Machine Learning Engineer Nano degree

Capstone Project

Predicting the sustainability of hand pumps

NAGIREDDY SEELAM

February 17th, 2019

Proposal:

Predicting the Sustainability of Hand Pumps

I. Definition

Project Overview:-

In Sub-Saharan Africa, an estimated 184 million people rely on hand pumps for their water supply. The goal of this study is to develop an algorithm that can predict hand pump sustainability in low-income countries based on a minimum of data collected on the field. Predicting the sustainability of a hand pump at a given point in time can help shorten the time for NGOs to provide support and organize targeted maintenance operations in remote areas. Using the “Taarifa” dataset, we trained, compared and optimized different machine learning algorithms to predict three categorical features of the dataset that were identified as possible indicators of hand pump sustainability: functionality of the hand pump, quantity of water delivered, and quality of water delivered.

History:

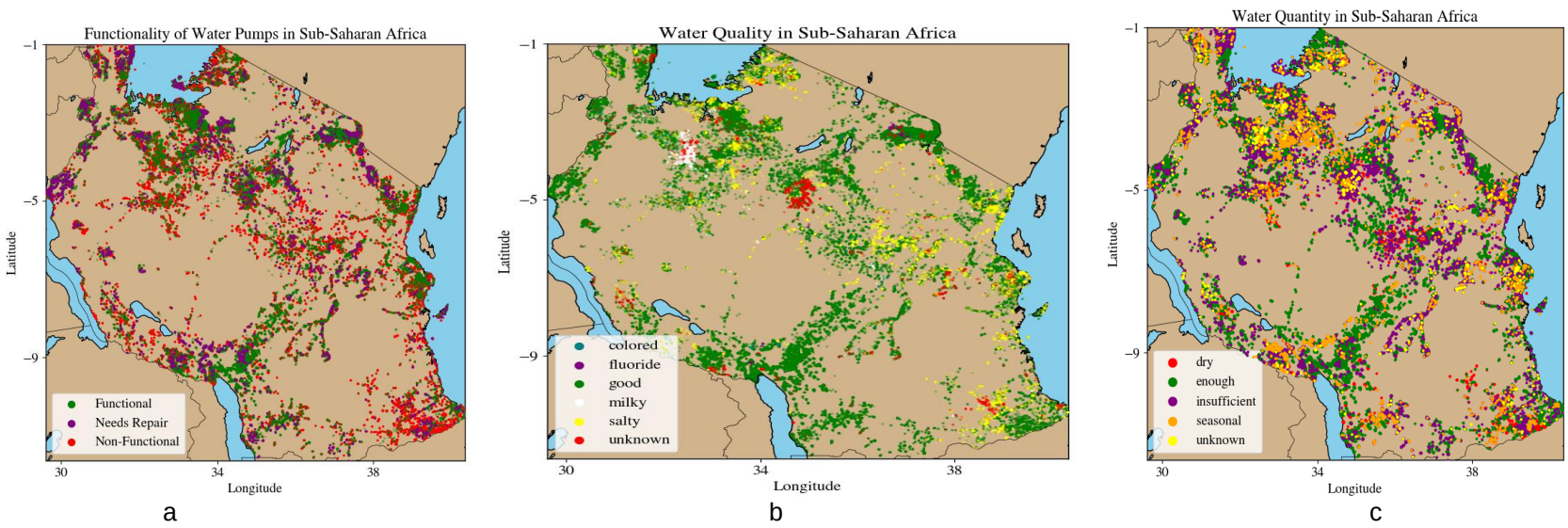
In 2015, an estimated 184 million people living in Sub-Saharan Africa relied on hand pumps for their water supply and more than 300 million people lacked access to an improved water source. Historically, development agencies have been supporting those populations by providing infrastructures, such as hand pumps, but very little attention had been directed to their sustainability and their maintenance. Functionality rate of hand pumps in selected Sub-Saharan countries was 36% in 2009, and is respectively 15% and 25% one year and two years after construction in 2016

Related work:

The Taarifa dataset that we used in this study, and variants of it, has been extensively explored in the field of access to Water Sanitation and Hygiene services in developing countries. Most of those studies however do not use machine learning methods to analyse the dataset. A recent study from 2017 used a Bayesian network to analyse correlations in the data. Some groups have used Machine Learning methods but did so for other purposes: a study from 2013 used STATA to perform Multivariate Logistic Regression but only looked at relationships between features and non-functionality of the hand pump.

