CS551G_Assessment2_ZHIXI_TANG_52097136

STUDENT: ZHIXI TANG

STUDENT ID: 52097136

Index

Abstract

This report includes three parts, the first part is a short essay to describe the distributed Learning Big Data Ecosystem. The second part is an experiment report of apache spark implementations, the experiment is consisting of 5 tasks. The third part is the bonus task but not complete, I wrote the codes but did not get the correct result.

In this report I will show some code snippets I think it is important and essential to interpret my solution. For more details information about the source code please kindly check my .ipynb file.

Task 1: Description of Distributed Learning Big Data Ecosystem

Essay Abstract

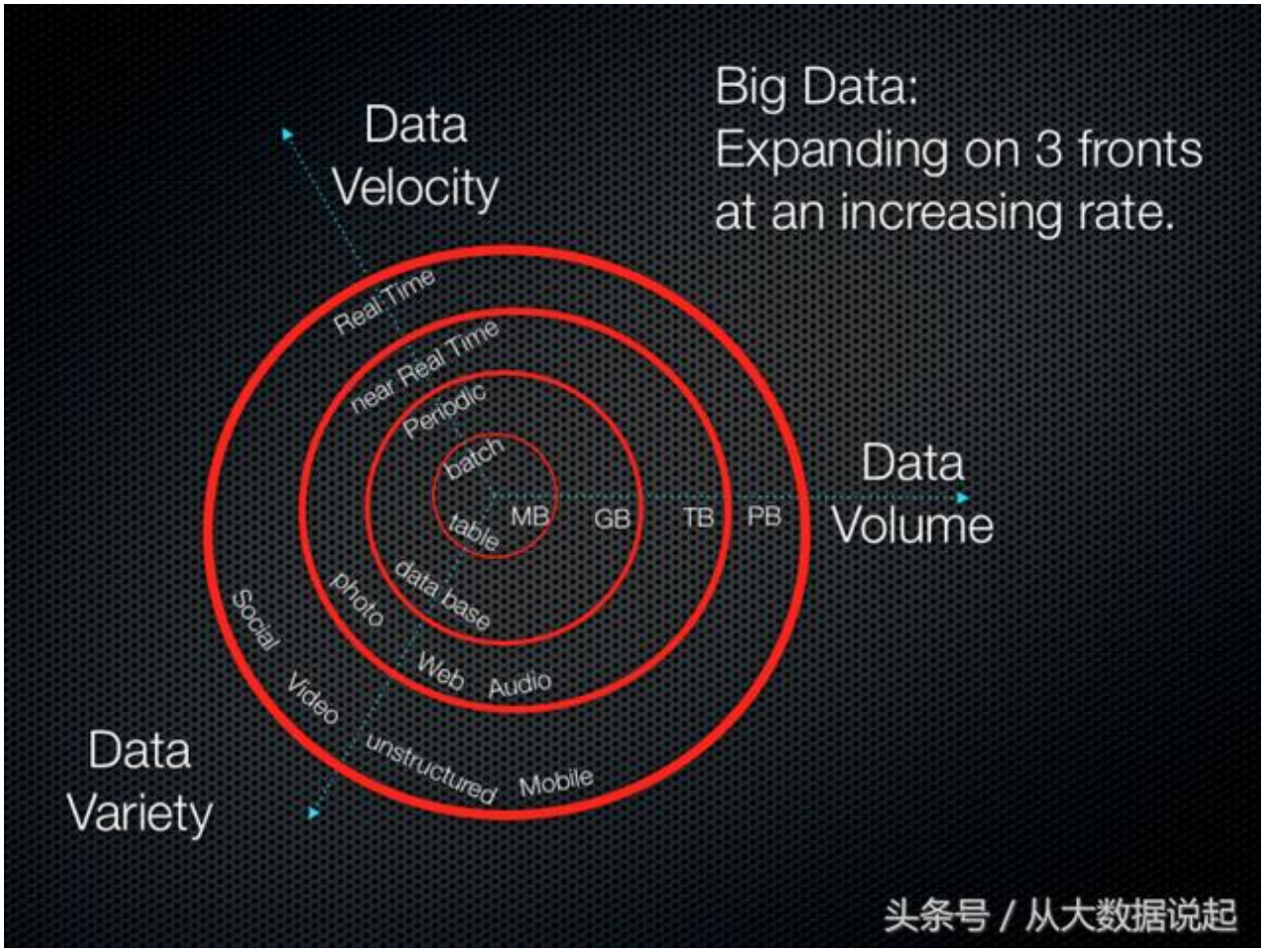
This is an essay of an overview of Distributed Learning Big Data Ecosystem. The first part described what is the Big Data and what is the Big Data Ecosystem. The second part is the main structure of Apach Hadoop, a very popular project for Big Data Ecosystem. The third part is the introduction of Apach Spark, a distributed computation framework. The Last part is a brief instroduction of the implement and deployment of Big Data Ecosystem, and brief described some features of Kubernetes, a Docker container.

