UNIVERSITY OF ABERDEEN

CS551G - Data Mining and Visualization

Assessment 2

Name: Avinash Bagul

Student Id: 51987820

Note: Extension was given for this course work. Submission deadline: 03 May, 2020.

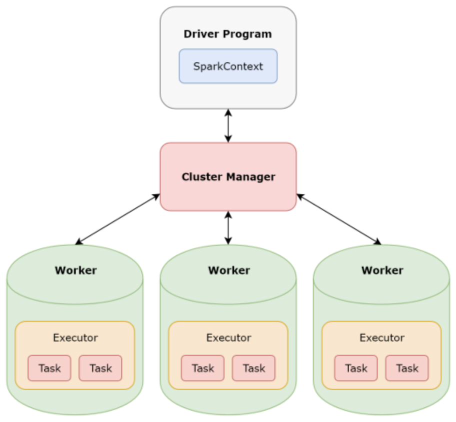
Table of Contents

1. **Task 1:** Description of Distributed Learning Big Data Ecosystem

HDFS is a distributed file system that handles huge amount of datasets running on off-the-shelf hardware. HDFS is one of the major component of Apache Hadoop and it is used for scaling a single hadoop cluster to hundreds and thousands of nodes. Other major components of Apache Hadoop are MapReduce and YARN. **Yet Another Resource Negotiator (YARN)** is a resource manager which is responsible for managing global assets among all nodes.

Data is often considered as “big data” if it can be expressed in terms of Volume, Variety, Velocity and Veracity. **Volume** in big data is the large amount of data in databases, different forms of data in big data is expressed as **Variety**. Frequently updated or changed data is termed as **Velocity** of data. **Veracity** means poor or unknown quality data in the big data. Fault tolerance and resilience serve as an effective way to label user’s reliability. **Fault Tolerance** in hadoop HDFS is the robustness of the system in a non-favorable situations and ability of the system to handle that situation. whereas, **Resilience** is the ability to recover from the fault and maintain the service. Data warehouse and **Data Lakes** are used for storing the big data. Data Lake is a vast pool of raw data which can be used for reporting, visualization, advanced analytics, machine learning and Data Mining purposes.

A well-known example of a general-purpose distributed processing engine is **Apache Spark**, it is capable of handling very large amount of data (petabytes of data). Apache Spark services run in **Java Virtual Machines**, Spark allows applications to be written in several languages i.e. Python, Java, Scala and R language. Spark provides effective support for query execution, machine learning, and data mining through rich high level libraries. Apache Spark employs a master/slave architecture as shown below:



Apache Hive is an Extract, transform, loading and data warehousing tool that facilitates reading, writing and managing large datasets residing in distributed storage using SQL. It is developed on the top of Hadoop Distributed File System (HDFS) and it provides interface to query data stored in various databases that integrate with Hadoop.

To support python with Apache Spark, **PySpark** API is used which is written in python. It basically allows user to integrate and work with Resilient Distributed Datasets in python programming language. Unlike PySpark, **Spark ML** is a package which provides high-level API that helps user to create and tune practical machine learning pipeline.

Apache **Spark SQL** is a spark module which simplifies working with structures and organized data with the help of abstractions and DataFrames in programming languages like Python, Java and Scala. Spark SQL fast in processing compared to Hive because it employs Spark core as processing engine to perform operations, and spark is at least 50 times faster than normal processing as it supports in-memory processing.

Docker is a tool for building, distributing and running **Docker Containers**. A Container in a docker is a standard unit of software that packages the code and its dependencies for reliability and faster execution of applications from one working environment to other. Example of Docker Container is Kubernetes, it is an open-source container management tool which provides platform for deployment, scaling and operations of containers across clusters.

