FIT5045 Introduction to Data Science

Assignment 4 submission: Business & Data Case Study - Report

Topic:

Introducing Maintenance Prediction and Damage Occurrence of Trains and Trails of Public Transport Victoria using Big Data

Due 18 October 2019, 11.55 p.m.

Name (as per enrolment)	Farhad Ullah Rezwan
Student ID	30270111
Allocated Tutorial Class / Day / Time	Activity 30/Thursday/18:00 to 20:00
Name of Tutor	Dilini Rajapaksha Hewa Ranasinghage

Introducing Maintenance Prediction and Damage Occurrence of Trains and Trails of Public Transport Victoria using Big Data

1.0 Introduction:

Big data and data science are now a growing term and has great influence to solve many major problems of todays world. Usage of data is no longer now only business cantered, it has changed our daily life now though providing us with better entertainment, healthcare and safety. No one can deny the importance of data for the existence of better-quality life today. This plan illustrates how a maintenance and damage occurrence of train and trail system of Victorian train can be predicted using Big Data and Data Science applications.

2.0 Project Description:

This is a sample project that will use big data and data science technologies to improve the public transport system of Melbourne specially the trains. In short, this project will apply data science for the better-quality railway and trail engineering, predicting the maintenance pattern and helping to avoid unexpected delays and accidents.

Purpose:

- Railway trail and train maintenance using predictive analysis.
- Cost effective way of finding out the requirement of maintenance.
- Predict the train brake incidents.
- Analysing data from sensors to reduce emergency repairs

3.0 Data Science Job Roles:

Some of the sophisticated job roles has to be played during and after the implementation of this project. For example, for any sort of predictive situations regression model can be used and data scientist must take that responsibility. In another example the role of Machine Learning Engineer is also vital for this system to use algorithms like decision tree, clustering etc to come up with the appropriate type of machinery replacement that is required for the current situation. For this maintenance prediction and damage occurrence system for the train system of Public Transport Victoria, these are some vital data science job roles:

Number	Data Science Job	Roles Description
1.	Data Scientists	• Cleans data gathered form sensors from train engines and other parts. • Arrange streamed data to come up with real time analysis.
2.	Data Architect	• Protect data sources with appropriate maintenance
3.	Business Analyst	• Business intelligence understanding, managing the stakeholders. • Communicating with stakeholders. • Modelling the data for the usage of other data science job holders.
4.	Data Engineer	• Evolve, set-up, tryout and provide support system for the architecture system.
5.	Machine Learning Engineer	• Taking data analyst's output to implement into deployable real-world software system.
6.	Statistician	• Analysing the information form data and mark out the actions to be taken in priority basis. (i.e., maintenance requirements)

7.	System Analyst	• Define issues by consulting with the stakeholders(i.e. Train engineers) • Validates the test result of system implementation and document it. • Find out specifications considering feedback from the stakeholders.
8.	Operations Analyst	• Supporting the core operations of the system by linking with each of the data science and operations personal (i.e., data architect, system analyst, machine learning engineer). • Many cases apply data knowledge skills to visualise and cleanse the data.

4.0 Business Application

When fully become operational, these are the business requirements this proposed system will try to meet.

1. When fully developed will reduce the maintenance
2. Reduce employee workload for the maintenance engineers
3. Reduce the inspection timing and place which always been unpredictable.
4. Track machinery degradation and appropriate time to replace
5. Remove machinery before the failure reducing service fallout
6. Direct real time data (streaming data) and real time updates
7. Reduce the service disruptions

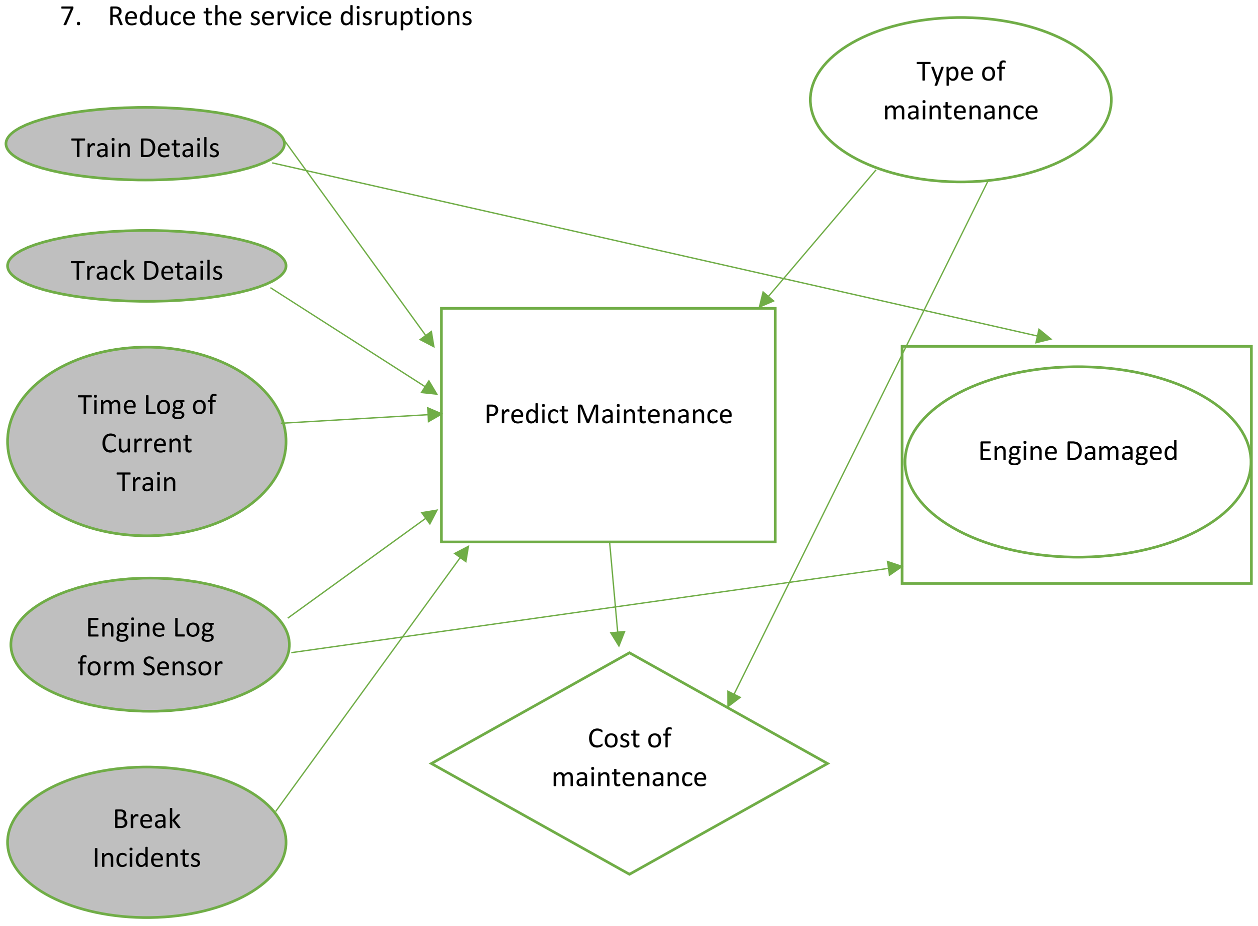


FIGURE 1: Influence diagram

5.0 Challenges

Some of the challenges of this project can be cost for applying and producing the sensors, network and data communication mechanisms with adequate signal strengths for the large amount of data, and high-performance distributed cluster computing system using high speed processor, controller and working(executioner) computers in Apache Spark Settings.

6.0 Values

The final outcome of this system for public transport Victoria has several opportunities for the different stakeholders.

Travellers: Travellers of PTV or the customers are the major stakeholders for this system. With the help of better and real-time maintenance system the schedule of timing of trains at the platform will be more accurate.

Train Drives: Drivers can maintain the speed and other parameters in accordance to the real time data for the train and the trail from the service centre.

Train and Trail Inspection Officers: Reduces the time and hassle for the train inspection, and trail inspection officers.

Control Room Officers: Provide them up with more current status of the trains. Controlling the traffic more logical manner.