1. What is Big Data and Distributed Learning Big Data Ecosystem

Nowadays, among the activities that humans need to use computers and the Internet, a large magnitude of various types of data set is generated, interacted with, and stored including standard data (i.g, generated by computer systems) and nonstandard data (i.g, generated by humans). Moreover, these data sets cannot be processed by traditional databases software and computer techonologies. Therefore, we call this kind of datasets as Big Data. Typically, Big data has the following features:

- Volume: Huge amount of data
- Variety: There are various of types of data
- Velocity: The speed of data processing should be very fast so that the user can make full use of it in time.

The challenge of big data management comes from the expansion of all three features. As the following is a diagram of it:



Moreover, big data also has other features:

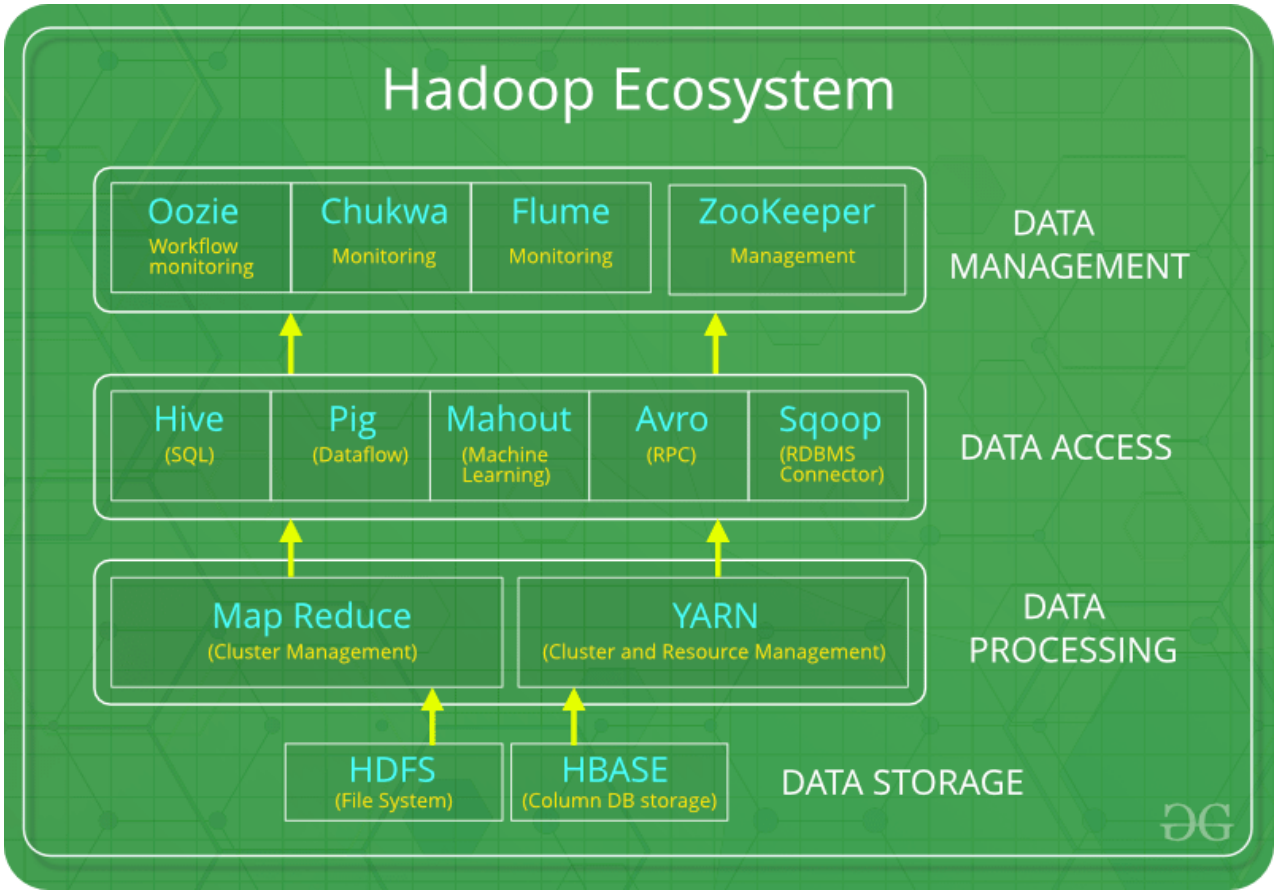
- Veracity: The quality of data.
- Value: Added value of big data to a particular domain.
- Variablity: Data is dynamic instead of static.
- Visualization: A good representation of data.

Therefore, in order to acquire, store, manage, and analyze big data, we need to introduce the technology of the distributed learning big data ecosystem. It can make users coding and running distributed applications to process large batches of data.

2. Main Structure of Big Data Ecosystem

Typically, the Big Data Ecosystem architecture includes storage, computation, query, and data mining and learning.