2. **Task 2:** Develop distributed models in Apache spark to classify nuclear reactors

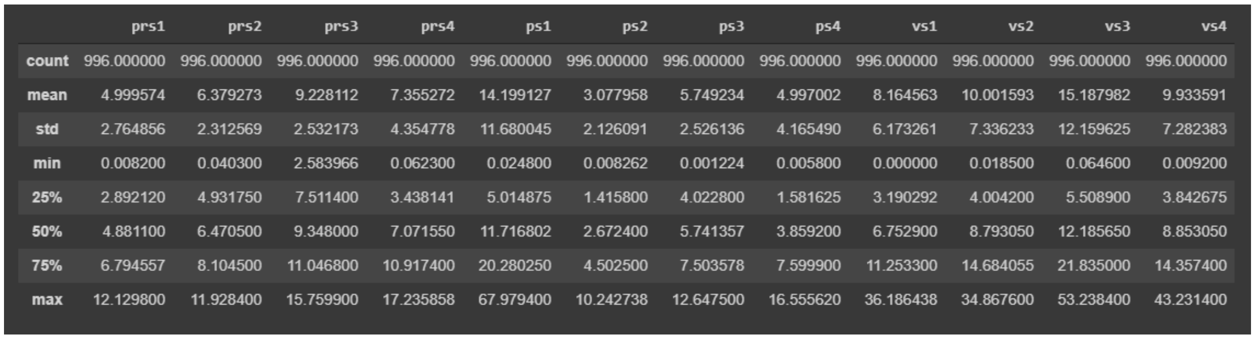
2.1. Data Description:

The given dataset is of the Pressurized Water Reactor containing data for three different types of sensors i.e. Power range sensor, Pressure sensor and Vibration sensor. There are four sensors of each category mentioned before. One data column “Status” describes the status of the water reactor as normal and abnormal. There is no null value in the given dataset. Data is well distributed with 498 cases of normal status and 498 cases of abnormal status, making it of shape (996, 13).

2.2. Step 1:

I have used CoLab to create Apache Spark environment. To read the dataset first I uploaded the dataset to CoLab and then read it using spark.read.csv() and also using pandas read_csv() function.

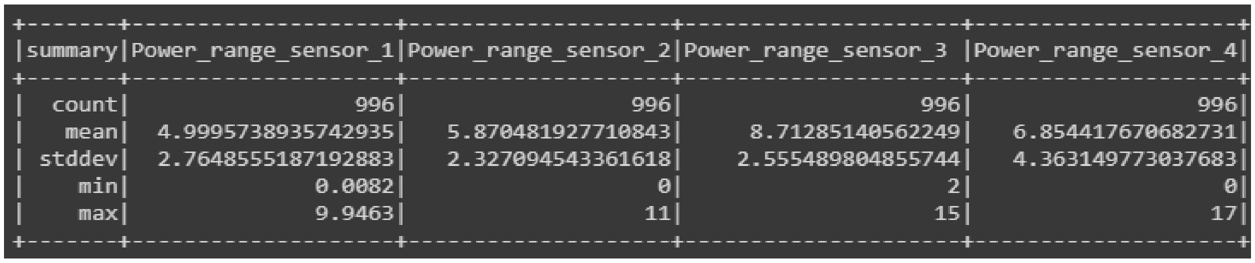
To provide insight of data statistics I have used two methods, one is using pandas library and other using PySpark. Pandas describe data in a nice tabular view giving almost every statistic detail: count, mean, standard deviation, min, max, 25% and 75% for all the features in the dataset as shown in the fig(2.1).



	prs1	prs2	prs3	prs4	ps1	ps2	ps3	ps4	vs1	vs2	vs3	vs4
count	996.000000	996.000000	996.000000	996.000000	996.000000	996.000000	996.000000	996.000000	996.000000	996.000000	996.000000	996.000000
mean	4.999574	6.379273	9.228112	7.355272	14.199127	3.077958	5.749234	4.997002	8.164563	10.001593	15.187982	9.933591
std	2.764856	2.312569	2.532173	4.354778	11.680045	2.126091	2.526136	4.165490	6.173261	7.336233	12.159625	7.282383
min	0.008200	0.040300	2.583966	0.062300	0.024800	0.008262	0.001224	0.005800	0.000000	0.018500	0.064600	0.009200
25%	2.892120	4.931750	7.511400	3.438141	5.014875	1.415800	4.022800	1.581625	3.190292	4.004200	5.508900	3.842675
50%	4.881100	6.470500	9.348000	7.071550	11.716802	2.672400	5.741357	3.859200	6.752900	8.793050	12.185650	8.853050
75%	6.794557	8.104500	11.046800	10.917400	20.280250	4.502500	7.503578	7.599900	11.253300	14.684055	21.835000	14.357400
max	12.129800	11.928400	15.759900	17.235858	67.979400	10.242738	12.647500	16.555620	36.186438	34.867600	53.238400	43.231400