Train Engineers: Providing the engineers with the latest and updated equipment information using sensors. Can use this system to get the root problems and find out the solutions regarding maintenance.

7.0 4v's and NIST Analysis:

Public Transport Victoria: Analysis

Data Sources: Sensors Installed in trail Carriage and Wheels, Train Inspection Equipment, Idea from (1), through Wireless Sensor Network

Data Volume: In total 220 six carriage train buggy, 965 KM of tracks (Metro Wikipedia).

Data Variety: Vibration Data, Heat, Fire, Cracks.

Data Velocity: Whenever Detects High Vibration, Wagon Data, irregulates of heat from machine data.

Data Veracity: Current Implementation is sensor based so prone to signal error or noice.

Software: MongoDB, Apache Spark, Apache ML.

Analytics: Visualization for Level of track damage, Machine Damage etc.

Processing: Clusters, Distributed Systems.

Capabilities: Models to support Large Amount of data Processing, Real Time data Processing

Security/Privacy: NA

8.0 Data Collection in Real World:

Acquiring data is an investment in the real world the sensor can be useful medium to get data. The typical real data for this project can be captured form multiple sensors and transferred with Wireless Sensor Network (WSN)

- Collecting data form running engine or when the train is operational, Heat/Temperature sensor, Fire Sensor, Humidity sensors can track vital data from engine.[1]
- Railway Axle cracks can be identified by vertical and horizontal vibration measurement [2], and Vertical vibration can be used for the purpose of wheel issues as well.
- When the trail is curved from the original position gyroscopic sensor can help to detect [1] Besides, Automated Ultrasonic Inspections can help to find out the cracks of the trail lines, Railway Switches or Turnouts (guides one track to another).

9.0 Data Exploration:

Data exploration is done to show users the most relevant data of the data set.

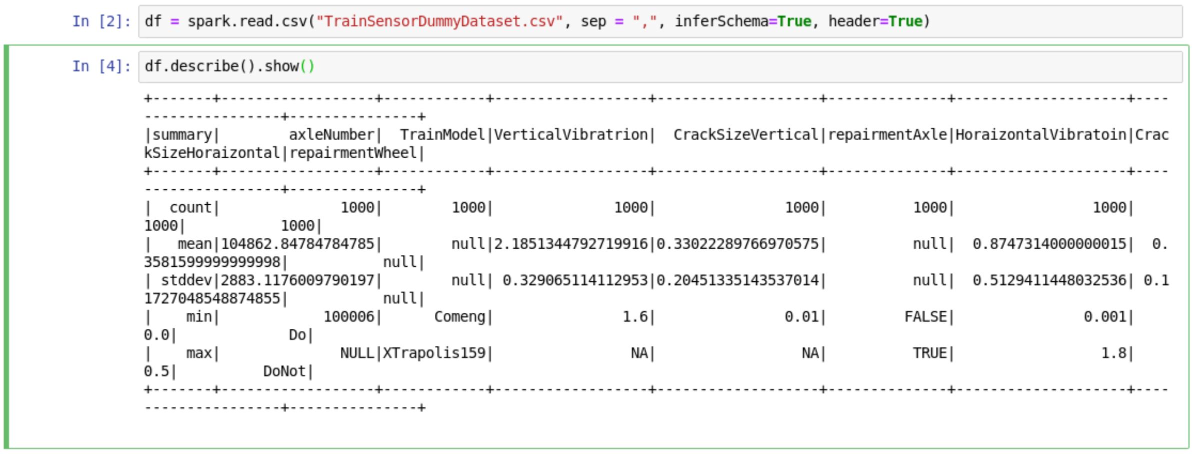
9.1 Descriptive Statistics

For the purpose of usability and understandability of data, summarizing and organizing the data is called Descriptive Statistics.[3] Descriptive statistics is categorized into 2-part, Measure of distribution and Measure of Central Tendency.

9.2 Statistical Testing

Statistical Tests are conducted to portray the statistical signification of an observation. On other word it is basically the hypothesis which is made for the importance of an observed sample.

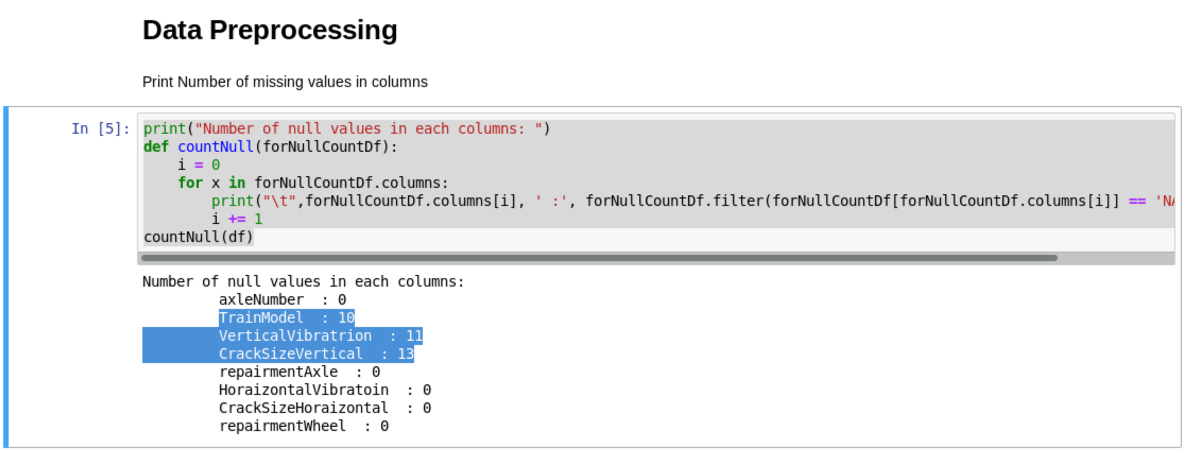
9.3 Simple Descriptive Analytics:



```
In [2]: df = spark.read.csv("TrainSensorDummyDataset.csv", sep = ",", inferSchema=True, header=True)

In [4]: df.describe().show()

+-------+------------------+------------+------------------+-------------------+--------------+--------------------+----
----------------+---------------+
|summary|        axleNumber|  TrainModel|VerticalVibratrion|  CrackSizeVertical|repairmentAxle|HoraizontalVibratoin|Crac
kSizeHoraizontal|repairmentWheel|
+-------+------------------+------------+------------------+-------------------+--------------+--------------------+----
----------------+---------------+
|  count|              1000|        1000|              1000|               1000|          1000|                1000|
1000|           1000|
|   mean|104862.84784784785|        null|2.1851344792719916|0.33022289766970575|          null|  0.8747314000000015|  0.
3581599999999998|           null|
| stddev|2883.1176009790197|        null| 0.329065114112953|0.20451335143537014|          null|  0.5129411448032536| 0.1
1727048548874855|           null|
|    min|            100006|      Comeng|               1.6|               0.01|         FALSE|               0.001|
0.0|             Do|
|    max|              NULL|XTrapolis159|                NA|                 NA|          TRUE|                 1.8|
0.5|          DoNot|
+-------+------------------+------------+------------------+-------------------+--------------+--------------------+----
----------------+---------------+
```

10.0 Data Pre-processing:

Data Pre-processing helps to transform raw data into useful data. By using data pre-processing for this project will help to analyse and figure out the prediction of repairment of wheels for the trains. For the data form the sensor of train's axle, the data computed in Apache Spark platform. The first step of data pre-processing is to deal with the null values. And finally, the dataset then fit into the model of victor assembler for the purpose of fitting the data frame into machine learning algorithms.



