

CS 6375.003

Machine Learning

Spring 2018

Final Project

**Pump It Up: Data Mining the Water Table**

**Submitted by:**

Ankita Patil (asp160730)

Abhilash Gudasi (abg160130)

Shiva Ranga Chawala (sxc167630)

**Instructor:**

Anurag Nagar

# **Table of contents**

Introduction………………………………………………………………………………………………………………………………3

Project Description……………………………………………………………………………………………………………………4

Target Audience………………………………………………………………………………………………………………………. 4

Dataset Description…………………………………………………………………………………………………………………. 5

Languages, Packages, Tools Used………………………………………………………………………………………………8

Workflow…………………………………………………………………………………………………………………………………..9DataExploration……………………………………………………………………………………………………………………...10

Preprocessing/Feature Engineering……………………………………………………………………………………….. 13

Algorithms Implemented………………………………………………………………………………………………………. 16

Training and Validation………………………………………………………………………………………………………….. 16

Results…………………………………………………………………………………………………………………………………… 19

Challenges/Issues………………………………………………………………………………………………………………….. 22

Conclusion…………………………………………………………………………………………………………………………….. 23

References…………………………………………………………………………………………………………………………….. 23

# **Introduction**

The major motivation for the project is based on the article - [Tanzania: Addressing the Challenge of Water](http://allafrica.com/stories/201605240460.html) [1]. According to this article, Tanzania has been facing water shortage problem for many years. Many other countries such as Yemen, Libya, Jordan, Western Sahara, Djibouti are facing water crisis similar to Tanzania [2]. Hence in order to inspect various factors which are causing water crisis we have chosen the project – Pump It Up: Data Mining the Water Tables from the DrivenData competition [4] - <https://www.drivendata.org/competitions/7/pump-it-up-data-mining-the-water-table/>

The challenge is to predict the operating condition of water pumps in Tanzania installed by various firms based on various factors such as type of waterpoint, when it was installed/constructed, its maintenance, population around the well, etc. We are trying to address the water shortage problem by predicting operating condition of waterpoints using machine learning techniques.

A group of people standing in front of a crowd

Description generated with very high confidence

# **Project Description**

The data for Pump It Up challenge is gathered from Taarifa which aggregates the data from the Tanzanian Ministry of Water. Using the datasets obtained from the competition, we have analyzed various features associated with a waterpoint, performed feature engineering, data cleaning and built machine learning models. These models are used predict whether the water pump is functional, non-functional or needs repair.

A screen shot of a computer

Description generated with very high confidence

# **Target Audience**

The prediction given by machine learning models from this project can be used by Government officials of Tanzanian Ministry of Water to make informed decisions, enforce laws and regulations for better management of water pumps. This project can help them to solve the water-crisis problem in efficient way. The analysis made through visualization can help them to identify which factors are majorly responsible for the operating condition a of waterpoint. This project can also be used by other countries which are facing water-crisis problem to make strategies to deal with water shortage problem.

# **Dataset Description**

The dataset [5] for this project is downloaded from <https://www.drivendata.org/competitions/7/pump-it-up-data-mining-the-water-table/data/>

|  |  |
| --- | --- |
| **Number of Features** | **40** |
| **Number of Labels** | 3 |
| **Number of instances** | 59400 |

There are **59400 instances** in the dataset where each instance has about **40 features**. The label for each instance is given in separate file.

**Feature Description**

Following table describes total 40 features present in the dataset.

|  |  |
| --- | --- |
| **Feature** | **Description** |
| **amount\_tsh** | Total static head (amount water available to waterpoint) |
| **date\_recorded** | The date the row/record 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 |

**Description of labels**

There are **3 classes** which are described in the following table.

|  |  |
| --- | --- |
| **Labels** | **Description** |
| **functional** | the waterpoint is operational and there are no repairs needed |
| **functional needs repair** | the waterpoint is operational, but needs repairs |
| **Non-functional** | the waterpoint is not operational |