Dataset:-

The dataset used in this study was collected by the Tanzania Ministry of Water, aggregated by Taarifa, and Figure shows the features that will be classified. This dataset contains data for 59,400 hand pumps, each with 40 features. Some of the features are binary/categorical, and some numerical. These include the location of the water pump, water source type, date of construction, the population it serves, and whether there were public meetings for the point.



a b c

Maps of Tanzania's hand pump (a) functionality, (b) water quality, and (c) quantity.

Data used in this comes from *DrivenData*, an online web platform for data science practice competitions aimed at tackling social challenges. The datasets used are a compilation of data from *Taarifa* and the *Tanzanian Ministry of Water*.
The data has been downloaded from:
'https://s3.amazonaws.com/drivendata/data/7/public/4910797b-ee55-40a7-8668-10efd5c1b960.csv'
This dataset contains data for 59,400 hand pumps, each with 40 features. Some of the features are binary/categorical, and some numerical. These include the location of the water pump, water source type, date of construction, the population it serves, and whether there were public meetings for the point.

Problem Statement:

The aim of this project is to predict the functionality of a hand pump as well as the quantity and the quality of water it outputs based on a minimum of data collected on the field. (Predicting those characteristic of a hand pump at a given point in time can help shorten the time required for managing agencies to provide support and plan targeted maintenance operations of hand pumps in Remote areas).

Solution:-

I will apply some classification algorithms Like Logistic Regression, Random Forest and Neural Networks. And then I will choose the best one among them. Models were optimized using a grid search with CV to fine tune hyper parameters. Final results were obtained using 5-fold CV with a 75%25% train –test split. Algorithms were evaluated and optimized based on the F1 score. My expected solution algorithm is Random Forest. Because I choose Benchmark model of Gradient Boosting Classifier with very high F1 scores. So to achieve F1 scores greater than this benchmark i think Random forest will do it.

Metrics:

The micro average F1 is used as an evaluation metric.

Precision = TP/TP+FP

TP=total positives
FP=false positives
FN=false negatives

Recall = TP/TP+FN

F1 Score = 2*(Recall * Precision) / (Recall + Precision)

Why F1 score:-

F1 Score is the weighted average of Precision and Recall. Therefore, this score takes both false positives and false negatives into account. Intuitively it is not as easy to understand as accuracy, but F1 is usually more useful than accuracy, especially if you have an uneven class distribution.
Related to this problem,the data set I have taken contains Uneven Class distributions. So I choose F1 score as an evaluation metric.

II. Analysis

Data Exploration:

This dataset contains data for 59,400 hand pumps, each with 40 features. Some of the features are binary/categorical, and some numerical.
The data has been downloaded from:
'https://s3.amazonaws.com/drivendata/data/7/public/4910797b-ee55-40a7-8668-10efd5c1b960.csv'

For each water pump we were provided with 39 attributes (excluding a unique record identifier) that could potentially be used as predictor variables. The response variable can assume one of three possible values:
functional: The pump is operational and not in need of maintenance
functional needs repair: The pump is operational but is in need of repair
non-functional: The pump is inoperable
54.3% of all pumps in the data set were found to be functional, while 38.4% were non-functional and 7.3% had a status of functional needs repair.

Each independent variable can be characterized as belonging to one of three classes: non- categorical numeric variables; categorical variables; and administrative variables. Variables classified as "Administrative" in nature appear to serve solely as data management attributes within the data set:

Administrative Variables	Comments
id	Unique ID for each data record
date_recorded	Date of data collection by survey company
recorded_by	Name of the data collection / survey company

While detailed analyses for each of the remaining independent variables can be found in the Data Exploration section of the Appendix, key findings from that work are described below.