Data Preprocessing

Print Number of missing values in columns

```
In [5]: print("Number of null values in each columns: ")
        def countNull(forNullCountDf):
            i = 0
            for x in forNullCountDf.columns:
                print("\t",forNullCountDf.columns[i], ' :', forNullCountDf.filter(forNullCountDf[forNullCountDf.columns[i]] == 'N
                i += 1
        countNull(df)
```

```
Number of null values in each columns:
         axleNumber  : 0
         TrainModel  : 10
         VerticalVibratrion  : 11
         CrackSizeVertical  : 13
         repairmentAxle  : 0
         HoraizontalVibratoin  : 0
         CrackSizeHoraizontal  : 0
         repairmentWheel  : 0
```

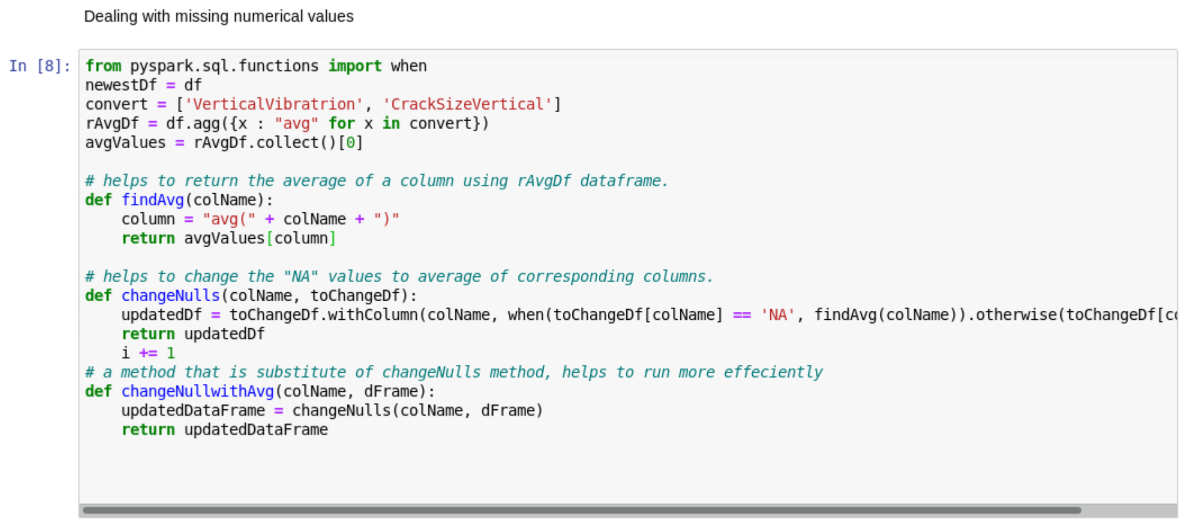
FIGURE 5: Showing number of null values in the dataset.

1. Dealing with Missing Values

Here for the better preparing the data frame for the processing the missing numerical data is replaced with the column average and the missing categorical data is replaced with the maximum occurrence value.

a. Numerical Columns:

The VerticalVibration and CrackSizeVertical are the numerical values, from the FIGURE 5 we can see that there are 11 and 13 missing values in those columns respectively. To fix the missing value first the column average is calculated, and then the average values are inserted removing the null values FIGURE 6.

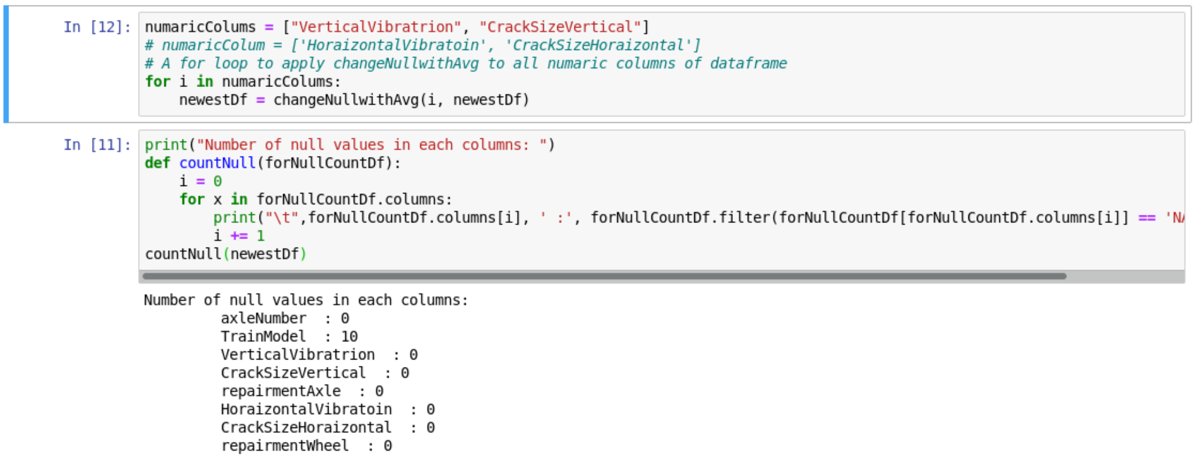


Dealing with missing numerical values

```
In [8]: from pyspark.sql.functions import when
        newestDf = df
        convert = ['VerticalVibratrion', 'CrackSizeVertical']
        rAvgDf = df.agg({x : "avg" for x in convert})
        avgValues = rAvgDf.collect()[0]

        # helps to return the average of a column using rAvgDf dataframe.
        def findAvg(colName):
            column = "avg(" + colName + ")"
            return avgValues[column]

        # helps to change the "NA" values to average of corresponding columns.
        def changeNulls(colName, toChangeDf):
            updatedDf = toChangeDf.withColumn(colName, when(toChangeDf[colName] == 'NA', findAvg(colName)).otherwise(toChangeDf[c
            return updatedDf
            i += 1
        # a method that is substitute of changeNulls method, helps to run more effeciently
        def changeNullwithAvg(colName, dFrame):
            updatedDataFrame = changeNulls(colName, dFrame)
            return updatedDataFrame
```

FIGURE 5: Programming for changing null values with average.

After applying the method ChangeNullwithAvg(), where the list of numerical columns and the data frame is given, we can see that the current null value status for VerticalVibration and CrackSizeVertical is zero FIGURE 6.

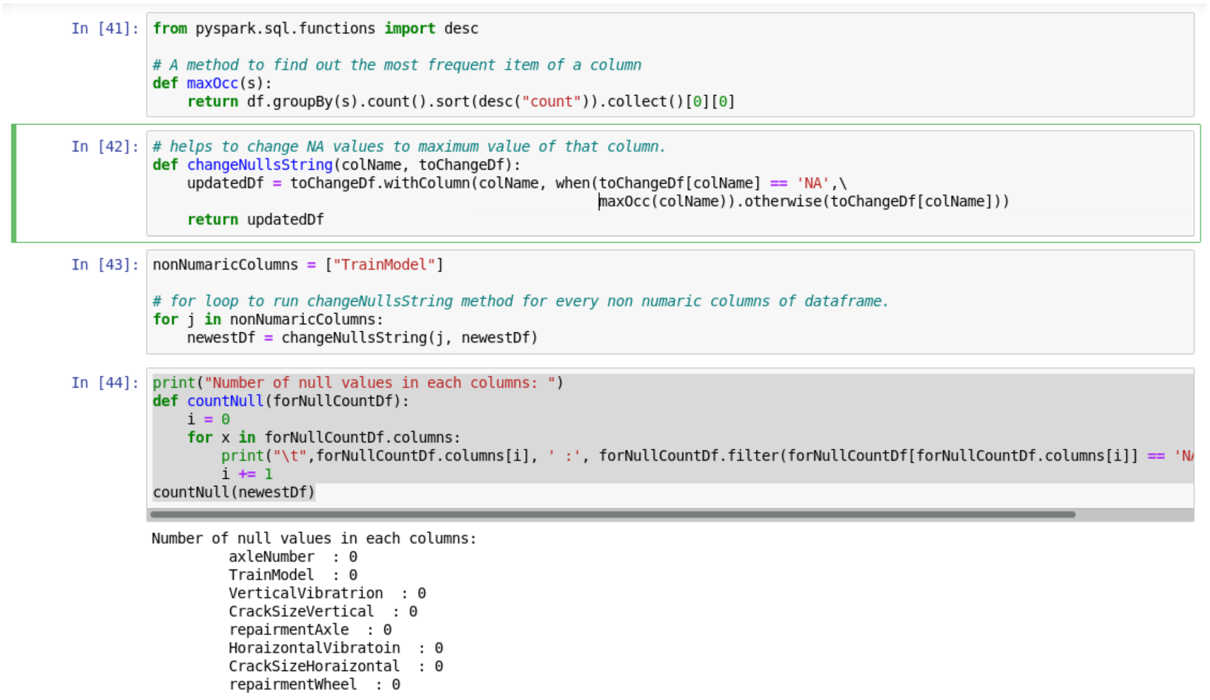


```
In [12]: numaricColums = ["VerticalVibratrion", "CrackSizeVertical"]
         # numaricColum = ['HoraizontalVibratoin', 'CrackSizeHoraizontal']
         # A for loop to apply changeNullwithAvg to all numaric columns of dataframe
         for i in numaricColums:
             newestDf = changeNullwithAvg(i, newestDf)

In [11]: print("Number of null values in each columns: ")
         def countNull(forNullCountDf):
             i = 0
             for x in forNullCountDf.columns:
                 print("\t",forNullCountDf.columns[i], ' :', forNullCountDf.filter(forNullCountDf[forNullCountDf.columns[i]] == 'N
                 i += 1
         countNull(newestDf)

         Number of null values in each columns:
                  axleNumber  : 0
                  TrainModel  : 10
                  VerticalVibratrion  : 0
                  CrackSizeVertical  : 0
                  repairmentAxle  : 0
                  HoraizontalVibratoin  : 0
                  CrackSizeHoraizontal  : 0
                  repairmentWheel  : 0
```