Fig(2.1)

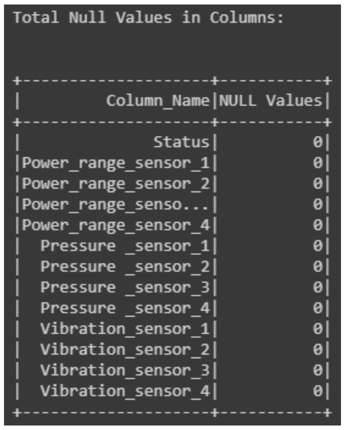
Using describe() of the pyspark, statistics shown in (fig(2.2)) can be obtained. Before describing the data "|Status" column was dropped out from data according to the requirement in assessment.



```
+-------+--------------------+--------------------+---------------------+--------------------+
|summary|Power_range_sensor_1|Power_range_sensor_2|Power_range_sensor_3 |Power_range_sensor_4|
+-------+--------------------+--------------------+---------------------+--------------------+
|  count|                 996|                 996|                  996|                 996|
|   mean|  4.9995738935742935|   5.870481927710843|     8.71285140562249|   6.854417670682731|
| stddev|  2.7648555187192883|   2.327094543361618|    2.555489804855744|   4.363149773037683|
|    min|              0.0082|                   0|                    2|                   0|
|    max|              9.9463|                  11|                   15|                  17|
+-------+--------------------+--------------------+---------------------+--------------------+
```

Fig(2.2)

After this I have shown if there is any null value in data again using both pandas and pyspark, Pandas simply has a isnull() function which when used with sum() function gives the null count for each data column. But in case of pyspark I had to loop each item in feature columns, check for isnull() and then count the number of null in the dataset. Output obtained for checking null values is shown in the fig(2.3).



```
Total Null Values in Columns:


+--------------------+-----------+
|         Column_Name|NULL Values|
+--------------------+-----------+
|              Status|          0|
|Power_range_sensor_1|          0|
|Power_range_sensor_2|          0|
|Power_range_senso...|          0|
|Power_range_sensor_4|          0|
|   Pressure _sensor_1|          0|
|   Pressure _sensor_2|          0|
|   Pressure _sensor_3|          0|
|   Pressure _sensor_4|          0|
|   Vibration_sensor_1|          0|
|   Vibration_sensor_2|          0|
|   Vibration_sensor_3|          0|
|   Vibration_sensor_4|          0|
+--------------------+-----------+
```

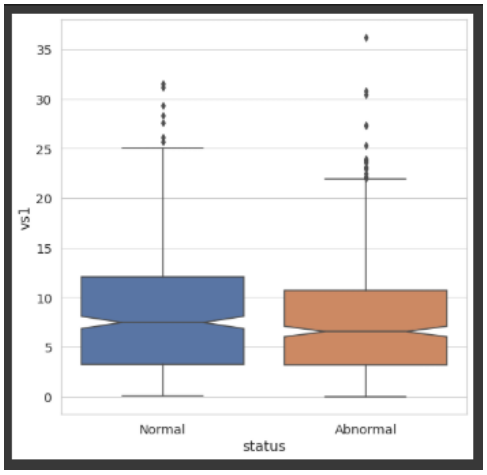
Fig(2.3)

So, there is no null value in any feature of the dataset.

2.3. Step 2:

This step of the assessment, demands plotting box plot and density plot for vibration sensor 1 and vibration sensor 2.