Analysis of Missing Data Values

Our analysis found that every non-categorical numeric variable contains what appear to be significant quantities of invalid data values represented by zeroes:

Numeric Variables	Invalid Data Comments
amount_tsh	41,639 of 59,400 records = "0"
gps_height	20,438 of 59,400 records = "0"
population	21,381 of 59.400 records = "0"
longitude	1,812 zero values: likely invalid
latitude	1,819 values < -1: likely invalid
num_private	58,643 of 59.400 records = "0"

Eight categorical variables were also found to have missing data values (as indicated by either 'NA' values, zeroes, or blank character strings):

Categorical Var.	Distinct Values	Missing	Comments
funder	1898	3635	3582 NA's coinc. w installer
installer	2146	3655	3582 NA's coinc. w funder
subvillage	19288	371	
public_meeting	3	3334	valid values: TRUE/FALSE
scheme_management	13	3877	
scheme_name	2697	28166	
permit	3	3056	valid values: TRUE/FALSE
construction_year	55	20709	valid values: 1960 - 2013

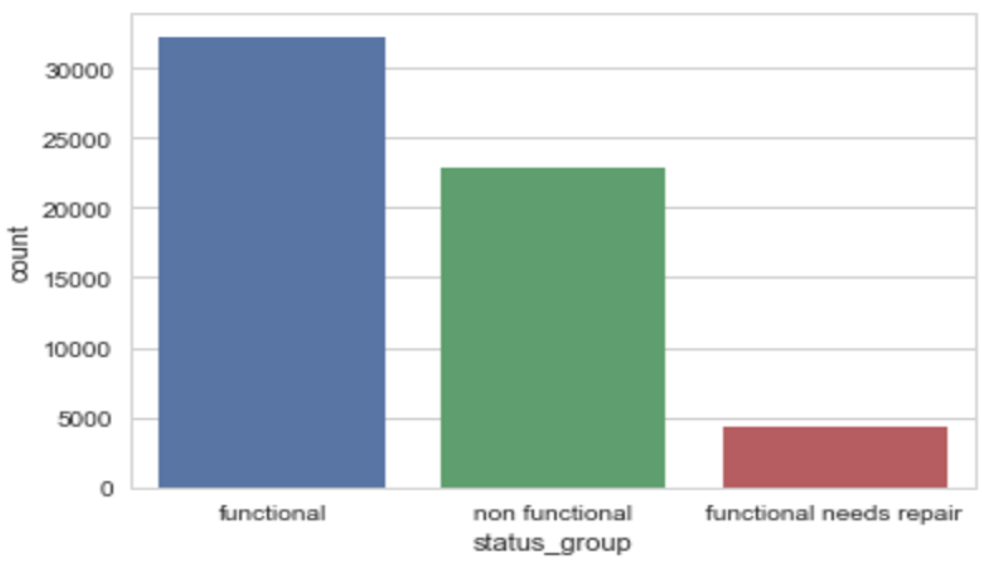
Further analysis of the missing data yielded the following key data insights:
Missing data values are found within 31,587 of the 59,400 records within the data set (53.17%).
69.2% of subvillages, 52.1% of wards, 32% of lga's, and 19% of regions have no valid amount_tsh values.
Four regions (Dodoma, Kagera, Mbeya, Tabora) were found to have no non-zero values for the following variables: amount_tsh, gps_height, construction_year, num_private, and population. These 4 regions comprise 12,115 of the 59,400 records in the data set (20.39%), including 27 of the unique lga's, 514 of the unique wards and 4644 of the unique subvillages. The 12,115 records covered by these regions represent approximately 60% of the zero values found within the gps_height (12,115 / 20,438), population (12,115 / 21,381), and construction_year (12,115 / 20,709) variables.
3,582 of the missing funder and installer values coincide with each other
1,812 of the missing gps_height values coincide with missing latitude and longitude values
43 wards have no valid public_meeting values; 75 wards have no valid scheme_management values; 73 wards and 3 lga's have no valid permit values.
The widespread incidence of missing data throughout the data set was a strong indicator of the need for the development of statistically valid data imputation algorithms for many of the affected variables.

Exploratory Visualization:

For example I have taken the first 5 samples from the data set .Then the following results are obtained.

	id	status_group
0	69572	functional
1	8776	functional
2	34310	functional
3	67743	non functional
4	19728	functional

And for the total data set the result will be like…



Here I split the data into 75% of training data and 25% of test data (after data preprocessing).

I use this code to split the data….