The most common Big Data Ecosystem project is Apache Hadoop. The diagram of this system is shown as below:



(*images resource: https://media.geeksforgeeks.org/wp-content/cdn-uploads/HadoopEcosystem-min.png*)

The main architectures of Traditional Hadoop are:

- Hadoop Distributed File System (HDFS): it saved the files from all the DataNodes on the cluster. HDFS is fault-tolerant and resilience, can be deploied on low-cost hardware, and provide high throughput data.
- MapReduce: It is a distributed parallel computation model to process big dataset.
- Hive: Toolkit of database
- Hbase: Distributed database
- YARN(Yet Another Resource Negotiator): YARN is the component helps managing the resources across the clusters, it performs scheduling and resource allocation for the Hadoop.

However, the traditional Hadoop framework has problems in computing speed and compatibility with other computing engines. Therefore, developers improved Hadoop to make it has some other functions. The new version Hadoop can be combined with other computing engines (such as Apache Spark), which greatly improves the data processing capabilities of the entire cluster. In the next part I will introduce Apache Spark, a faster cluster computation framework.

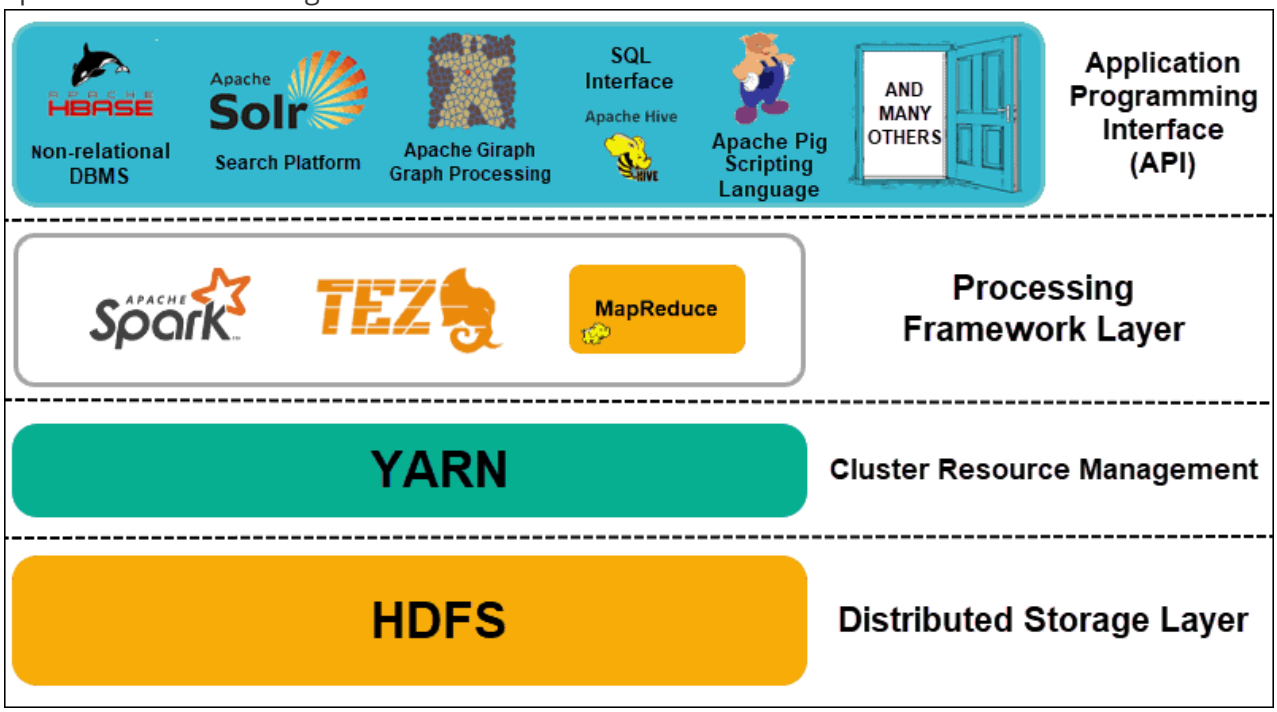
3. Apache Spark

Apache Spark is a framework of cluster computation. The computational output can be saved in memory, in this case, Spark doesn't have to read the data from HDFS. Therefore, the computation speed of Spark is faster than Hadoop MapReduce. The core elements of Spark are:

- Spark Core and RDDs: The core is the foundatioin of the overall project, it provides functions of distributed task dispatching, schema, and basic I/O. RDDs is a data collectioin with fault-tolerance which can be paralleled executed.
- Spark SQL: It provide the query to constructed and semi-constructed data. Programmer can use Python, Java, Scala for execution.
- Spark Streaming: it executes the data streaming analytics based on the fast scheduling capability of the Spark core.
- MLlib (Machine Learning Library): it is a machine learning framwork on the top of Spark core. It includes many algorithms of machine learning
- GraphX: It is a distributed graph processing framwork.

