This is the handin service for **UK Students**.

COMP6208 - Advanced Machine Learning

Lecturer: <>
Assignment: Group Project Report
Student: SU, Hang (hs1a18)
Student ID: 30005019

Receipt of paper copy not yet acknowledged.

```
Submission Log

--------------


              Name: Hang Su

           User id: hs1a18

              Date: Tue May  7 19:16 2019

All their own work: yes

    Submitted From: 10.14.153.51

      Logged on as: hs1a18

       Web browser: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_4) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/74.0.3729.131 Safari/537.36


report.pdf: 927223 bytes from report.pdf, MD5 checksum 3fb711618eadc74e6b41bf7158ecd9ef


Student's Plagiarism Statement

------------------------------

This is all my group's own work and all sources have been correctly acknowledged.


Student's Comments

------------------

None
```

AML Project

Hang Su, Jing Meng, Zhihan Wang, and Xiaochuan Deng

University of Southampton,
hs1a18@soton.ac.uk, jm4y17@soton.ac.uk,
zw3u18@soton.ac.uk, xd1u18@soton.ac.uk

April 2019

1 Introduction

This group project report provides answers for the assignment from *Southampton COMP6208 Advanced Machine Learning 2018/19, Group Project Report*. In this assignment, we chose "Pump it Up: Data Mining the Water Table" from drivendata.org[1] as our project and our main target is to predict the status of water pumps according to the previous data. In this paper, we will talk about the methods we used to explore and clean the data, and then discuss the algorithms used in our project and their corresponding results.

2 Data Exploration & Cleaning

In "Pump it Up: Data Mining the Water Table" competition, data from Taarifa and the Tanzanian Ministry of Water is provided and our group need to use this data to predict whether a pump is functional, need repair or not working. In this project, we have a chance to construct a good predictor can improve the maintenance operations through the pumps across Tanzania. From the dataset, we get the data from many variables such as the type of pump, the time of pump installment, and the manage information of pumps. However, the data need to be cleaned because there are many missing values and some columns have a large number of values. In this section, we will talk about the methods we used to clean the data.

The topic says we need to use the given data set to classify the pumps into functional, unfunctional and needs to be repaired these three categories, thus this topic is a triple classification problem. Before deciding which features have the most predictive power and which algorithms are more suitable in this topic, we need to explore the data and do some data cleaning first.

There are two data-sets, a training set and a test set, so we need to explore them parallel. The training set has 59400 rows and 40 columns and the test set have 14850 rows and 40 columns. After printing the labels of the columns, it is obvious that all the features of them are the same. However, there are multiple data types among them, which means we have to explore and clean the data column by column.

First of all, we found that there are lots of null values in "funder", "installer", "subvillage", "public_meeting", "scheme_management", "scheme_name", "permit". Moreover, some of the features represent the same meaning, thus removing the redundant columns becomes the first task in data cleaning. As for the geographical distribution of the pumps, we plot a data point map to gain the general information according to the longitude and latitude, the result is shown in Figure 1. Moreover, we explored distributions of some other features for status, the results are shown in Figure .

The data type of "Data Recorded" is timestamp, but the meaningful data type should be time range, thus we found the nearest time and then transform the data into time delta type and restore them in a new column called "days_since_recorded". As for the features that contain lots of categories, we just