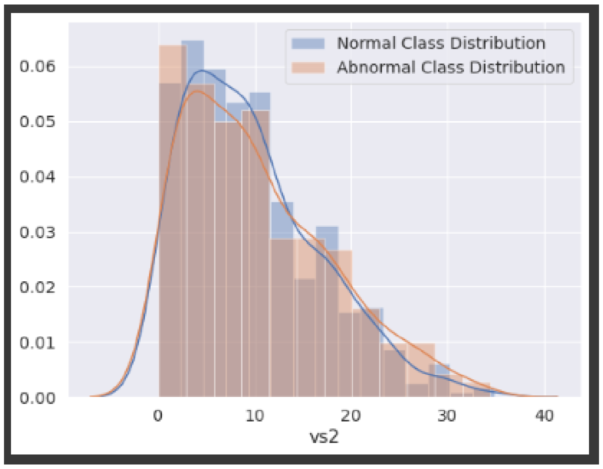
Box Plot: I have used seaborn library to plot the box plot with x-axis as Status (normal/abnormal) and y-axis as vibration sensor 1 as shown in fig(2.4).



Fig(2.4)

From the above plot we can understand the distribution of vibration sensor 1 for normal and abnormal classes. It shows min, 25th quartile, median, 75th quartile and max along with outliers. Sensor values for abnormal and slightly lower in comparison to the values for normal. All the values above 25 and 22 for normal and abnormal class respectively are considered as outliers.

Density Plot: For plotting density plot, I have used dist plot from seaborn library. Plot is used for plotting values of vibration sensor 2. Density plot is shown in the fig(2.5)



Fig(2.5)

In the above density plot legend shows that blue is normal class distribution and orange is abnormal class distribution. This plot shows us the distribution of vibration sensor two values for both normal and abnormal. Before plotting I divided dataset into two data based on the class (normal, abnormal). Vibration sensor 2 data from both newly created datasets is used to plot the distribution on distplot.

2.4. Step 3:

Splitting Dataset:

According to the assessment description I have split the data set into 75% train data and 25% test data using random split. This train data will be used to train the logistic regression model and test data will be used only for evaluation.

Developing Binary Logistic Regression Model:

Following libraries were used for developing binary logistic regression model

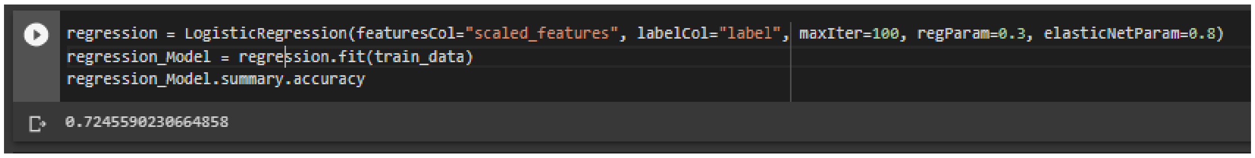
LogisticRegression was used from pyspark.ml.classification library.

VectorAssembler, StringIndexer and standardScaler was used from pyspark.ml.feature.

Logistic Rgression is a classification techniques which is an essential part of data mining problems and applications. Logistic Regression algorithm is a statistical method for predicting binary classes as it computes the probability of occurrence of an event.

Before training the model, data was scaled using StandardScaler(). Scaling the data is an essential step to normalize the data, which also helps to speeding up the calculations in our binary logistic regression algorithm used for this task.

LogisticRegression model was trained with maxIter=100, regParam=0.3 and elasticNetParam=0.8. Obtained accuracy was 0.7254.