Spark provides many APIs to developers using various programming languages. For Python, Spark provides Pyspark moduel which is consisting of four sub-modules such as pyspark.sql, pyspark.streaming, pyspark.ml, pyspark.mllib.

In the last part, we mentioned that Hadoop can be used with other computation framwork like Spark. As below is a diagram of it:



(*images resource: https://phoenixnap.com/kb/wp-content/uploads/2020/04/hadoop-ecosystem-layers.png*)

Advantages of Spark:

- Faster computatioin speed
- Abundent moduels: Spark Core, Spark SQL, Spark Streaming, MLlib, GraphX
- Support various recouse manager: such as Hadoop YARN.
- Easier way of operation.

4. Implementation and Delopyment

After we introduce the concept and technology architecture of Big Data Ecosystem, how to deploy it on hardware is an important issue. Because the computing resources consumed by the different system. Hereby, we introduce the concept of Docker Container.

After the introduction of containers, we can deploy the Big Data Ecosystem on hardware, and different container frameworks support schedulars with different levels. The current popular container scheduling architectures are Swarm, Mesos, and Kubernetes. Mesos has a relatively good support for big data applications. Spark can run on Mesos and Kubernetes. The major big data application running on Kubernetes are Spark applications. Here we will briefly introduce the features of Spark on kubernetes.

Comparing with Spark on YARN and Mesos, the advantages of Spark on Kubernetes are:

- Kubernetes original scheduler: the schedular can be scheduled directly, and the resources managed by the scheduler can be shared with other applications.
- Resource isolation: The scheduler dispatch each user a namespace to limit the resource quota of users.
- Resource distribution: The scheduler can limit the resource for each spark task.
- Monitoring: Kubernetes can monitor and calculate the resource consumption for each user who submitted the task.
- Log: In Kuberentes, all the logs of applications will be collected, then user can check the log according to the corresponding labels.

Task 2: Develop distributed models in apache spark to classify gas turbines

Preparations

Firstly, we have to upload the provided dataset on Google Colab, then we will install Java environment, Apache Spark, and set up the related configurations. These preparations have been taught in the practical session.

Secondly, we have to load the dataset to Spark, since PySpark can only read dataset from .json, .csv file. I will use Pandas read the .xlsx file as pandas.DataFrame then convert it to Spark.DataFrame. The code snippet is shown below:

```
# read the file via pandas
import pandas as pd
df = pd.read_excel("/content/CS551G_DMV_Assessment_2_Dataset.xlsx")
# convert Pandas.DataFrame to Spark.DataFrame
spark_df = spark.createDataFrame(df)
spark_df.show()
```

Subtask 2.1

The Task 2.1 is creating a table showing the sumary information of this dataset.(mean values, range, standard deviations, min/max values, median values, and 25%/50%/75% percentile values. My solution is shown as the following:

- Using spark.DataFrame.describe() function, it will return a spark.DataFrame include the summary information of counts, mean, standard deviation, min, max.

- From the above function we can know the min and max value of each column, so we can get $range = max - min$. We will create a new spark.DataFrame to store these values then combine with the previous spark.DataFrame.

- Using mathmatic calculation functions of pandas.DataFrame *(i.g, df.median(),df.quantile(), and so on)*. We can get the rest values of summary information. Then we will create a new spark.DataFrame, then combine with the previous Spark.DataFrame.

I wrote three functions to calculate the range, median, and quantiles, then finally I use a function called "description" to call all these functions and return a table as required. The code snippet is shown below:

```
a = description(spark_df, df)
a.show()  # show the table
```

The part of table is shown as below, for the complete table please kindly check my .ipynb file.

```
+-----------+-------------------+-------------------+-------------------+-----
|    summary|Temperature_Sensor_1|Temperature_Sensor_2|Temperature_Sensor_3|Flow_
+-----------+-------------------+-------------------+-------------------+-----
|      count|                996|                996|                996|
|       mean|  4.9995738935742935|   6.379273152610439|   9.228112114457828| 7.3
|     stddev|   2.764855518719287|   2.312568802265547|   2.532172890894333| 4.3
|        min|             0.0082|             0.0403|           2.583966|
|        max|            12.1298|            11.9284|            15.7599|
|      range|  12.121599999999999|            11.8881|           13.175934|
|     median|             4.8811|  6.4704999999999995|   9.347999999999999|
|25% quantile|          2.8921195|            4.93175|   7.511399999999999|3.43
|50% quantile|             4.8811|  6.4704999999999995|   9.347999999999999|
|75% quantile|          6.7945575|   8.104500000000002|            11.0468|
+-----------+-------------------+-------------------+-------------------+-----
```

Subtask 2.2

In Task 2.2, we will draw two plots and interpret what information we can get from these plots. The first one is a boxplot containing two classes (normal and abnormal) vibration_sensor_1 values. The second one is a scatterplot that also containing two class values of vibration_Sensor_1 against vibration_Sensor_2.