FIGURE 6: Null values for numerical columns are changed.

b. Categorical Column:

The TrainModel is the categorical column, from the FIGURE 5 we can see that there are 10 missing data in this column. To fix the missing value first the column maximum occurred value is figured out, and then the maximum occurred value is inserted removing the null values of that column.



```
In [41]: from pyspark.sql.functions import desc

         # A method to find out the most frequent item of a column
         def maxOcc(s):
             return df.groupBy(s).count().sort(desc("count")).collect()[0][0]

In [42]: # helps to change NA values to maximum value of that column.
         def changeNullsString(colName, toChangeDf):
             updatedDf = toChangeDf.withColumn(colName, when(toChangeDf[colName] == 'NA',\
                                                         maxOcc(colName)).otherwise(toChangeDf[colName]))
             return updatedDf

In [43]: nonNumaricColumns = ["TrainModel"]

         # for loop to run changeNullsString method for every non numaric columns of dataframe.
         for j in nonNumaricColumns:
             newestDf = changeNullsString(j, newestDf)

In [44]: print("Number of null values in each columns: ")
         def countNull(forNullCountDf):
             i = 0
             for x in forNullCountDf.columns:
                 print("\t",forNullCountDf.columns[i], ' :', forNullCountDf.filter(forNullCountDf[forNullCountDf.columns[i]] == 'N
                 i += 1
         countNull(newestDf)

         Number of null values in each columns:
                  axleNumber  : 0
                  TrainModel  : 0
                  VerticalVibratrion  : 0
                  CrackSizeVertical  : 0
                  repairmentAxle  : 0
                  HoraizontalVibratoin  : 0
                  CrackSizeHoraizontal  : 0
                  repairmentWheel  : 0
```

FIGURE 7: Null values for categorical column are changed

After applying the method ChangeNullwithAvg(), where the list of numerical columns and the data frame is given, we can see that the current null value status for VerticalVibration and CrackSizeVertical is zero FIGURE 6.

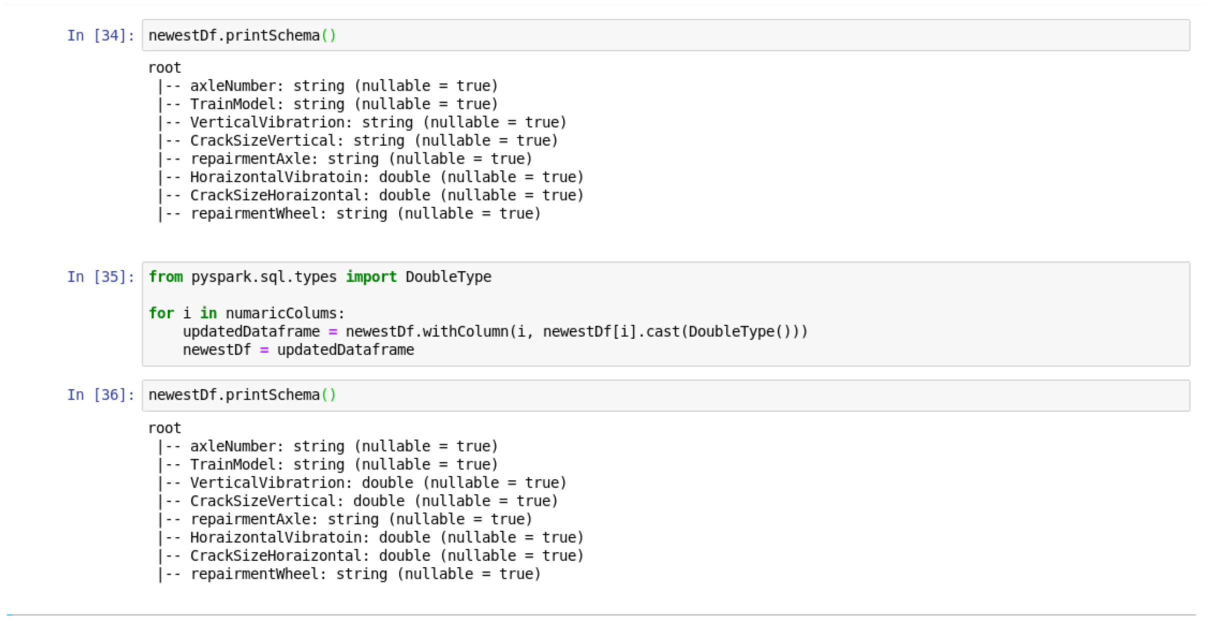
Finally we can see that after applying changeNullString() method the null values of TrainModel columns are replaced with the highest occurred value. As a result the current null value count for the all columns are zero [FIGURE 7].

2. Changing Data Type before processing

In the data pre-processing stage, the data-frame column type is changed to the correct type for the purpose of effective calculation.

a. Numerical Columns:

Here we can see that, at first the printSchema() method shows that all the column's data type is string. And after transforming the data the numerical values are type casted to double [FIGURE 8].

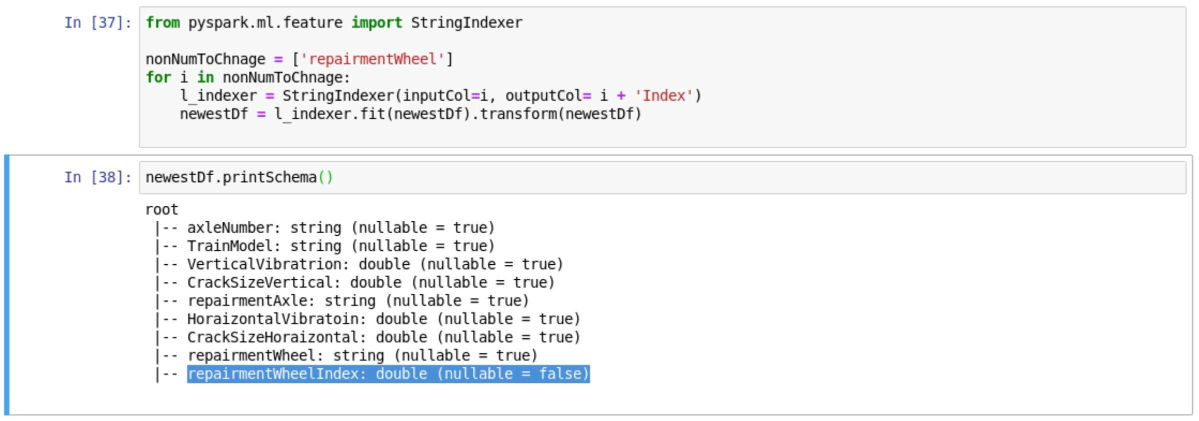


```
In [34]: newestDf.printSchema()

root
 |-- axleNumber: string (nullable = true)
 |-- TrainModel: string (nullable = true)
 |-- VerticalVibratrion: string (nullable = true)
 |-- CrackSizeVertical: string (nullable = true)
 |-- repairmentAxle: string (nullable = true)
 |-- HoraizontalVibratoin: double (nullable = true)
 |-- CrackSizeHoraizontal: double (nullable = true)
 |-- repairmentWheel: string (nullable = true)

In [35]: from pyspark.sql.types import DoubleType

         for i in numaricColums:
             updatedDataframe = newestDf.withColumn(i, newestDf[i].cast(DoubleType()))
             newestDf = updatedDataframe

In [36]: newestDf.printSchema()

root
 |-- axleNumber: string (nullable = true)
 |-- TrainModel: string (nullable = true)
 |-- VerticalVibratrion: double (nullable = true)
 |-- CrackSizeVertical: double (nullable = true)
 |-- repairmentAxle: string (nullable = true)
 |-- HoraizontalVibratoin: double (nullable = true)
 |-- CrackSizeHoraizontal: double (nullable = true)
 |-- repairmentWheel: string (nullable = true)
```

FIGURE 8: Type casting numerical data into double.

b. Categorical Columns:

Here, string indexer is used to generate a new column named repairementWheelIndex for the purpose of its usability in ML algorithms [FIGURE 9].

