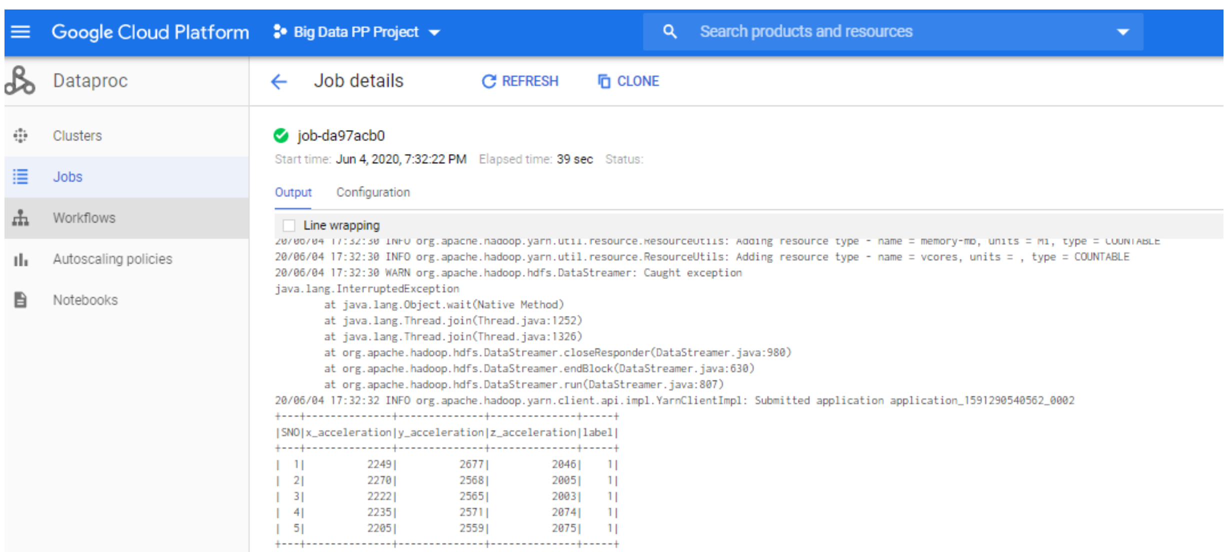


Step7 : Launch Jupyter notebook in the browser by providing the external ip address and the port number configured.

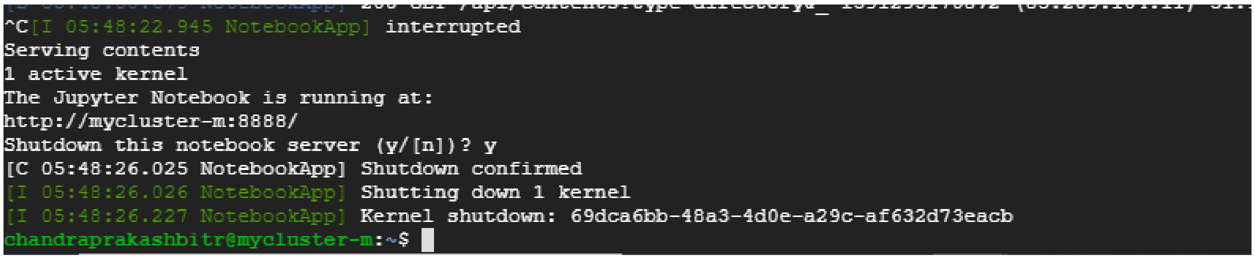
```
http://<external-ip-address>:enter< your port number without angular
brackets>
```

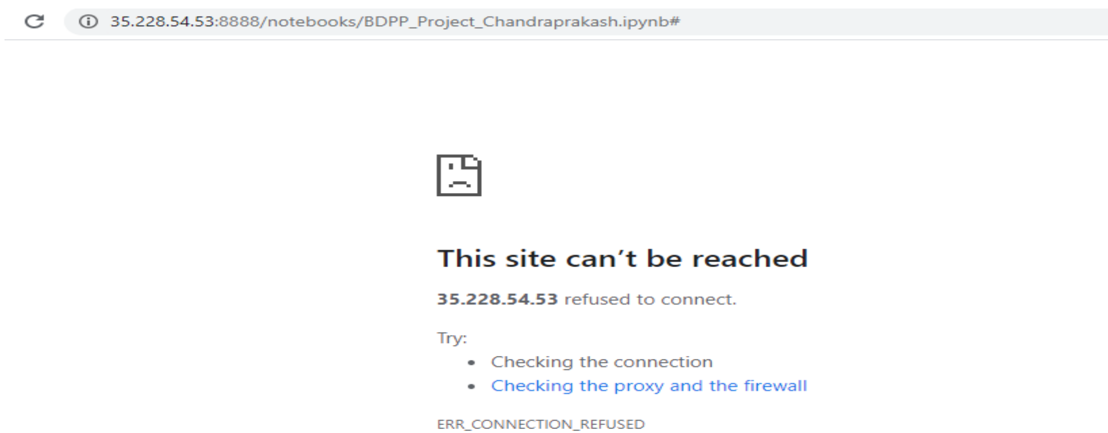


Step 8 : Create Job System for running or scheduling project job.



Step 9: For shutdown the Jupyter notebook server, write command in SSH terminal CTL 'C ' and give yes command, Jupyter notebook will be shoutdown and not accessible.





Results with Different Model:

I have checked the dataset with different models and below are the output result got after performing all the steps :

S.No.	Model Name	Test data Accuracy	Model Tunning Accuracy	Time Taken to run in local machine	Time Taken to run in GCP
1	Logestic Regression	96.12	99.88	14.26sec	7.48Sec
2	Decision tree Classifier	98.56	98.91	12.92Sec	4.91Sec
3	Random forest Classifier	99.9815	99.9872	11.28Sec	7.23Sec

If the model has more accuracy, the model is best for that dataset, so we have checked with three models, started with logistic regression, In which got 96.12 Accuracy which is lesser than the rest of the models. Moreover, when checked with Decision Tree Classifier model, It gave 98.56 accuracy which is higher than the LR Model, then checked for Random Forest classifier got 99.98 % accuracy which was the highest than the LR and DT model. After that hypertunning was performed on LR, DT and Random forest Classifier and found that Random forest Classifier is most suitable for the provided data set among other models for this project.

Conclusion

Since majority of health issues are directly proportional to the present lifestyle or day to day activities, this kind of movement monitoring approach based on machine learning for the recognition of working, standing, walking, climbing, talking while standing or walking help to identify and mitigate many real time challenges and give a better approach to solve from critical problems. In the machine learning model, Random Forest classifier is the best model for the provided data set after measuring and recording the activities of 15 participants by Activity Recognition from Single Chest-Mounted Accelerometer.

References

1. http://archive.ics.uci.edu/ml/datasets/Activity+Recognition+from+Single+Chest-Mounted+Accelerometer
2. https://www.researchgate.net/publication/221258784_Human_Activity_Recognition_from_Accelerometer_Data_Using_a_Wearable_Device
3. http://groupware.les.inf.puc-rio.br/public/papers/2012.Ugulino.WearableComputing.HAR.Classifier.RIBBON.pdf
4. https://tudip.com/blog-post/run-jupyter-notebook-on-google-cloud-platform/

--XXXXXXXX--