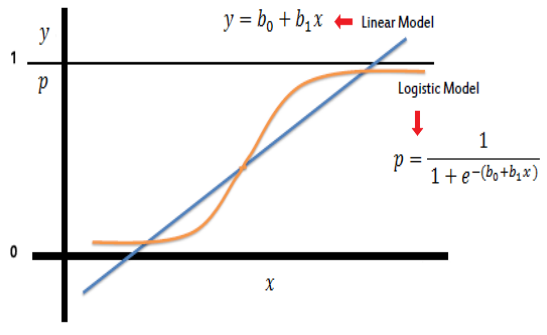
X_train, X_test, y_train, y_test = sklearn.model_selection.train_test_split(training, test, test_size = 0.25, random_state = 0)

Algorithms and Techniques:-

1)Logistic Regression:-

Reason for choosing : Logistic regression was chosen because it is a robust learning algorithm that makes few, and usually reasonable, assumptions about the data. The penalty factor and the type of regularization were optimized. Best performance for this algorithm on all three classification tasks was achieved for an L2 regularization with penalty factor of 1.0.

Logistic regression predicts the probability of an outcome that can only have two values (i.e. a dichotomy). The prediction is based on the use of one or several predictors (numerical and categorical). A linear regression is not appropriate for predicting the value of a binary variable for two reasons:



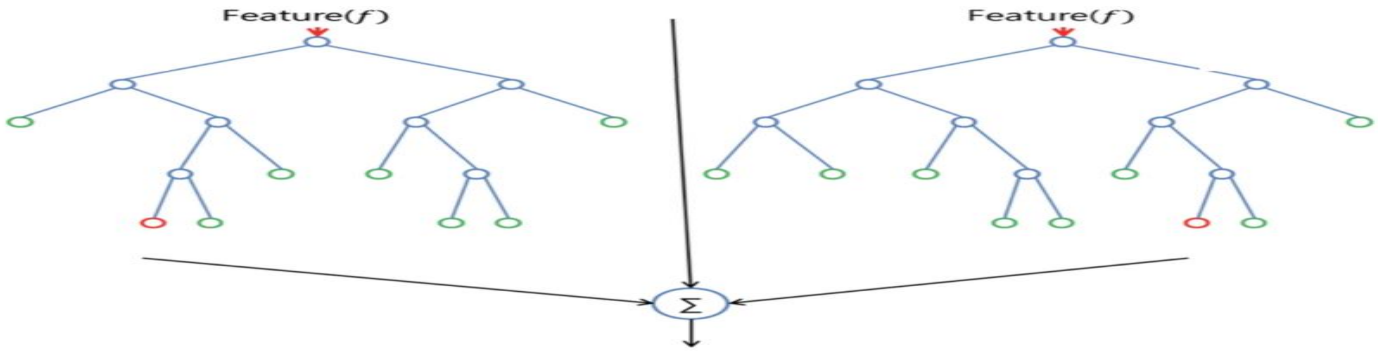
- A linear regression will predict values outside the acceptable range (e.g. predicting probabilities outside the range 0 to 1)
- Since the dichotomous experiments can only have one of two possible values for each experiment, the residuals will not be normally distributed about the predicted line.

On the other hand, a logistic regression produces a logistic curve, which is limited to values between 0 and 1. Logistic regression is similar to a linear regression, but the curve is constructed using the natural logarithm of the “odds” of the target variable, rather than the probability. Moreover, the predictors do not have to be normally distributed or have equal variance in each group.

2) Decision Trees:
Reason:
A decision tree model seemed a particularly promising idea given the number of features used in our algorithm, especially after the OHE process. Having many features, none of which have an obvious effect on the output alone, means that the causal relationship between the features and the output might come from different combination of the features that cannot be modelled well by algorithms that rely on assumptions about data distributions. We tried the Random Forest (RF), AdaBoost, and Bagging and after tuning the parameters of each algorithm, RF performed the best.

Random Forest:

One big advantage of random forest is, that it can be used for both classification and regression problems, which form the majority of current machine learning systems. I will talk about random forest in classification, since classification is sometimes considered the building block of machine learning. Below you can see how a random forest would look like with two trees:



Random Forest has nearly the same hyperparameters as a decision tree or a bagging classifier. Fortunately, you don’t have to combine a decision tree with a bagging classifier and can just easily use the classifier-class of Random Forest. Like I already said, with Random Forest, you can also deal with Regression tasks by using the Random Forest regressor.

