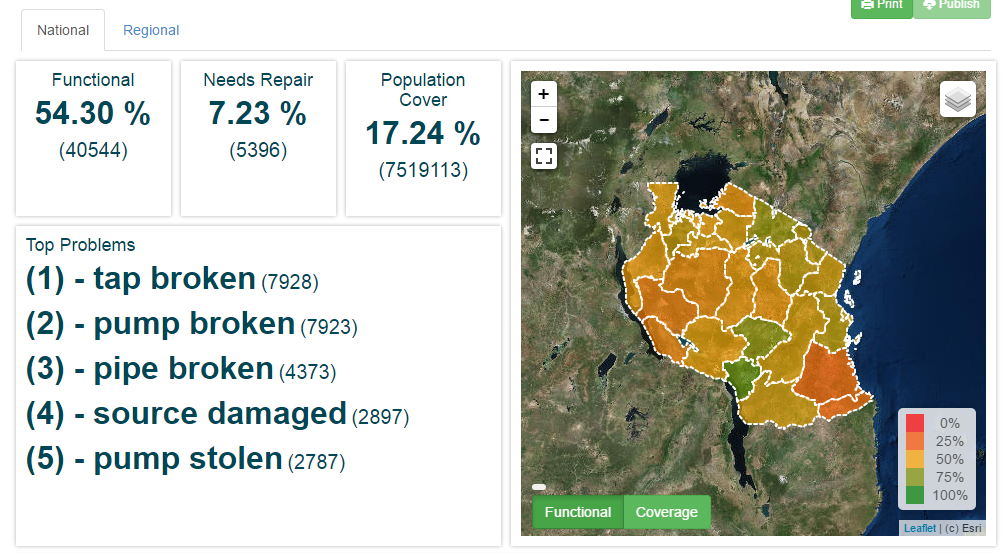
Machine Learning Project: Final Report

**1. Problem Description:**

In Tanzania villages, we have a number of water pumps provided. Unfortunately, not all are in working condition. Moreover, many require repair. Based on the given compiled dataset, we need to predict, based on various factors such as the kind of pump operating, when it was installed, and how it is managed, sub-village, region, funder etc, that if the test instance falls under which class label. The class labels are as follows: functional, non-functional, functional but needs repair.

**2. Related Work:**

According to the website: <http://dashboard.taarifa.org/#/dashboard> the national data for the current situation for water pumps in Tanzania is as follows: 

This data has been collected, along with a large number of other attributes, which have been described below.

**3. Dataset Details:**

* No of features - 39
* Instances - 59400
* Data Distribution - See the attached .pdf with R code to find correlation of different attributes with the class labels. Also the histograms depicting relation of class labels with the most important attributes are attached in the .pdf
* Feature summary is as follows:
  + amount\_tsh - Total static head (amount water available to waterpoint)
  + date\_recorded - The date the row was entered
  + funder - Who funded the well
  + gps\_height - Altitude of the well
  + installer - Organization that installed the well
  + longitude - GPS coordinate
  + latitude - GPS coordinate
  + wpt\_name - Name of the waterpoint if there is one
  + num\_private -
  + basin - Geographic water basin
  + subvillage - Geographic location
  + region - Geographic location
  + region\_code - Geographic location (coded)
  + district\_code - Geographic location (coded)
  + lga - Geographic location
  + ward - Geographic location
  + population - Population around the well
  + public\_meeting - True/False
  + recorded\_by - Group entering this row of data
  + scheme\_management - Who operates the waterpoint
  + scheme\_name - Who operates the waterpoint
  + permit - If the waterpoint is permitted
  + construction\_year - Year the waterpoint was constructed
  + extraction\_type - The kind of extraction the waterpoint uses
  + extraction\_type\_group - The kind of extraction the waterpoint uses
  + extraction\_type\_class - The kind of extraction the waterpoint uses
  + management - How the waterpoint is managed
  + management\_group - How the waterpoint is managed
  + payment - What the water costs
  + payment\_type - What the water costs
  + water\_quality - The quality of the water
  + quality\_group - The quality of the water
  + quantity - The quantity of water
  + quantity\_group - The quantity of water
  + source - The source of the water
  + source\_type - The source of the water
  + source\_class - The source of the water
  + waterpoint\_type - The kind of waterpoint
  + waterpoint\_type\_group - The kind of waterpoint

**4. Pre-processing Techniques:**

Preprocessing is a very important part of this project. We have taken 2 steps for the same.

1. Remove all non-available data (missing values). This ensures all the data we have is complete and to the mark, resulting in more accurate results.

2. Check correlation of all attributes with the class labels so that we can reduce the attributes that are not required. This reduces the dimensionality of the dataset. The results are less complex and more efficient results. Basically, correlation shows the relation of each attribute with the class label. A higher value of absolute magnitude means the attribute is highly related to the class label and should not be ignored. Whereas attributes with a correlation value tends to 0 means it can be removed and ignored. Using this, we removed certain attributes to reduce complexity of the dataset.