# **Languages, Tools, Packages Used**

# **Workflow**

1. Data Exploration

* Univariate Analysis
* Correlation Graph

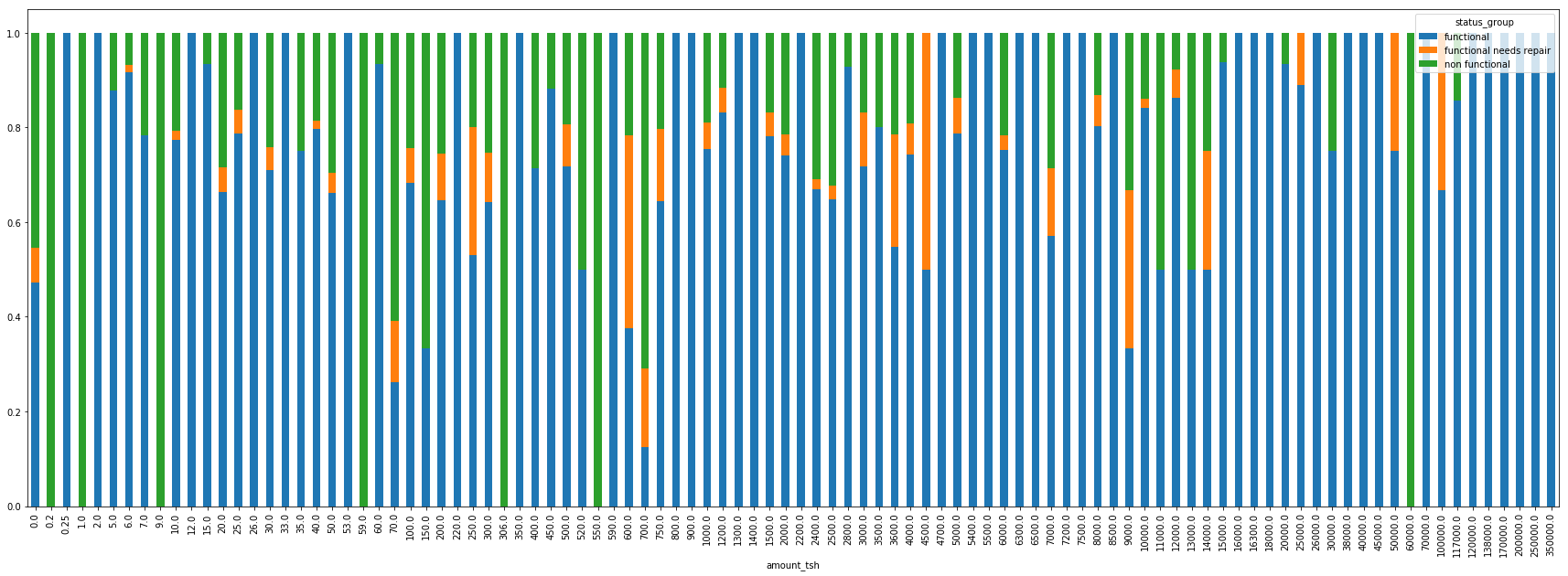
1. Pre-processing/Feature Engineering
2. Algorithm Implementation

* Model training and parameter tuning using GridSearchCV
  + With train-test split
  + With k-fold cross validation

1. With the trained model, predict the accuracy on the test data

# **Data Exploration**

**Amount\_tsh vs Status\_group**



The above stacked graph depicts the operational condition of water pumps(**status\_group**) vs the amount of water point available in waterpoint (**amount\_tsh**). We observe that, on an average the water pumps with less amount of water are non-functional. However, for few waterpoints even though the amount of water level is moderate, the water pumps are non-functional or needs repair.

**Basin vs status\_group**

A close up of a logo

Description generated with very high confidence

From above bar graph, we observe that the operational condition of waterpoint (**status\_group**) is uniformly spread across basins (**basin**) in Tanzania.

**Construction\_year vs status\_group**

A picture containing writing implement, stationary, pencil

Description generated with very high confidence

We observe that there are more functional waterpoints which were constructed in recent years. We observe that the pattern for functional waterpoints is increasing over the years whereas the pattern for non-functional waterpoints is decreasing over the years.

**Correlation Graph before pre-processing**

A close up of a logo

Description generated with high confidence

The above correlation graph describes the correlation between all the features. We observe that, most of the features are highly correlated with other features. This implies that all highly correlated features are influencing the operating condition of waterpoint in similar way. At this stage where the dataset is not yet preprocessed we cannot make solid conclusions on the operating condition of a waterpoint for a particular instance. Hence, we need to pre-process the data.