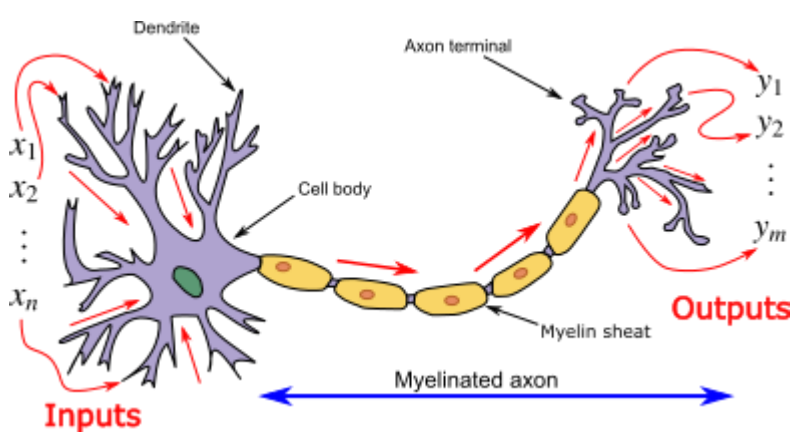
Random Forest adds additional randomness to the model, while growing the trees. Instead of searching for the most important feature while splitting a node, it searches for the best feature among a random subset of features. This results in a wide diversity that generally results in a better model.

Therefore, in Random Forest, only a random subset of the features is taken into consideration by the algorithm for splitting a node. You can even make trees more random, by additionally using random thresholds for each feature rather than searching for the best possible thresholds (like a normal decision tree does).

3)Neural Networks:

Reason: Our last algorithm was a NN because of the NN's ability to generalize and to respond to unexpected patterns. During training, different neurons are taught to recognize various independent patterns and then the combination of all the neurons manages to capture all those different patterns and combine them in one final output node. Since our dataset likely contained unexpected and difficult interactions, this was a promising choice

Artificial neural networks (ANNs): ANNs are computing systems vaguely inspired by the biological neural networks that constitute animal brains and humans. these systems “learn” to perform tasks by considering
examples, generally without being programmed with any task-specific rules.
A typical brain contains something like 100 billion miniscule cells called neurons (no-one knows exactly how many
there are and estimates go from about 50 billion to as many as 500 billion).
Each neuron is made up of a cell body (the central mass of the cell) with a number of connections coming off it:
numerous dendrites (the cell's inputs—carrying information toward the cell body) and a single axon (the cell's
output—carrying information away). Neurons are so tiny that you could pack about 100 of their cell bodies into a
single millimeter. Inside a computer, the equivalent to a brain cell is a tiny switching device called a transistor. The
latest, cutting-edge microprocessors (single-chip computers) contain over 2 billion transistors; even a basic
microprocessor has about 50 million transistors, all packed onto an integrated circuit just 25mm square (smaller
than a postage stamp)!



ANN is a set of connected neurons organized in layers:

・input layer: brings the initial data into the system for further processing by subsequent layers of artificial neurons. Like dendrites in human neuron

・hidden layer: a layer in between input layers and output layers, where artificial neurons take in a set of weighted inputs and produce an output through an activation function. Like nucleus in human neuron.

・output layer: the last layer of neurons that produces given outputs for the program. Like axon in human neuron

Benchmark:-

The Benchmark model I chosen is the gradient Boosting classifier. Because some previous works have been done on this project using Gradient Boosting Classifier and it gives test accuracy score of 79.37 and train accuracy score of 92.38.
And F1 score

	Test(75% of original)	Train(25% of original)
Functionality	80.5	71
Quality	85.5	74.9
Quantity	92.6	92.4

Link to the previous project is: https://towardsdatascience.com/predicting-the-functional-status-of-pumps-in-tanzania-355c9269d0c2

III. Methodology

Data Preprocessing:

- **Dropping The Features With Similar Representation of Data(Duplicates):**

The group of features of (extraction_type, extraction_type_group, extraction_type_class), (payment, payment_type),
(water_quality, quality_group), (source, source_class), (subvillage, region, region_code, district_code, lga, ward), and (waterpoint_type, waterpoint_type_group) all contain similar representation of data in different grains. Hence, we risk over fitting our data during training by including all the features in our analysis, which can be dropped.