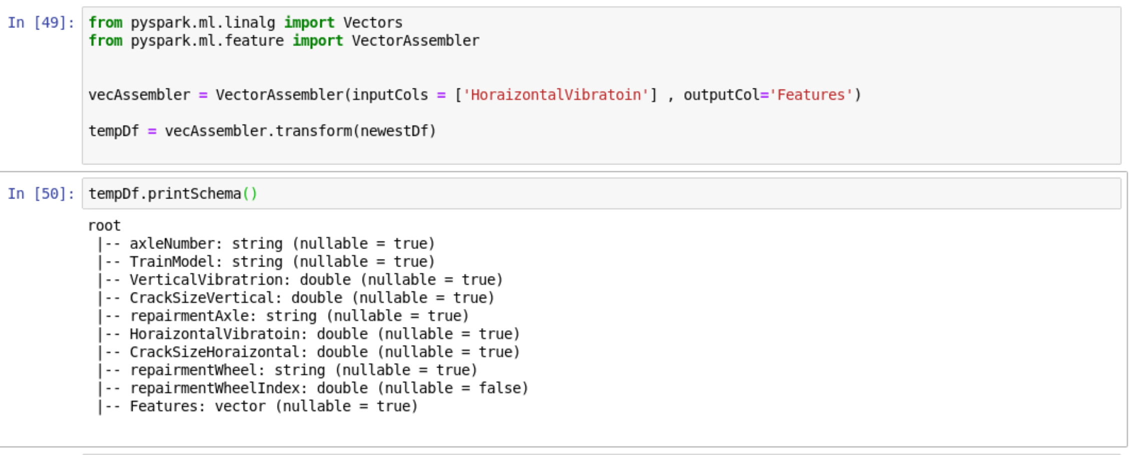


```
In [37]: from pyspark.ml.feature import StringIndexer

         nonNumToChnage = ['repairmentWheel']
         for i in nonNumToChnage:
             l_indexer = StringIndexer(inputCol=i, outputCol= i + 'Index')
             newestDf = l_indexer.fit(newestDf).transform(newestDf)

In [38]: newestDf.printSchema()

root
 |-- axleNumber: string (nullable = true)
 |-- TrainModel: string (nullable = true)
 |-- VerticalVibratrion: double (nullable = true)
 |-- CrackSizeVertical: double (nullable = true)
 |-- repairmentAxle: string (nullable = true)
 |-- HoraizontalVibratoin: double (nullable = true)
 |-- CrackSizeHoraizontal: double (nullable = true)
 |-- repairmentWheel: string (nullable = true)
 |-- repairmentWheelIndex: double (nullable = false)
```

FIGURE 9: Using String indexer to generate usable column as type Double.

3. Using Vector Assembler to Prepare for Machine Learning Algorithm

Here the feature (HoraizontalVibratoin) which will be used as input column in the ML algorithms is transformed to Features column as vector type. This data pre-processing step is conducted using the appropriate features 2 or more for the new column. Here for the general explanation purpose one column is used [FIGURE].



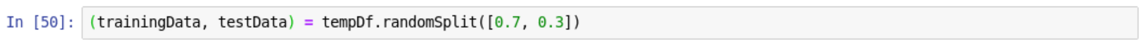
```
In [49]: from pyspark.ml.linalg import Vectors
         from pyspark.ml.feature import VectorAssembler


         vecAssembler = VectorAssembler(inputCols = ['HoraizontalVibratoin'] , outputCol='Features')

         tempDf = vecAssembler.transform(newestDf)
```

```
In [50]: tempDf.printSchema()

         root
          |-- axleNumber: string (nullable = true)
          |-- TrainModel: string (nullable = true)
          |-- VerticalVibratrion: double (nullable = true)
          |-- CrackSizeVertical: double (nullable = true)
          |-- repairmentAxle: string (nullable = true)
          |-- HoraizontalVibratoin: double (nullable = true)
          |-- CrackSizeHoraizontal: double (nullable = true)
          |-- repairmentWheel: string (nullable = true)
          |-- repairmentWheelIndex: double (nullable = false)
          |-- Features: vector (nullable = true)
```

FIGURE 10: Using Vector Assembler to form vector feature.

11.0 Applying Machine Learning Algorithms

To Apply machine learning algorithms at first, we need to divide the dataset into two. Here the dataset is divided into trainingData and testData using randomSplit method and for trainingData 70 percent of data form the dataset is used, and for testdata 30 percent of dataset is used [FIGURE 11].

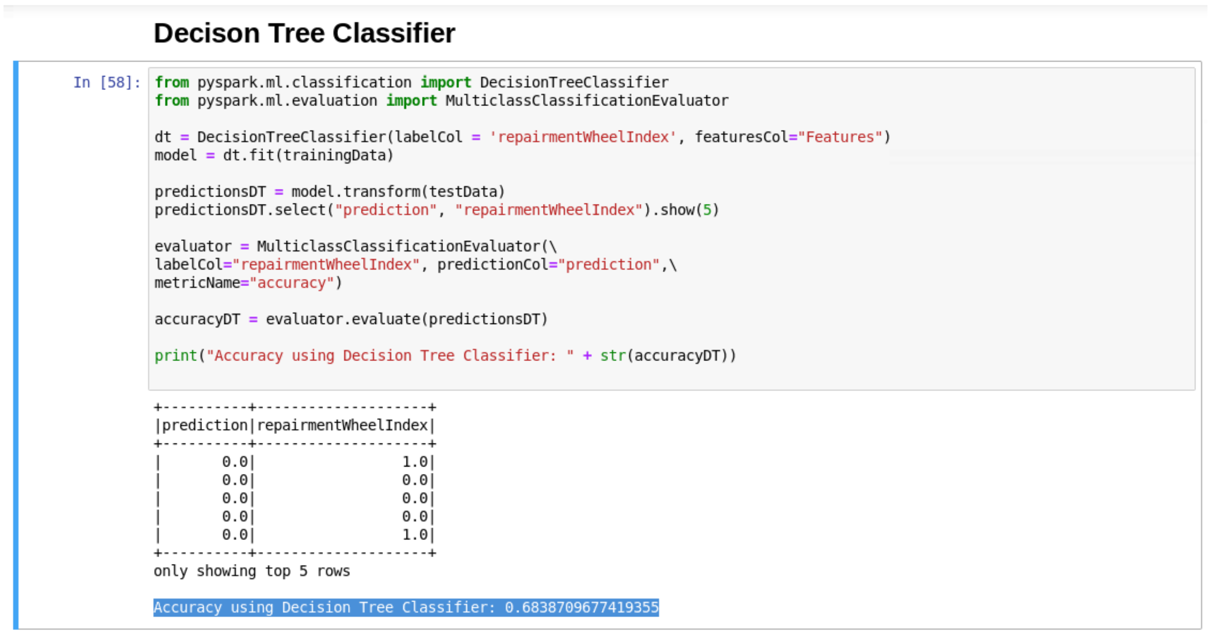


```
In [50]: (trainingData, testData) = tempDf.randomSplit([0.7, 0.3])
```

FIGURE 11: Splitting the dataset into training and testing data.