Solutions:

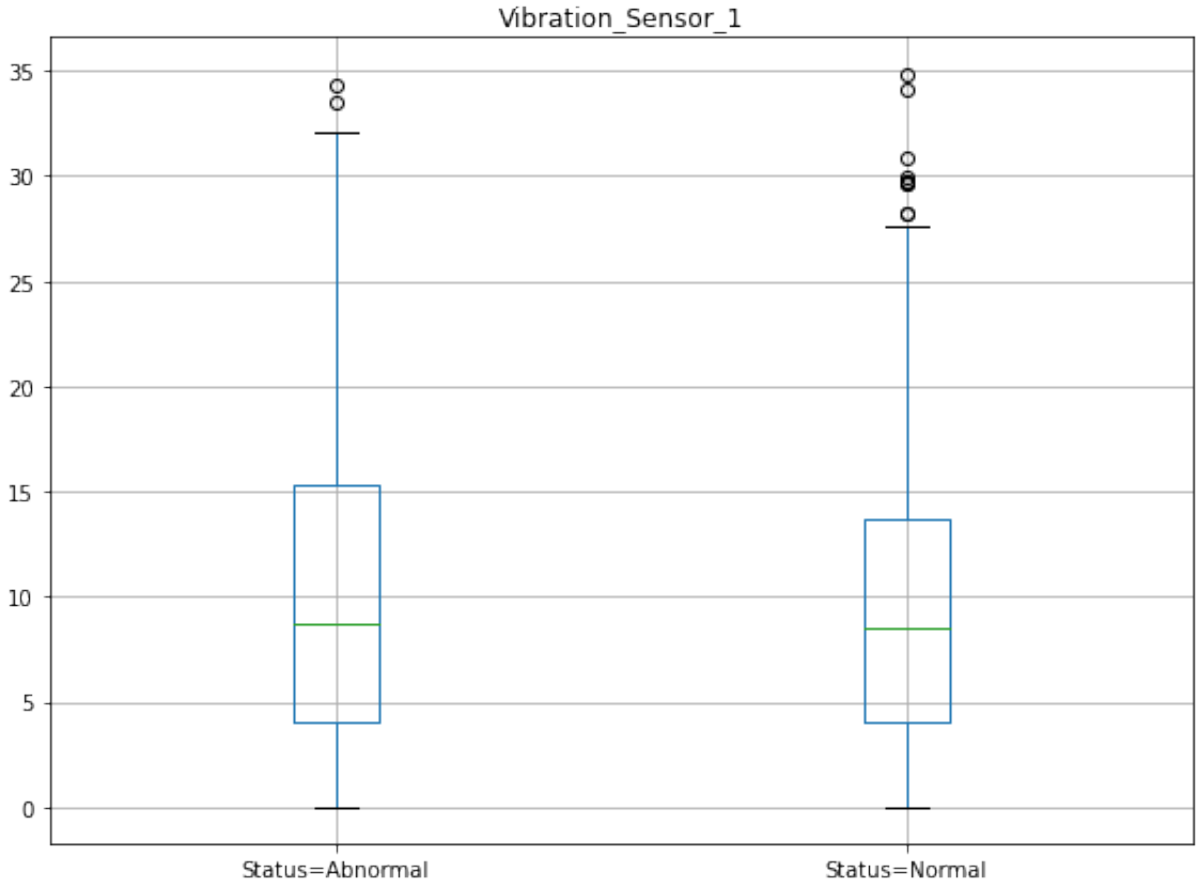
For the first plot, as the assessment mentioned, we can use HandySpark. There is a function in HandySpark called stratify. We use this function to stratify the whole HandSpark.DataFrame according to Status (normal or abnormal). Then we can plot as required.

For the second plot, there are two solutions:

1. We use the function of HandySpark.cols to extract the values from columns "Status", "Vibration_Sensor_1", and "Vibration_Sensor_2". It will return a pandas.DataFrame. Then we use SeaBorn creating a SeaBorn.FacetGrid object to plot the figure.
2. Firstly, we use Spark.filter() to create two spark.DataFrames classified by the "Status" column. Then we convert them to HandySpark DataFrame. Extracting the values of "Vibration_Sensor_1", and "Vibration_Sensor_2" by HandySpark.cols, we will get two pandas.DataFrames. Then we use matplotlib to plot two figures then combine them together.

The code snippet and first plot are shown below:

```
# make sure handspark is installed on the computer
from handyspark import *
h_df = spark_df.toHandy() # convert Spark.DataFrame to Handy.DataFrame
h_df.stratify(['Status']).cols['Vibration_Sensor_1'].boxplot(figsize=(8,6))
# plot the figure
```



Obtained Information (boxplot):

(*The numbers in the below content is approximately recognized from the figure*)

From the above figure, we can tell that the range of Vibration Sensor 1 abnormal values is from 0 to 32.5. Any values are greater than 32.5 would be counted as the outlier (There are two), the 25% quantile value of this dataset is around 4.1. The median value is around 8. 75% quantile is around 15.5.