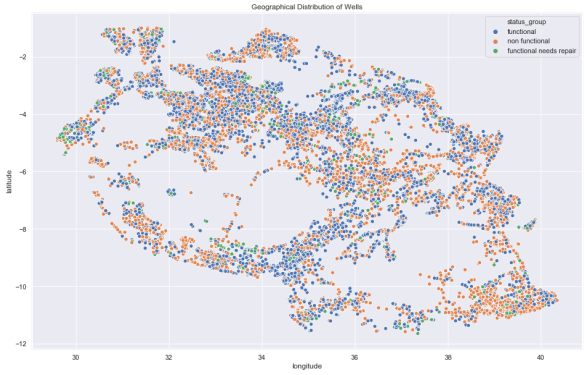


Figure 1: Geographical information from dataset

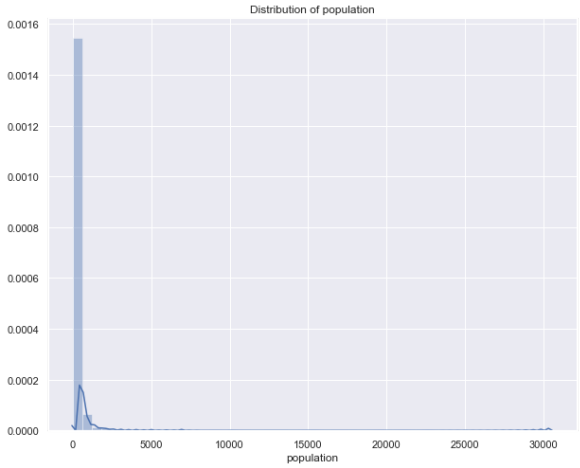
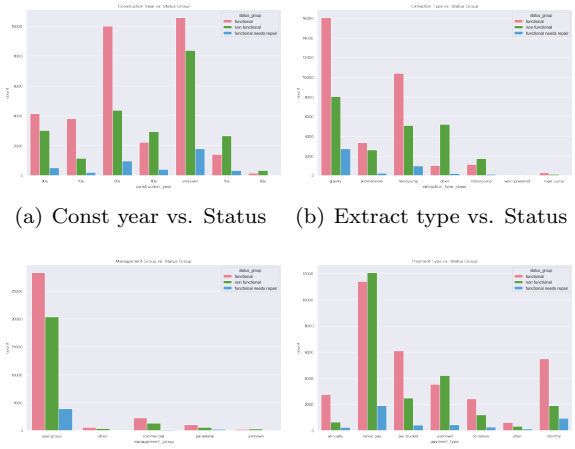


Figure 3: Distribution of population



(a) Const year vs. Status (b) Extract type vs. Status

(c) Manage group vs. Status (d) Payment type

Figure 2: Distribution

kept the top 5 most categories to reduce the burden of computation.

Furthermore, we also found that the distributions of "population" and "amount_tsh" are very skewed, as shown in Figure 2 and Figure 3, which means these two features need to be transformed in following stages. Also, we removed all the features that we thought have no significant predictive power. Finally, we check the correlation coefficient between each two of the features to make sure there is no close relationship between them.

Then, because the test set has almost the same features like the training set, we implemented the same operations on the test set. In the end, for the convenient of using the model, we transformed the final dataset into a one-hot matrix.

According to the given dataset, most of the variables have specific meanings and contribute to predicting the status of pumps. However, for some columns, we do not know their implication and some columns are redundant or have a big number of missing values. As a result, we dropped 18 columns and retained 22 variables for prediction. We also filled the missing values with "unknown".

For some columns that we thought were important, additional processing had been added. The "construction_year" column contains the year when a pump was constructed, and we categorized the values into seven values("60s","70s","80s","90s","00s","10s","unknown") because the original values were from 1960 to 2010.

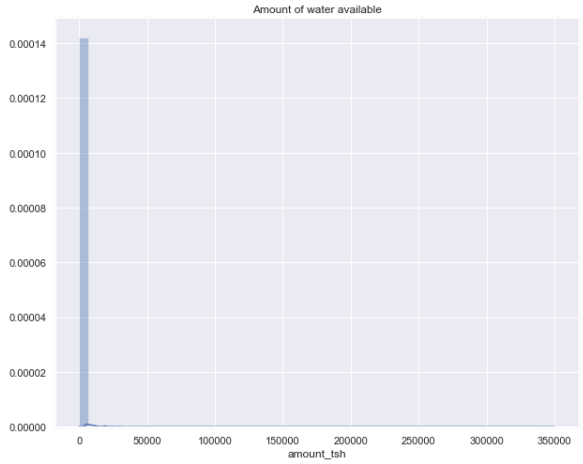


Figure 4: Amount of water available

We also consider that the funders and installers for pumps would have impact on the status of pumps. We found that there are so many values in "funder" and "installer" columns, so we also categorized the values of these variables. According to the speculation of values, we categorized the values into "government","community","individual","other" by using the regularization in Python. Besides, we also retained the values with more than 500 in occurrence number.

Since there are so many missing values in the raw data, after scanning the detailed information, firstly fill some missing parts with the value of neighborhood and others with zero or NaN. Secondly, for some repeated messages, directly deleted.

For example, 'longitude' and 'latitude' are able to completely represent location; therefore, only keep these two parts and drop other location-related features. In addition, it is good to reconstruct some information as redefined data are more suitable to our problem. For example, we combined 'existing year' with 'detecting time' and generated a new feature which represents the time period from existence to detecting time and much more persuadable.