11.1 Decision Tree Algorithm

Decision tree algorithm is one of the popular algorithms that is used in classification and regression problems.[4] Here we found out that the accuracy of prediction of decision tree algorithm is 68.38 percent.



Decison Tree Classifier

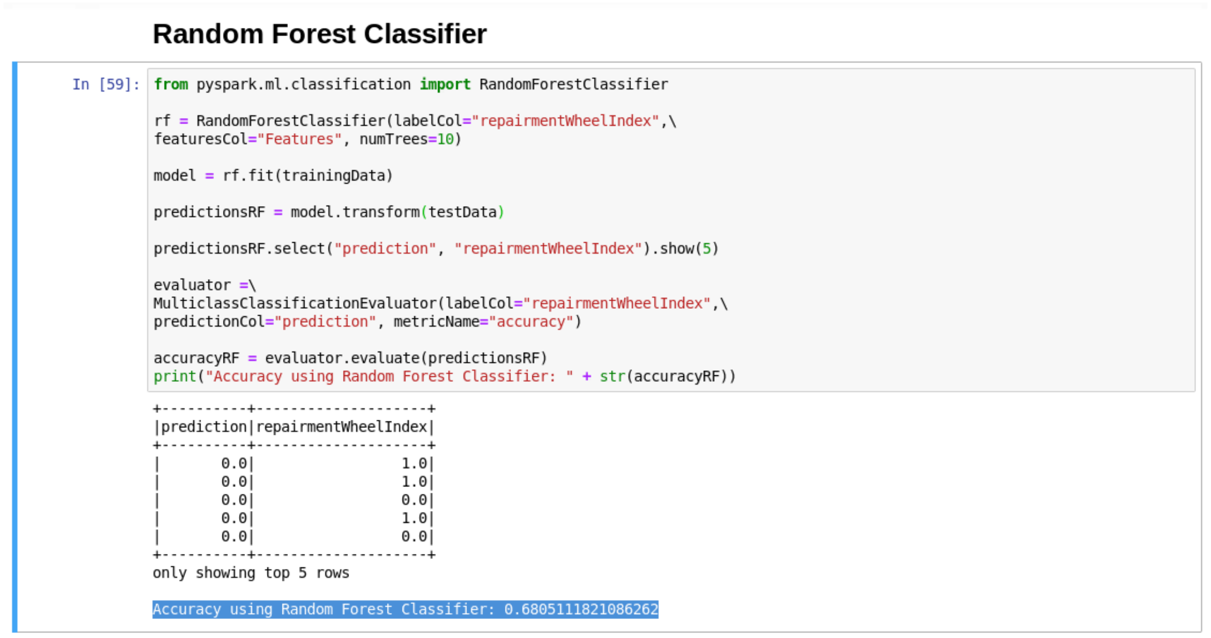
```
In [58]: from pyspark.ml.classification import DecisionTreeClassifier
         from pyspark.ml.evaluation import MulticlassClassificationEvaluator

         dt = DecisionTreeClassifier(labelCol = 'repairmentWheelIndex', featuresCol="Features")
         model = dt.fit(trainingData)

         predictionsDT = model.transform(testData)
         predictionsDT.select("prediction", "repairmentWheelIndex").show(5)

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

         accuracyDT = evaluator.evaluate(predictionsDT)

         print("Accuracy using Decision Tree Classifier: " + str(accuracyDT))
```

```
+----------+--------------------+
|prediction|repairmentWheelIndex|
+----------+--------------------+
|       0.0|                 1.0|
|       0.0|                 0.0|
|       0.0|                 0.0|
|       0.0|                 0.0|
|       0.0|                 1.0|
+----------+--------------------+
only showing top 5 rows

Accuracy using Decision Tree Classifier: 0.6838709677419355
```

FIGURE 12: Decision Tree Algorithm.

11.2 Random Forest Algorithm

Another Classification Algorithm in Machine learning is Random Forest algorithm. Decision trees are the building block of Random Forest Classifier.[5] Here we can see that the accuracy of Random Forest algorithm for the dataset to predict the requirement of wheel replacement for trains is 68.05 percent.



Random Forest Classifier

```
In [59]: from pyspark.ml.classification import RandomForestClassifier

         rf = RandomForestClassifier(labelCol="repairmentWheelIndex",\
         featuresCol="Features", numTrees=10)

         model = rf.fit(trainingData)

         predictionsRF = model.transform(testData)

         predictionsRF.select("prediction", "repairmentWheelIndex").show(5)

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

         accuracyRF = evaluator.evaluate(predictionsRF)
         print("Accuracy using Random Forest Classifier: " + str(accuracyRF))
```

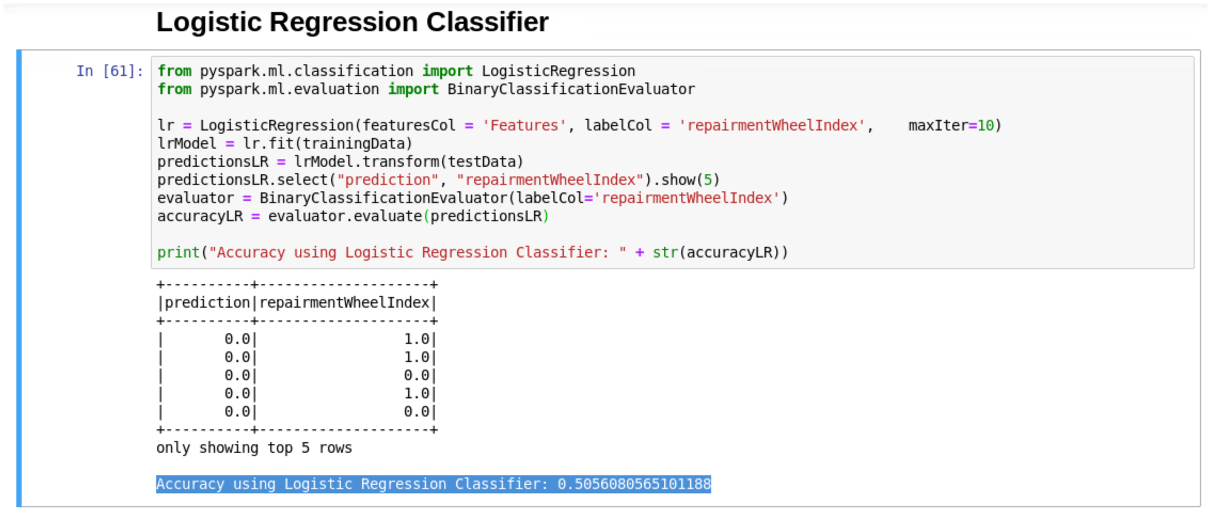
```
+----------+--------------------+
|prediction|repairmentWheelIndex|
+----------+--------------------+
|       0.0|                 1.0|
|       0.0|                 1.0|
|       0.0|                 0.0|
|       0.0|                 1.0|
|       0.0|                 0.0|
+----------+--------------------+
only showing top 5 rows

Accuracy using Random Forest Classifier: 0.6805111821086262
```

FIGURE 13: Random Forest Algorithm.

11.3 Logistic Regression Algorithm

Logistic Regression is another popular machine learning algorithm. Logistic Regression and Linear Regression is quite same; however, the main difference is that the Logistic Regression is used for

Classification activities.[6] The prediction using Logistic Regression is poor, generating only 50.56% of accuracy [FIGURE 14].