- id can be droped because it is unique for each instance.
- num_private is ~99% zeros

- Data Cleaning and Analysis

- Looking at the data, some of the features that seemed discriminative based on human intuition. amount_tsh (amount of water available to water point), gps_height, basin, installer, population, scheme_management, construction year, extraction_type, management_group, water_quality, payment type, source, and waterpoint_type seemed like they could be extremely important in identifying the pump status.
- There are null values in our features which are needed to be updated for better training of our model. And i updated the null values with the mean of the specific feature.
- And I use “One Hot Encoding” for the categorical data features. By using OHE process I manage all the categorical values.
- Here I normalize the numerical features(binary form)
- Resampling the data to handle the class imbalance
- For example editing variable from hundreds of categories to “funder = 1” if village/villagers, 0 otherwise.
- After Dropping the unwanted features and cleaning the data set we are left with 20 features which can be used to train our model. The features are which we are going to use for training are:

['amount_tsh', 'days_since_recorded', 'funder', 'gps_height',
'installer', 'basin', 'subvillage', 'population', 'public_meeting',
'scheme_management', 'scheme_name', 'permit', 'construction_year',
'extraction_type', 'payment_type', 'water_quality',
'quantity_group', 'source_type', 'source_class', 'waterpoint_type',
'waterpoint_type_group']

Implementation

I have implemented this problem with 3 classification algorithms
#here I mentioned the code which I think was important and also complicated when I implemented that.

1) Logistic regression:-

The steps involved in this Logistic Regression are as follows

- optimizing hyper parameters:- We used GridSearchCV which performs k-fold cross validation (k=5 for us) and searches a grid of specified parameters to find the best parameters.
- Then calculates the Micro averaged f1 score using function f1_score function in metrics package.
- Then performs the 5-fold cv on the test set.
- And finally prints the F1 score for both Training and Test set.

Compications:

The complicated code part is mentioned here

- Here I choose the parameters as
  param_dist = {
  'penalty': ['l1','l2'],
  'C': [0.001,0.01,0.5,1]
  }
- And for best classifier found with GridSearchCV can be calculated as
  lr_best = lr_search.best_estimator_
- Here I choose K-fold cross validation for fitting the data, but it takes a very huge amount of time.

2) Random Forests:-

The steps involved in this implementation are as follows

- The first step I have done is optimize the hyper parameters using GridSearchCV to find the best parameters.
- Calculated the F1 score
- Performs 5-fold cv on test set
- And prints the results.

Compications:

- I choose the parameters as
  parameters = {'n_estimators':(100,70,50),'max_depth':(30,25,20,5)}
- Here also K-fold cross validation takes lot of time

3) Neural Networks:-

The steps involved are:

- I choose the MLPclassifier
- And the same procedure is repeated i.e., optimizing the hyper parameters using GridSearchCV
- Perform 5-fold CV
- Finally Prints the results.

Compications:

- parameters ={
  'learning_rate': ["constant", "invscaling", "adaptive"], 'hidden_layer_sizes': [(138,60,2), (100,10), (60,5,1), (75,30,5), (138)], 'activation': ["logistic", "tanh"]
  }
- Here GridSearchCV takes infinite time

IV. Results

1) Logistic Regression:

F1 score results are:

	Train(75% of original)	Test(25% of original)
Functionality	65.37%	64.3%
Quality	68.6%	57.0%
Quantity	77.0%	59.8%

Classification report:

Comparing F1 scores:

- For functionality class: Train split has 65.37% F1 score and test split has 64.3% F1 score. So for low f1 score the classification algorithm is not that much efficient

- For Quality class: Test split has 68.6% F1 score and train split has 57% F1 score. So for low f1 score the classification algorithm is not that much efficient
- For Quantity class: Test split has 77% F1 score and train split has 59.8% F1 score. So for low f1 score the classification algorithm is not that much efficient.

➢ For all the three classes the algorithm results are not as that the predicted. I think that this classification algorithm doesn't suites this problem.