Similarly, in the dataset of normal vibration sensor 1, the value range is from around 0 to 27.5. Any values are greater than 27.5 would be counted as the outlier (There are more than 4 outliers). The 25% quantile of this dataset is around 4.1, the median value is around 8, the 75% quantile is around 13.5.

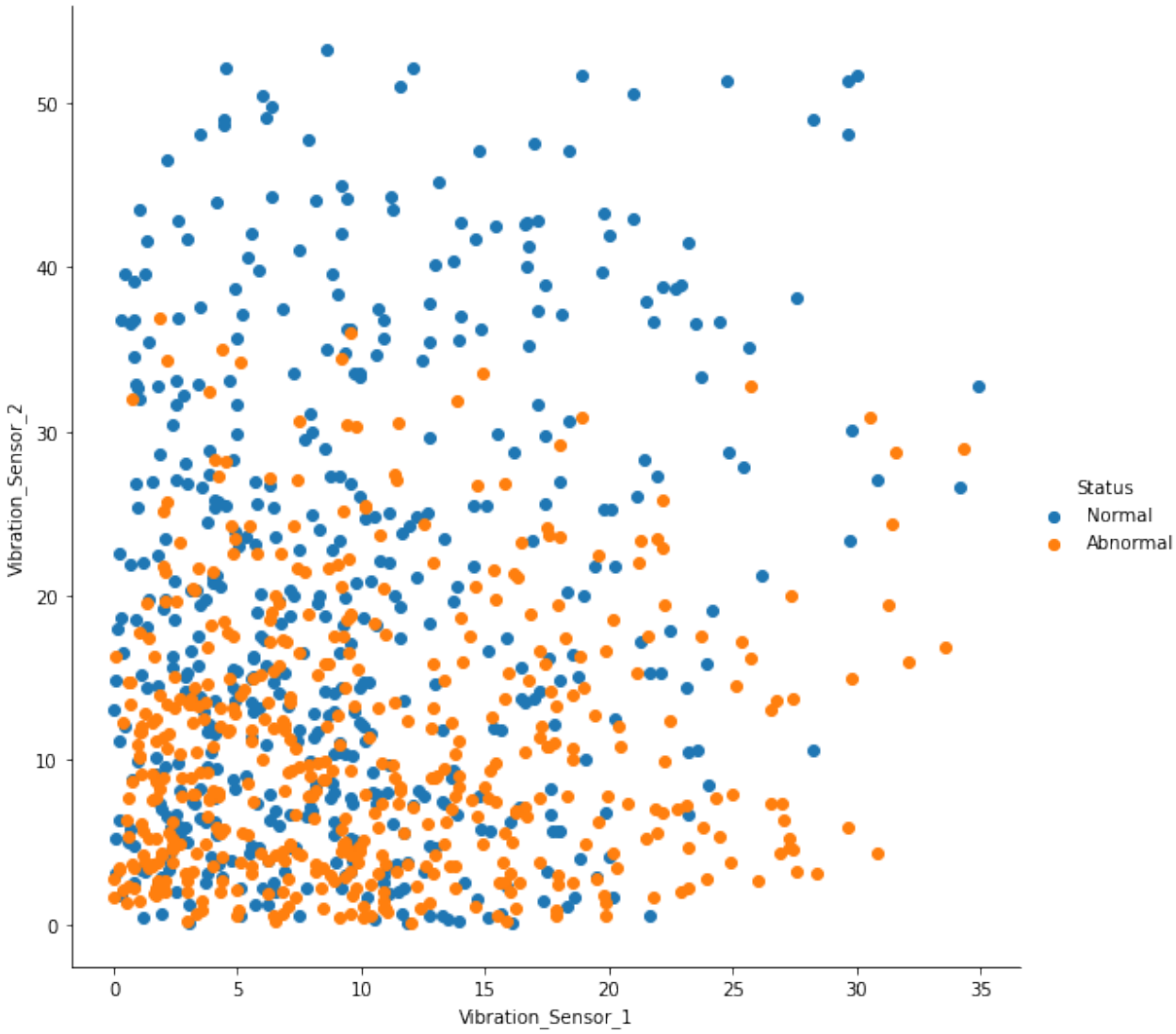
Conclusion:

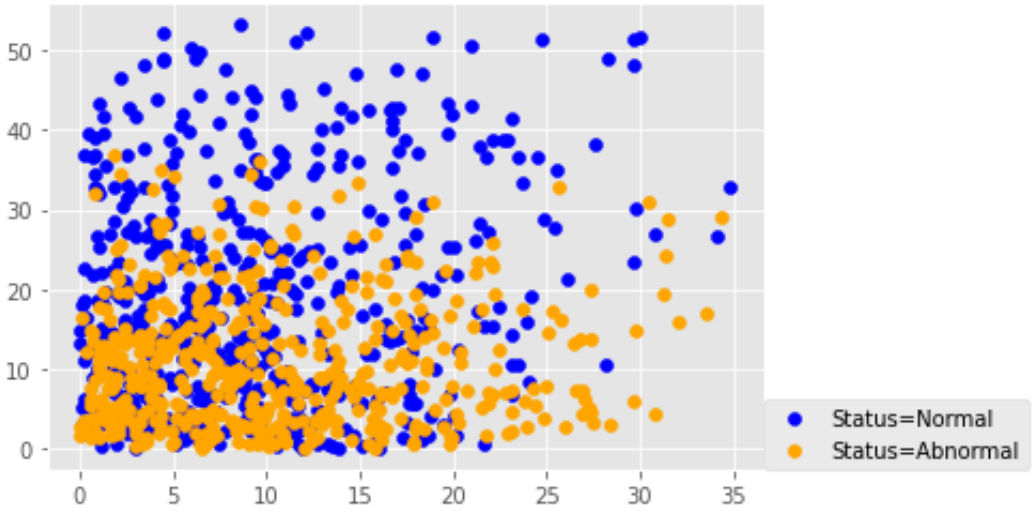
The normal vibration sensor 1 has more outliers than the abnormal sensor. normal sensor's value range is smaller than the abnormal sensor. Their minimum values, median values, and 25% quantiles are almost the same but the maximum value and 75% quantile of Abnormal sensor is greater than Normal sensor.

The code snippets and scatterplots for both solutions are shown below:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

```

```
# Solution - 1
status = ['Normal', 'Abnormal']
data = h_df.cols[['Status', 'Vibration_Sensor_1', 'Vibration_Sensor_2']][:]

fg = sns.FacetGrid(data=data, hue='Status', hue_order=status, height=8, aspect=1)
fg.map(plt.scatter, 'Vibration_Sensor_1', 'Vibration_Sensor_2').add_legend()
plt.show()

# Solution - 2
import matplotlib.pyplot as plt
plt.style.use('ggplot')
import pandas as pd
import numpy as np

# Split the dataframe by Status = 'Normal' and 'Abnormal'
d1 = spark_df.filter(spark_df.Status == 'Normal').toHandy().cols[['Status', 'Vibration_Sensor_1', 'Vibration_Sensor_2']][:]
d2 = spark_df.filter(spark_df.Status == 'Abnormal').toHandy().cols[['Status', 'Vibration_Sensor_1', 'Vibration_Sensor_2']][:]

# drawing image of normal and abnormal status individually.
plt.scatter(d1['Vibration_Sensor_1'], d1['Vibration_Sensor_2'], color='blue', label='Status=Normal')
plt.scatter(d2['Vibration_Sensor_1'], d2['Vibration_Sensor_2'], color='orange', label='Status=Abnormal')

plt.legend(loc=(1,0))
plt.show()
```





Obtained Information (boxplot):

The numbers in the below content is approximately recognized from the figure

Here we say Vibration Sensor 1 as VS1, Vibration Sensor 2 as VB2.

As we can see from the above figure, regarding the distribution of VS1 and VS2 values, the overall distribution of normal values is looser than the abnormal values. For the values of both statuses, when VS1 increases, its distribution becomes looser. For a fixed VS1 interval, the distribution of normal values is more uniform than abnormal values. The values of abnormal are mainly concentrated in the range of $VS1 \in [0, 15]$, the corresponding interval of VS2 is $[0, 20]$

Subtask 2.3

For this task, we will use all the columns but column "Status" as the input features to generate a feature to classify the Status, the following are the steps:

1. As we will classify the dataset as normal (represented by 1) and abnormal (represented by 0). so we will adding an additional column in the Spark.DataFrame named "Status_Value", the values of this column depend on its status (1 for normal, 0 for abnormal). If we print the schema, it would show as below:

```
root
 |-- Status: string (nullable = true)
 |-- Temperature_Sensor_1: double (nullable = true)
 |-- Temperature_Sensor_2: double (nullable = true)
 |-- Temperature_Sensor_3: double (nullable = true)
 |-- Flow_Rate_Sensor_1: double (nullable = true)
 |-- Flow_Rate_Sensor_1.1: double (nullable = true)
 |-- Flow_Rate_Sensor_1.2: double (nullable = true)
 |-- Pressure _sensor_1: double (nullable = true)
 |-- Pressure _sensor_2: double (nullable = true)
 |-- Pressure _sensor_3: double (nullable = true)
 |-- Vibration_Sensor_1: double (nullable = true)
 |-- Vibration_Sensor_2: double (nullable = true)
 |-- Vibration_Sensor_3: double (nullable = true)
 |-- Status_Value: double (nullable = true)
```


2. When I was trying to create a Spark.Assembler, the Python compiler reported an error about the data type of column "Flow_Rate_Sensor1.1" and "Flow_Rate_Sensor1.1", the compiler indicated the needed data type is struct type but I think both data types should be Double and they are indeed Double, so I changed their column names as "Flow_Rate_Sensor_2" and "Flow_Rate_Sensor_3". Then this problem is solved.
3. Transform the data, putting all the features(all the columns of the Spark.DataFrame but column "Status" and "Status_Value") into one vector. Get the predictors column as output. Choose the column "predictors" and "Status_Value" as the values to build a model for the training of the Random Forest Classifier. The data of the training model is shown as below:

```
+--------------------------------------------------------------------------------------+------------+
|predictors                                                                            |Status_value|
+--------------------------------------------------------------------------------------+------------+
|[4.5044,0.7443,6.34,1.9052,29.5315,0.8647,2.2044,6.048,14.4659,21.648,15.3429,1.2186]   |1.0         |
|[4.4284,0.9073,5.6433,1.6232,27.5032,1.4704,1.9929,5.9856,20.8356,0.0646,14.8813,7.3483]|1.0         |
|[4.5291,1.0199,6.113,1.0565,26.4271,1.9247,1.942,6.7162,5.3358,11.0779,25.0914,9.2408]  |1.0         |
|[5.1727,1.0007,7.8589,0.2765,25.1576,2.609,2.9234,6.7485,1.9017,1.8463,28.664,4.0157]   |1.0         |
|[5.2258,0.6125,7.9504,0.1547,24.0765,3.2113,4.4563,5.8411,0.5077,9.37,34.8122,13.4966]  |1.0         |
+--------------------------------------------------------------------------------------+------------+
only showing top 5 rows
```

4. Build and train the model.

For the complete source code please kindly check my .ipynb file.

Subtask 2.4

After the training, we will import MulticlassClassificationEvaluator from pyspark Machine Learning package, then we will evaluate the trained model in Subtask 2.3 and print the accuracy, precision, and AUROC.

The results are shown as fellowing:

```
The accuracy is 0.843537
The precision is 0.844635
The AUROC is 0.843100
```

Subtask 2.5

The procedures of this task is almost same as Task 2.4 but there are some points we have to notice:

1. When define the layers of MultilayerPerceptronClassifier, the nerve number of the first layer should be 12, I tried other numbers but the program would report an error, I assume it is because I defined 12 features in the feature vector.
2. The nerve number of the last layer should be 2, as this classification will classify two classes (normal and abnormal).

The evaluation results are shown as fellowing:

```
The accuracy is 0.704082
The precision is 0.705190
The AUROC is 0.701636
```

Bonus - Optional

In this task, I referred the code provided by the instructor and the original code on Github. As the following is the steps:

1. Install the configure the experiment environment.
2. I noticed that in the original code, the data set of training and test are saved in two individual .csv files, so I write a script to process the original excel dataset.
3. Import the original code I meantioned at the outset. Amending hyperparameters to fit with my task.
4. Train the model.
5. Show the result. As the following is part of the result.

```
>>> Fit model
>>> Synchronous training complete.
+------------+--------------------+
|index_Status|          prediction|
+------------+--------------------+
|         1.0|[0.07423058152198...|
|         1.0|[0.21709659695625...|
|         0.0|[0.94231098890304...|
|         1.0|[1.17160408990457...|
|         1.0|[0.00207798345945...|
|         1.0|[0.10999401658773...|
|         0.0|[0.87173652648925...|
|         1.0|[0.44925847649574...|
|         0.0|[0.94360321760177...|
|         1.0|[0.65069752931594...|
|         0.0|[0.79311126470565...|
|         1.0|[0.01574497111141...|
|         1.0|[0.02992281317710...|
|         1.0|[0.85060328245162...|
|         1.0|[0.12688443064689...|
|         0.0|[0.80256026983261...|
|         0.0|[0.88509386777877...|
|         1.0|[0.05394018813967...|
```

6. Evaluate the result.

However, the program report an error when I try to print the accuracy, which means I couldn't evaluate the training result.
I checked the my data processing and I think it is correct. In the sparkDataFrame, the predictor is a list instead of 1 or 0, which is strange.
Sometimes the predictor is 1 or 0, sometimes it is a list but the error occurred under the both condition. I tried to run the original code with the original dataset and the error is still occurred. I think the architecture of the model caused the error, I did not figure it out although I have tried many ways.

Conclusion, Supplimentary, and Argumentation

In subtask 2.5, I personaly not satified with the result of experiment, although I read from the intetnet that the accuracy of multilayer Perceptron Classifier is lower than Random Forest, I still think I can improve the result somehow. I think I can optimize it with changing the architecture of training model when I have more perception and knowledge of machine learning.

In the Bonus task, I tried many ways to solve the error, however they are all failed. One problem is I could not clear figure out what the error is. At the same time, I realized Apache Spark is a huge computation engine which includes many languages (python, scala, julia, jave, and so on). To solve this error, reading the documentation or source code maybe a good way.