Logistic Regression Classifier

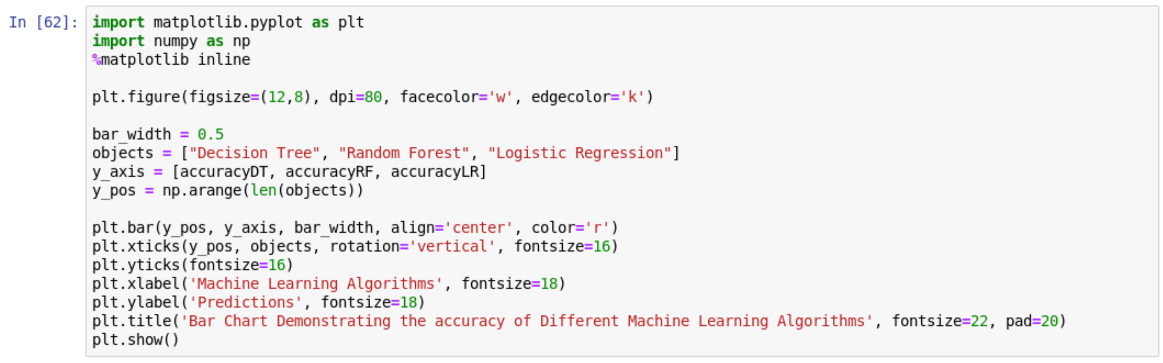
```
In [61]: from pyspark.ml.classification import LogisticRegression
         from pyspark.ml.evaluation import BinaryClassificationEvaluator

         lr = LogisticRegression(featuresCol = 'Features', labelCol = 'repairmentWheelIndex',    maxIter=10)
         lrModel = lr.fit(trainingData)
         predictionsLR = lrModel.transform(testData)
         predictionsLR.select("prediction", "repairmentWheelIndex").show(5)
         evaluator = BinaryClassificationEvaluator(labelCol='repairmentWheelIndex')
         accuracyLR = evaluator.evaluate(predictionsLR)

         print("Accuracy using Logistic Regression Classifier: " + str(accuracyLR))
```

```
+----------+--------------------+
|prediction|repairmentWheelIndex|
+----------+--------------------+
|       0.0|                 1.0|
|       0.0|                 1.0|
|       0.0|                 0.0|
|       0.0|                 1.0|
|       0.0|                 0.0|
+----------+--------------------+
only showing top 5 rows

Accuracy using Logistic Regression Classifier: 0.5056080565101188
```

FIGURE 14: Random Forest Algorithm.

11.4 Comparing Different Algorithms Based on Prediction

Analysing all the algorithm on the basis of prediction we can see that the Logistic Regression is the worst among three. Random Forest and Decision Tree has quite similar accuracy. The best fit for this project dataset and for predicting the train wheel reparation will be using either Decision Tree or Random Forest algorithm.



```
In [62]: import matplotlib.pyplot as plt
         import numpy as np
         %matplotlib inline

         plt.figure(figsize=(12,8), dpi=80, facecolor='w', edgecolor='k')

         bar_width = 0.5
         objects = ["Decision Tree", "Random Forest", "Logistic Regression"]
         y_axis = [accuracyDT, accuracyRF, accuracyLR]
         y_pos = np.arange(len(objects))

         plt.bar(y_pos, y_axis, bar_width, align='center', color='r')
         plt.xticks(y_pos, objects, rotation='vertical', fontsize=16)
         plt.yticks(fontsize=16)
         plt.xlabel('Machine Learning Algorithms', fontsize=18)
         plt.ylabel('Predictions', fontsize=18)
         plt.title('Bar Chart Demonstrating the accuracy of Different Machine Learning Algorithms', fontsize=22, pad=20)
         plt.show()
```

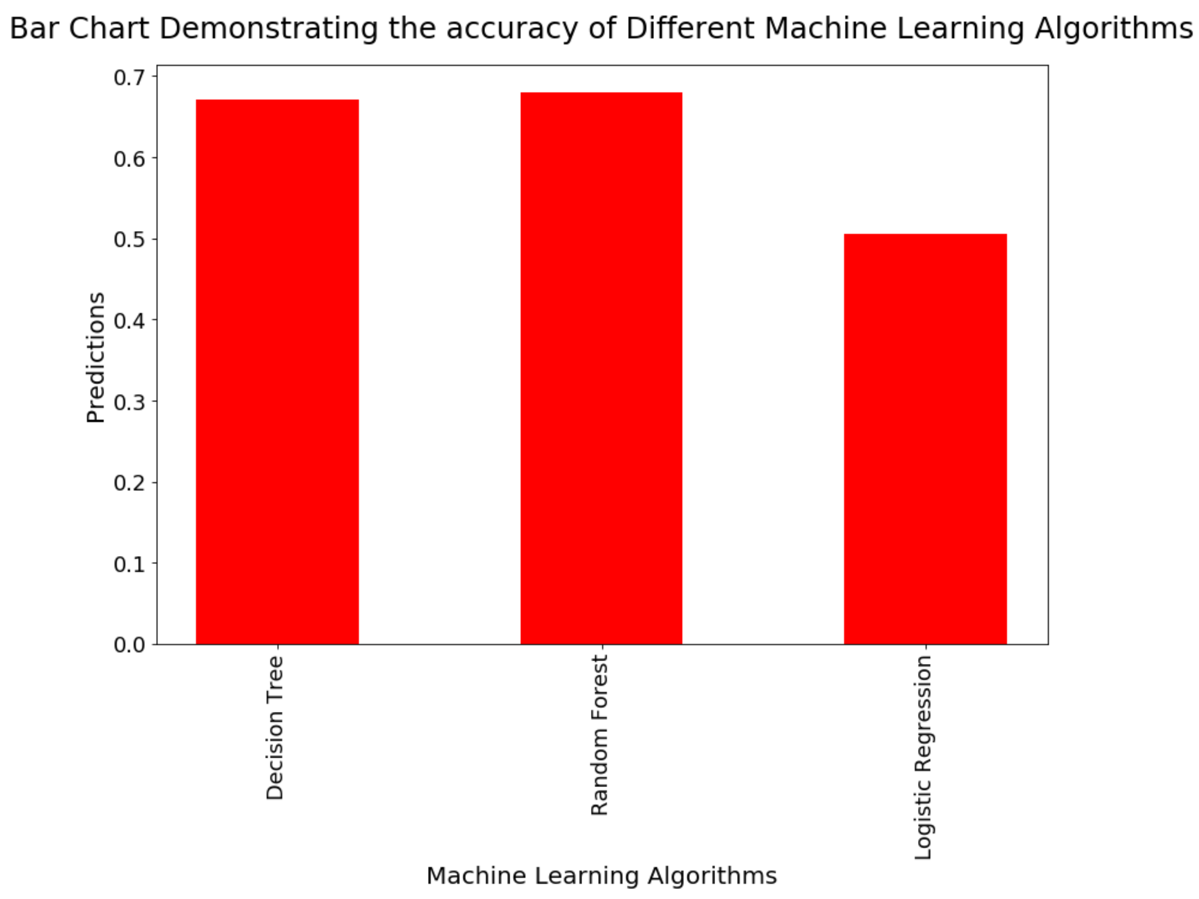


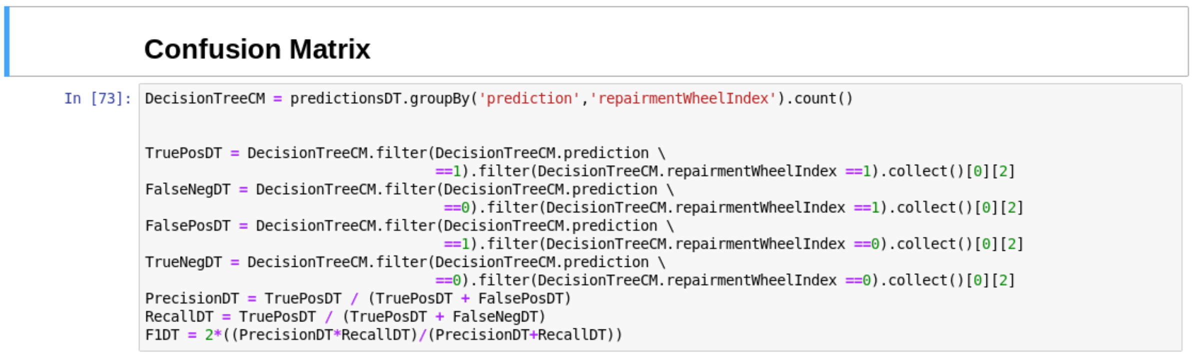
FIGURE 15: Random Forest Algorithm.