.

Justification:

When compared to my Benchmark model (Gradient Decent Classifier) the present classification algorithm F1 score are very low for all the three classes (Functionality, Quality, Quantity).So this algorithm is not achieves the result. So my assumption is wrong.

2) Random Forests:

F1 scores are:

	Train(75% of original)	Test(25% of original)
Functionality	86.2	76.8
Quality	91.9	78.9
Quantity	97.9	97.4

Classification report:

- For functionality class: Test split has 76.8% F1 score and train split has 86.2% F1 score. So for a high f1 score the classification algorithm is very much efficient and it clearly predicts the labels.
- For Quality class: Test split has 78.9% F1 score and train split has 91.9% F1 score. So for very high f1 score the classification algorithm is very much efficient.
- For Quantity class: Test split has 97.4% F1 score and train split has 97.9% F1 score. So for high f1 score the classification algorithm is very much efficient

Justification:

Compared to the Benchmark model the F1 scores for all the three classes are a bit high and all are above 90%(except functionality class).So F1 score are very good.

So I can confidently say this algorithm can be used. My assumption is right for this classification.

3)Neural Networks:

F1 score is

	Train(75% of original)	Test(25% of original)
Functionality	70.5	67.3
Quality	79.6	66.4
Quantity	91	73.3

Classification report:

- For functionality class: Test split has 67.3% F1 score and train split has 70.5% F1 score. So for low f1 score the classification algorithm is not that much efficient
- For Quality class: Test split has 66.4% F1 score and train split has 79.6% F1 score. So for low f1 score the classification algorithm is not that much efficient.
- For Quantity class: Test split has 73.3% F1 score and train split has 91% F1 score. So for low f1 score the classification algorithm is not that much efficient

Justification:

Compared to the bench mark model f1 score this classification algorithm f1 scores are very low. So this algorithm doesn't give valid predictions. But this algorithm is better than the Linear Regression algorithm.

Which algorithm to choose:-

By comparing the entire three algorithms the Random Forest algorithm gives very good F1 scores for all the three classes' .And also these scores are greater than the Benchmark results.

So without any confusion I can confidently say Random Forest Algorithm best suits this problem.

V. Conclusion

Here there is a comparison between the F1 scores of the three classifiers that I have chosen earlier.

	Functionality		Quantity		Quality	
	Train	Test	Train	Test	Train	Test
Linear Regression	65.37%	64.3%	68.6%	57%	77%	59.8%
Random Forest	86.2%	76.8%	91.9%	78.9%	97.9%	97.4%
Neural Networks	70.5%	67.3%	79.6%	66.4%	91 %	73.3%

As it will be clearly shown that Random Forest algorithm have higher results.

Reflection:-

In this capstone project I selected this problem because as my problem. Because I really interested in this type of real world problems and every time I interested to do this type of projects which will help the society in any manner.

While doing this project I came across these things:

- When I go through the present problem data set there is lot of noise in the data
- Noise means duplicates, Inconsistent values and also erroneous values
- And I learn how to remove these noises by applying machine leaning techniques in a very simple manner(in simple terms data preprocessing)
- How to effectively use the given data into training and testing data parts by splitting them.
- How to use different Machine learning classifiers.
- In this problem I chose three classifiers and it helped me very lot because it covers a lot of stuff.
- And also this project helped me to choose different classifiers related to different problem statements.
- Visualization results are very easy to conclude to a statement.
- And it helped me in learning Machine Learning efficiently.

Improvement:

- 1) One thing that can be improved from my models is we can use K fold method for splitting data while fitting model, when we have enough time.

- 2) Another thing I improved is “focusing on only the useful features instead of all the features in the data set”.

Final Note:

- When I start this project I think that the results of the three algorithms will be approximately same and all will be greater than the Benchmark model which I had chosen from earlier published paper. But after done with my implementation the benchmark model I have chosen is not a bad one. It also gives good results when compared with 2 models out of 3.And my earlier assumption was somewhat wrong.
- And finally I concluded my statement by saying that Random Forest algorithm gives best results. But doesn’t confidently say this is the “Best” because in future there may be chance of evolving some algorithms which gives accuracy more than my current results.