The specific steps we took on our dataset for preprocessing include the following:

* Removed the 'recorded' column from the data frame as it is not related
* Removed the 'scheme name' due to multiple distinct values and unnecessary levels
* In 'payment' and 'payment\_type' some of the categories are renamed but are of the same type, hence we removed payment column to reduce redundancy
* Removed 'quantity group' as 'quantity' and 'quantity group' have same values
* Removed 'waterpoint\_type\_group' since its values were the same as 'waterpoint\_type'.

**5. Proposed Solution and method:**

We decided to use SVM (Support Vector Machine) as our technique to combat this problem statement. We have been given the class labels here and can thus use this supervised learning model to overcome the issue of predicting the status of the new pump given to be tested.

Simple explanation of SVM: Here we will plot data in a nonlinear dimension, and consider the plotted points closest to the hyperplanes separating the data as our support vectors. Here we choose a non-linear kernel, as our data is quite complex and there will be no clear separators for the same. Moreover, since our data is supervised (i.e. class labels are given) we can use this technique with ease.

Pros of this method:

* Accuracy
* Works well on smaller cleaner datasets,
* It can be more efficient because it uses a subset of training points.

**6. Experimental Results and Analysis:**

* SVM isn’t suited to larger datasets as the training time with SVMs can be high, Less effective on noisier datasets with overlapping classes.
* One major problem faced with SVM was runtime to create model. We earlier tried our approach, with SVM and it was too time-consuming as the dataset contained almost about 60,000 instances. Also, low accuracy was another problem. Further, SVM does not work for dataset having data with too many levels and/or unscaled data. Hence, we implemented with random forest.
* Here we can vary certain parameters of input to change the accuracy of our output. These include the kernel (sigmoid, radial, polynomial etc…), gamma value, cost. A small gamma means a Gaussian with a large variance so the influence is more. If gamma is large, then variance is small implying the support vector does not have wide-spread influence. Technically speaking, large gamma leads to high bias and low variance models, and vice-versa. We have attached a table showing our experiments with the parameters and hence our deduction of the the best parameters to use based on accuracy:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| #Run | Classifier | mtries | Seed | ntrees | nfolds | Accuracy (%) | Accuracy on DrivenData test instances (%) |
| 1 | Random Forest | 8 | 12345 | 1200 | 3 | 85.19 | 69 |
| 2 | Random Forest | 8 | 14000 | 1200 | 2 | 82.67 | 68 |
| 3 | Random Forest | 6 | 14000 | 1100 | 2 | 84.29 | 68 |
| 4 | Random Forest | 6 | 13000 | 1000 | 3 | 85.37 | 69 |
| **5** | **Random Forest** | **7** | **13000** | **950** | **4** | **86.10** | **68** |

**7. Coding Language and Technique:**

Coding Language: R

Libraries used:

H2O:

* H2O is an open-source, fast, scalable and all around wonderful platform for machine learning. It is Java-based but it has an R interface which I use for the Pump it Up competition. (The caret package is great as well and it contains more models for prediction.)
* H2O is optimized for doing “in memory” processing of distributed, parallel machine learning algorithms H20 along with data.table can be used to run parallel algorithms on large datasets.
* H20 along with data.table can be used to run parallel algorithms on large datasets.

Technique: RANDOM FOREST

* Unfortunately, SVM takes too long to create a model on such a large dataset and due to the cons above, the accuracy was not as required. So we needed to change the technique to **Random Forest**.
* Random forests is an ensemble method for classification, which constructs a lot of decision trees at training time and shows the final resulting class as the mode of the class given as out as output by individual.
* Each tree gets trained on roughly 2/3rd of total training data. Classes are drawn at random with replacement from the original data. Out of the predictor variables which we filtered by feature selection, some predictor variables are taken at random and the best split on these is used to split the node.
* For each tree, using the leftover data, we calculate the misclassification rate. We aggregate this error from all trees to determine overall error rate for classification.

Why **Random Forests**?

* It is one of the most accurate learning algorithms available.
* Most importantly in our case, it runs very efficiently on large databases.
* Handling of thousands of inputs is done with ease.
* It gives estimates of what variables are crucial for the classification.
* It has an effective method to estimate missing data and it can maintain accuracy even when large data is missing from the dataset.

However, sometimes random forests tend to overfit some datasets with noisy classification. Hence, cross-validation is used in the code to take care for it. The parameter for that in the code is called “n-folds” which when adjusted to an appropriate value gives us the optimal result.

**8. Conclusion**:

The best accuracy obtained through our experiments and analysis is 85.16% which is highlighted in the accuracy table. The table also depicts the accuracy obtained when we submitted our code on DrivenData.org. Further scope of the project could include further feature engineering and change of parameters.

**9. Contribution**:

All team members contributed equally in all parts of the project. It was a group effort at every stage.

**10. References:**

* <http://dashboard.taarifa.org/#/dashboard>
* <https://www.drivendata.org/competitions/7/page/25/>
* <http://machinelearningmastery.com/pre-process-your-dataset-in-r/>
* <http://www.listendata.com/2014/11/random-forest-with-r.html>
* <http://blog.revolutionanalytics.com/2014/04/a-dive-into-h2o.html>
* http://h2o-release.s3.amazonaws.com/h2o/rel-lambert/5/docs-website/Ruser/rtutorial.html