12.0 Confusion Matrix and Precision, Recall and F1 Score.

Confusion Matrix is the measurement of performance of Machine Learning Algorithm.[7] To understand and Evaluate Confusion Matrix with math we use Precision Recall and F1 Score.

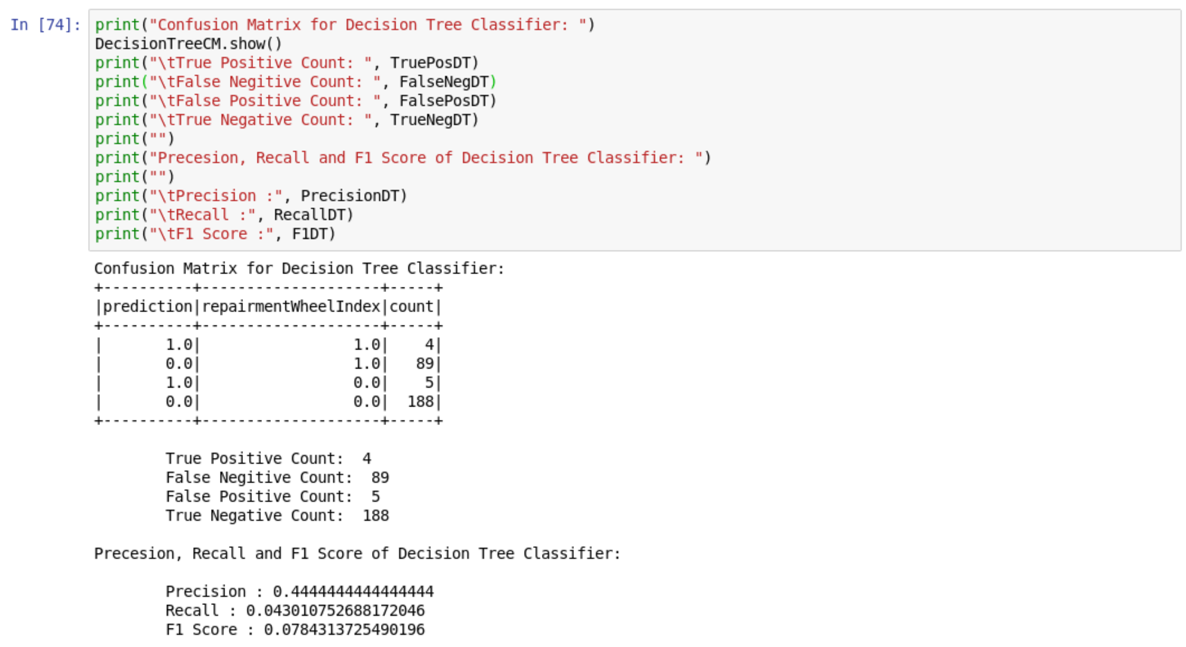
12.1 Confusion Matrix

For this project we choose Decision Tree, Random Forest and Logistic Regression algorithm. In basis of predicting the repairment of wheel of train and actual repairment data it is shown that Decision tree and Random Forest are the good fit. However, after confusion matrix calculations, the result shows us that the Decision Tree algorithm is best fit for the data-frame.



Confusion Matrix

In [73]:
```
DecisionTreeCM = predictionsDT.groupBy('prediction','repairmentWheelIndex').count()


TruePosDT = DecisionTreeCM.filter(DecisionTreeCM.prediction \
                                 ==1).filter(DecisionTreeCM.repairmentWheelIndex ==1).collect()[0][2]
FalseNegDT = DecisionTreeCM.filter(DecisionTreeCM.prediction \
                                  ==0).filter(DecisionTreeCM.repairmentWheelIndex ==1).collect()[0][2]
FalsePosDT = DecisionTreeCM.filter(DecisionTreeCM.prediction \
                                  ==1).filter(DecisionTreeCM.repairmentWheelIndex ==0).collect()[0][2]
TrueNegDT = DecisionTreeCM.filter(DecisionTreeCM.prediction \
                                 ==0).filter(DecisionTreeCM.repairmentWheelIndex ==0).collect()[0][2]
PrecisionDT = TruePosDT / (TruePosDT + FalsePosDT)
RecallDT = TruePosDT / (TruePosDT + FalseNegDT)
F1DT = 2*((PrecisionDT*RecallDT)/(PrecisionDT+RecallDT))
```



```
In [74]: print("Confusion Matrix for Decision Tree Classifier: ")
         DecisionTreeCM.show()
         print("\tTrue Positive Count: ", TruePosDT)
         print("\tFalse Negitive Count: ", FalseNegDT)
         print("\tFalse Positive Count: ", FalsePosDT)
         print("\tTrue Negative Count: ", TrueNegDT)
         print("")
         print("Precesion, Recall and F1 Score of Decision Tree Classifier: ")
         print("")
         print("\tPrecision :", PrecisionDT)
         print("\tRecall :", RecallDT)
         print("\tF1 Score :", F1DT)
```

```
Confusion Matrix for Decision Tree Classifier:
+----------+-------------------+-----+
|prediction|repairmentWheelIndex|count|
+----------+-------------------+-----+
|       1.0|                1.0|    4|
|       0.0|                1.0|   89|
|       1.0|                0.0|    5|
|       0.0|                0.0|  188|
+----------+-------------------+-----+

        True Positive Count:  4
        False Negitive Count:  89
        False Positive Count:  5
        True Negative Count:  188

Precesion, Recall and F1 Score of Decision Tree Classifier:

        Precision : 0.4444444444444444
        Recall : 0.043010752688172046
        F1 Score : 0.0784313725490196
```

12.2 Selecting Best Model of Machine Learning Algorithm

Here we got the Recall value of 0.043 which is higher for Decision Tree Algorithm compared to Random Forest and Logistic Regression which has precision values of 0.039 and 0.027 respectively. The F1 Score of 0.078 and the Precision value of 0.44 shows that the best model for this project of predicting the wheel repair requirement for each train will be the Decision Tree algorithm.

13.0 Conclusion:

Using big data to solve an issue which has daily impact on our life is not rocket science any more. With the advancement of new technology, high performing processors, data storage mechanisms and networking system, data can be transferred very fast and process the real time data to the information of taking serious action plan straight away. This resulting system will help improve the train transport system in Victoria can become handy in the near future for train management as well.

References:

1. Alawad, Hamad, Kaewunruen and Sakdirat. Wireless Sensor Networks: Toward Smarter Railway Stations. 2018 Jul; 3 : 24.
2. Hassan M, Bruni S, Carboni. Crack detection in railway axle using horizontal and vertical vibration measurements. 7th IET Conference on Railway Condition Monitoring 2016 (RCM 2016). 2016; 1-6. DOI: 10.1049/cp.2016.1190
3. Investopedia. Finance encyclopedia [Internet]. Descriptive Statistics [updated 2019 Jun 27; cited 2006 Nov 16]; [about 4 screens]. Available from: https://www.investopedia.com/terms/d/descriptive_statistics.asp
4. Grimaldi E. Decision Tree: an algorithm that works like the human brain. Towards Data Science. 1997;1(1-4):3-23. Available from: https://doi.org/10.1016/S1088-467X(98)00007-9.
5. Understanding Random Forest [Internet]. How the Algorithm Works and Why it Is So Effective; 2012 Jun. Available from: https://towardsdatascience.com/understanding-random-forest-58381e0602d2

6. Gandhi R. Introduction to Machine Learning Algorithms: Logistic Regression [Internet]. 2019 Oct 14. Available from: https://hackernoon.com/introduction-to-machine-learning-algorithms-logistic-regression-cbdd82d81a36
7. Narkhede S. Understanding Confusion Matrix [Internet]. 2018 May 9. Available from: https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62
8. Famili, Shen WM, Weber R, Simoudis E. Data preprocessing and intelligent data analysis. Intelligent Data Analysis. 1997;1(1-4):3-23. Available from: https://doi.org/10.1016/S1088-467X(98)00007-9.
9. Towards Data Science [Internet]. Data preprocessing and intelligent data analysis; 2018 Sep 11. Available from: https://towardsdatascience.com/decision-tree-an-algorithm-that-works-like-the-human-brain-8bc0652f1fc6