# **Pre-processing/ Feature Engineering**

The dataset contains about 40 features in which few are categorical and few are numerical. The data entries for few instances are also missing.

**Merging 2 datasets**

As mentioned in **Dataset Description** section, there is a separate file which has corresponding labels for instances. Hence the two files – a file containing instances and another file containing corresponding labels are merged into a single dataset on “id” attribute.

**Dealing with missing values**

For the features – latitude and longitude, few values are 0. Missing latitude and longitude values are obtained using the place name mentioned as feature **lga** using Geocoder package [3]. [ GeoCoder.google(lga) ]. Command to install GeoCoder package -> pip install –user geocoder

For the feature gps\_height, few values are 0. The missing values are obtained from latitude and longitude features in the dataset using GeoCoder package. [GeoCoder.elevation(latitude, longitude)]. For this, API key is required.

**Normalization using MinMaxScaler**

For features such as amount\_tsh, latitude, longitude, gps\_height and population, we have performed normalization in the range 0 to 1 using MinMaxScaler from scikit-learn.

**Dealing with categorical features using LabelEncoder**

The categorical features in the dataset are listed below:

|  |  |  |
| --- | --- | --- |
| Features | | |
| basin | lga | quality\_group |
| construction\_year | management | quantity |
| district\_code | payment | region |
| extraction\_type | payment\_type | region\_code |
| extraction\_type\_class | permit | scheme\_management |
| scheme\_name | source | source\_class |
| source\_type | subvillege | water\_qaulity |
| waterpoint\_type | waterPoint\_type\_group |  |

For converting this categorical data into numerical data, we have used LabelEndoder technique from scikit-learn.

**Merging features:**

From correlation graph, we observe that, construction\_year and date\_recorded are highly correlated and therefore we have merged construction\_year and date\_recorded into one single feature but we don’t want to lose any information, so we have replaced date\_recorded feature with date\_recorder-construction\_year and drop construction\_year feature.

**Correlation Graph after pre-processing**

A picture containing object

Description generated with high confidence

From this correlation graph we observe that, payment\_type and payment, extraction\_type and extraction\_type\_class as well as water\_point\_type and water\_point\_group are highly correlated. Therefore, we have dropped extraction\_type\_class, payment\_type and waterpoint\_type\_group.

After pre-processing, we observe in the correlation graph that the features are less correlated with each other. Before preprocessing, many features were overlapping in the correlation graph. However, after pre-processing we can say that the features are independently contributing in predicting the operational condition of a waterpoint.

# **Algorithms Implemented**

Justification of selecting supervised machine learning algorithms

By exploring the datasets, we observe that for each instance, a label is provided. When data with label is provided, supervised machine learning algorithms can be applied.

Following five algorithms are used in model creation for **Pump It Up: Data Mining the Water Table** dataset

1. Logistic Regression
2. Support Vector Machine
3. Adaboosting
4. Neural Net
5. Random Forest

# **Training and Validation**

All the models created using algorithms such as logistic regression, support vector machine, adaboosting, neural network, random forest are tested:

1. With train-test split 75-25% for Neural Net, Adaboost, Random forest
2. With train-test split 67-33% for Logistic Regression, Support Vector Machine
3. Using k-fold cross validation where k = 10.

We observe that, for algorithms such as Adaboosting, Random Forest, accuracy with k-fold were comparatively better as expected because each time we are training on different set of training data and testing on n/k data where n is total instances.