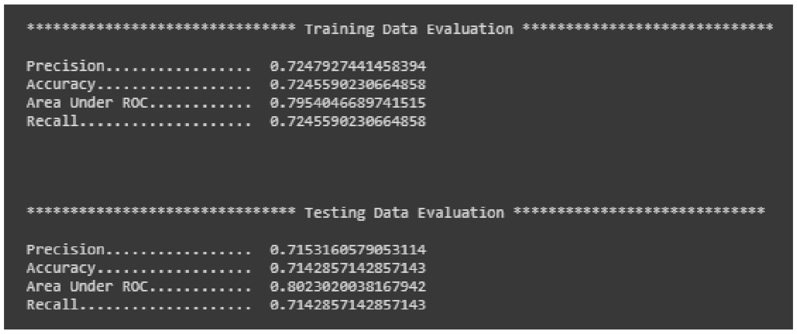
```
regression = LogisticRegression(featuresCol="scaled_features", labelCol="label", maxIter=100, regParam=0.3, elasticNetParam=0.8)
regression_Model = regression.fit(train_data)
regression_Model.summary.accuracy
```

```
0.7245590230664858
```

2.5. Step 4:

Performance Evaluation: Precision, Accuracy, Recall and Area Under the ROC metrics are used for evaluating the binary logistic regression model and gradient boosted tree classifier.

I have created a function which prints training metrics and test metrics for both regression model and tree classifier. This function is reusable. Below shown fig(2.6) is the output metrics for binary logistic regression model using this function.



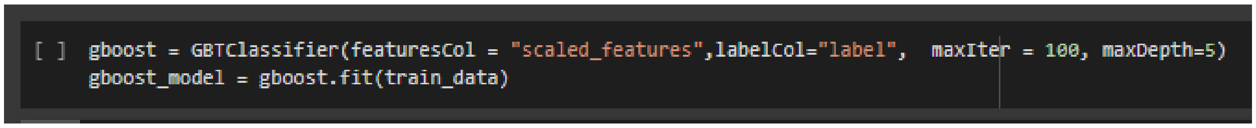
```
******************************** Training Data Evaluation ******************************

Precision................  0.7247927441458394
Accuracy.................  0.7245590230664858
Area Under ROC...........  0.7954046689741515
Recall...................  0.7245590230664858



******************************** Testing Data Evaluation ******************************

Precision................  0.7153160579053114
Accuracy.................  0.7142857142857143
Area Under ROC...........  0.8023020038167942
Recall...................  0.7142857142857143
```

Fig(2.6)

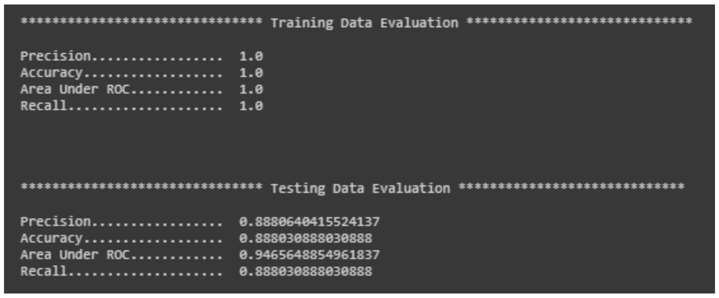
2.6. Step 5:

I have used GBTClassier from pyspark.ml.classification library. Gradient boosted tree classifier combines many learning models together to create strong model, it also uses decision trees when doing gradient boosting.

Gradient boosted tree classifier is trained using maxIter = 100 and max depth = 5. The results were shown with the help of show_result function.



```
[ ] gboost = GBTClassifier(featuresCol = "scaled_features",labelCol="label",  maxIter = 100, maxDepth=5)
    gboost_model = gboost.fit(train_data)
```



```
****************************** Training Data Evaluation *****************************

Precision................  1.0
Accuracy.................  1.0
Area Under ROC...........  1.0
Recall...................  1.0



****************************** Testing Data Evaluation *****************************

Precision................  0.8880640415524137
Accuracy.................  0.888030888030888
Area Under ROC...........  0.9465648854961837
Recall...................  0.888030888030888
```

2.7. Conclusion and Findings:

This assessment helped me to get basic understanding of distributed processing ecosystems for data mining and hands on PySpark library.

Talking about metric results associated with both the models, gradient boosted tree classifier performed better than logistic regression model.