|  |  |
| --- | --- |
| **Logistic Regression** | **Accuracy** |
| multi\_class='multinomial', solver='lbfgs', penalty='l2',max\_iter=1000 | 58.92% |
| multi\_class='multinomial', solver='lbfgs', penalty='l2',max\_iter=2000 | 60.31% |
| multi\_class='multinomial', solver='lbfgs', penalty='l2',max\_iter=2500 | 60.37% |
| multi\_class='multinomial', solver='lbfgs', penalty='l2',max\_iter=3000 | 60.42% |
| multi\_class='multinomial', solver='lbfgs', penalty='l2',random\_state=20,max\_iter=4000 | 60.41% |
| multi\_class='multinomial', solver='lbfgs', penalty='l2',random\_state=20,max\_iter=5000 | 60.41% |
| multi\_class='multinomial', solver='newton-cg', penalty='l2',random\_state=10,max\_iter=2000 | 68.53% |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  | **Support Vector Machine** | |  |  |
| C | Kernel | gamma | degree | random\_state | max\_iter | Accuracy |
| 1 | sigmoid | 0.1 | 3 | 10 | 10000 | 56.16 |
| 5 | rbf | 0.01 | 3 | 10 | 1000 | 36.8 |
| 10 | poly | 0.001 | 5 | 100 | 10000 | 38.57 |
| 1 | poly | 0.0001 | 7 | 100 | 1000 | 46.49 |
| 1 | sigmoid | 0.01 | 3 | 100 | 1000 | 37.27 |

|  |  |
| --- | --- |
| **Neural Net** | **Accuracy** |
| solver='adam',activation='relu',learning\_rate='constant',learning\_rate\_init=0.01,  alpha=0.0001,hidden\_layer\_sizes=(100) | 66.68% |
| solver='lbfgs',activation='tanh',learning\_rate='constant',learning\_rate\_init=0.01,  alpha=0.0001,hidden\_layer\_sizes=(100) | 55.32% |
| solver='lbfgs',activation='tanh',learning\_rate='adaptive',learning\_rate\_init=0.1,  alpha=0.01,hidden\_layer\_sizes=(100,40,20) | 56.09% |
| solver='adam',activation='relu',learning\_rate='constant',learning\_rate\_init=0.01,  alpha=0.01,hidden\_layer\_sizes=(100,100,25) | 58.84% |
| solver='sgd',activation='relu',learning\_rate='invscaling',learning\_rate\_init=0.01,  alpha=0.1,hidden\_layer\_sizes=(100,40,25,25) | 56.04% |

|  |  |
| --- | --- |
| **AdaBoost** | **Accuracy** |
| algorithm='SAMME.R',learning\_rate=0.01,n\_estimators=100, base\_estimator=DecisionTreeClassifier(criterion='gini', max\_depth=5, min\_samples\_leaf=1, min\_samples\_split=2, splitter='best' | 73.62% |
| algorithm='SAMME.R',learning\_rate=0.1,n\_estimators=125, base\_estimator=DecisionTreeClassifier(criterion='gini', max\_depth=5, min\_samples\_leaf=1, min\_samples\_split=2, splitter='best' | 77.5% |
| algorithm='SAMME.R',learning\_rate=0.001,n\_estimators=50, base\_estimator=DecisionTreeClassifier(criterion='gini', max\_depth=5, min\_samples\_leaf=4, min\_samples\_split=4, splitter='best' | 72.78% |
| algorithm='SAMME.R',learning\_rate=0.01,n\_estimators=400, base\_estimator=DecisionTreeClassifier(criterion='entropy', max\_depth=5, min\_samples\_leaf=1, min\_samples\_split=2, splitter='best' | 76.2% |
| algorithm='SAMME.R',learning\_rate=0.01,n\_estimators=600, base\_estimator=DecisionTreeClassifier(criterion='entropy', max\_depth=40, min\_samples\_leaf=1, min\_samples\_split=2, splitter='best' | 74.81% |
|  |  |

|  |  |
| --- | --- |
| **Random Forest** | **Accuracy** |
| n\_jobs=-1, max\_features='sqrt', oob\_score = True,n\_estimators=500,max\_depth=30,min\_samples\_leaf=2 | 82.3% |
| n\_jobs=-1, max\_features='log2', oob\_score = True,n\_estimators=500,max\_depth=30,min\_samples\_leaf=2 | 81.31% |
| n\_jobs=-1, max\_features='sqrt', oob\_score = True,n\_estimators=700,max\_depth=40,min\_samples\_leaf=2 | 81.41% |
| n\_jobs=-1, max\_features='sqrt', oob\_score = True,n\_estimators=100,max\_depth=30,min\_samples\_leaf=4 | 81.37% |
| n\_jobs=-1, max\_features='sqrt', oob\_score = True,n\_estimators=100,max\_depth=10,min\_samples\_leaf=6 | 77.25% |

# **Results**

|  |  |  |
| --- | --- | --- |
| **Algorithm** | **Accuracy** | **Best Parameters** |
| Support Vector Machine | 56.16% | C: 1.0 |
|  |  | kernel: sigmoid |
|  |  | degree: 3 |
|  |  | gamma: 0.1 |
|  |  | random\_state: 10 |
|  |  | max\_iter: 10000 |
| Neural Net | 66.67% | solver: adam |
|  |  | activation: relu |
|  |  | learning\_rate: constant |
|  |  | learning\_rate\_init: 0.01 |
|  |  | alpha: 0.0001 |
|  |  | hidden\_layer\_sizes: 100 |
| Logistic Regression | 68.53% | max\_iter: 2000 |
|  |  | multi\_class: multinomial |
|  |  | penalty: l2 |
|  |  | random\_state: 10 |
|  |  | solver: newton-cg |
| Adaboosting | 77.50% | algorithm: SAMME.R |
|  |  | learning\_rate: 0.1 |
|  |  | n\_estimators: 125 |
|  |  | base\_estimator:DecisionTreeClassifier(criterion='gini', max\_depth=5, min\_samples\_leaf=1, min\_samples\_split=2,  splitter='best')) |
| Random Forest | 82.35 | n\_jobs: -1 |
|  |  | max\_features: sqrt |
|  |  | oob\_score: True |
|  |  | n\_estimators: 500 |
|  |  | max\_depth: 30 |
|  |  | min\_samples\_leaf: 2 |

# **ROC Curves**

In this section, receiver operating characteristic (ROC) curve is plotted for all the algorithms that we have implemented. Please note that, in the following ROC curves **class 0** represents waterpoints which are **functional**, **class 1** represents waterpoints which **need repair** and **class 2** represents waterpoints which are **non-functional**.

**Support Vector Machine**

**A screenshot of a cell phone

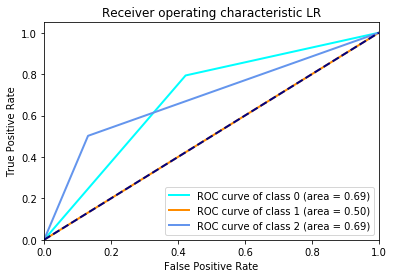
Description generated with very high confidence**

**Neural Net**

**A close up of text on a black background

Description generated with high confidence**

**Logistic Regression**

****

**Adaboosting**

**A screenshot of a cell phone

Description generated with very high confidence**

**Random Forest**

**A screenshot of a computer

Description generated with high confidence**

# **Challenges/Issues**

* Pre-processing the dataset
* Dealing with missing data for latitude, longitude and gps\_height
* Parameter Tuning to get better accuracy
* Plotting the ROC curve

# **Conclusion**

The project helped us to understand that feature engineering and data pre-processing are important steps before model is trained.

We created models using algorithms such as Support Vector Machine, Neural Network, Logistic Regression, Adaboosting, Random Forest. Among five models, models trained with techniques - Random Forest and Adaboosting have given comparatively better accuracies - 82.35% and 77.50% respectively. We performed logistic regression with multi\_class: multinomial as one of the parameters with accuracy 68%.

Surprisingly, support vector machine has given comparatively least accuracy – 56.16% with kernels – sigmoid, rbf. SVM performed poor with kernel – poly as we got accuracy of the model just 37%.

Neural Net was expected to perform good on classification but we suspect that our dataset was not large enough to train the model so after trying different combination of hidden layers we achieved 66.67%.

In AdaBoost, we combined many Decision tree classifier to produce a best classifier. In RandomForest, we combine Bootstrap sample gave moderate les with Decision tree classifier where we select random attributes for each tree model.

Thus, we can conclude that, ensemble methods performed better because in this we combine multiple classifiers and combine their outputs to get best result.

# **References**

[1] <http://allafrica.com/stories/201605240460.html>

[2] <https://mphdegree.arizona.edu/resources/articles/five-countries-with-the-greatest-water-scarcity-issues/>

[3] <http://geocoder.readthedocs.io/>

[4] <https://www.drivendata.org/competitions/7/pump-it-up-data-mining-the-water-table/>

[5] <https://www.drivendata.org/competitions/7/pump-it-up-data-mining-the-water-table/data/>