Then for some 'object' type features, such as 'installer', or 'funder', as the number of different 'object' is too large to well separate into few clusters, we redesigned a new classifier with fewer important labels instead of original complex names.

Finally, we use the one-hot method to reconstruct the columns with category values and then merged the variables and labels into a new csv document.

3 Algorithms and Results

In this section, we will talk about the algorithms we used to predict the status of water pumps according to the previous data. We tried 5 methods in the classification, including: neural network, support vector machine, Gradient boosting, random forest, and XGBoost. In the prediction, we found that the algorithms based on decision tree method can reach higher performance compared to other algorithms in this task.

3.1 Neural Network

The first algorithm we implemented is neural network. Firstly, we spilt the train data into two parts, the training dataset and the validation dataset. Then we implement a neural network model into the training dataset. The neural network model used here is from *scikit-learn* [2] library with some default values in Python. To perform better, we used grid search to find much better parameters. According to this, the best optimizer is 'lbfgs' , a quasi-Newton method, different from Adam and SGD optimizer. The parameters of the model with the best performance is as below:

- The number of neurons of the first layer: 50
- L2 penalty term: 0.00001
- The number of iterations: 150
- The activation function for the hidden layer: logistic

Besides, to well iterate, we set up 'warm_start' as True, and keep the previous solution as initial. The good point of this model is saving time. Compared with other algorithms like random forest, neural network requires less time. However, the bad point is the problem of accuracy score, only around 0.6 in the validation dataset even though using the parameters performing best in the grid search.

3.2 Support Vector Machine

The second algorithm we implemented is the support vector machine. First of all, we split 20% of

the training data as the validation set. Then we used the grid search to fit the training set and try to find the best learning rate. After this, applying the most suitable learning rate and the model on the test set to get the final submission. All the tools we used are from the scikit-learn library. The best learning rate of this model is 0.1 and the final score of the result is around 73%. Even though the accuracy of SVM is lower than random forest and XGBoost, it still can be said a not bad model in this question.

3.3 Gradient Boosting

We also used ensemble algorithms such as Gradient boosting, Random Forest, and XGBoost to train our model. We compared the performance and execution time in the same computer for these algorithms. All these algorithms packages and libraries are from scikit-learn in Python.

First of all, we use 'train_test_split' function from the 'cross_validation' package to split the training data set to training and validation sets with a percentage of 90% and 10% respectively.

Gradient boosting is an ensemble method of combining weak classifiers such as decision trees. For Gradient boosting, we set the learning rate as 0.075, the number of estimators is 100, The minimum number of samples required to be at a leaf node is 16, the proportion for the max features of all features is 10% leading to a reduction of variance and an increase in bias. The validation accuracy score is 81.04% and testing accuracy score is 80.72%, but the Gradient boosting is quite a time consuming with the execution time is 894.37s.

3.4 Random Forest

Random Forests is an ensemble learning method which is implemented by constructing many decision trees and choosing the final class label as the mode of the class labels. We consider to use the original and normalised cleaned data for training. We set 'n_estimators=1000, random_state=3134', in terms of 'min_samples_split' and 'max_features'. We use a range of number [2,5,10,20] for 'min_samples_split' and ['auto', 5,10,20] for 'max_features' to grid search to find the best parameters by the cross-validation method. The results of the grid search for un-normalised and normalised cleaned data are in Table 1 and Table 2.

Min Samples Split	Max Features	Mean of Score	Std. of score
2	auto	0.7946129	0.0014548
2	5	0.7931539	0.0021282
2	10	0.7946129	0.0014548
2	20	0.7954547	0.0026109
5	Auto	0.7979799	0.002233
5	5	0.7984289	0.0025979
5	10	0.7979799	0.002233
5	20	0.7985037	0.0033398
10	Auto	0.7976994	0.0035315
10	5	0.7961468	0.0038145
10	10	0.7976994	0.0035315
10	20	0.7981857	0.0034314
20	Auto	0.7919941	0.0025695
20	5	0.7898804	0.0026181
20	10	0.7919941	0.0025695
20	20	0.7941452	0.0030475

Table 1: The results of grid search for normalised data

From the results, we can find 'min_samples_split=5, max_features=20' achieves the highest score and std. is small, the difference between normalised data and un-normalised data is quite slight because the most parameters are one-hot style as mentioned before. We choose 'min_samples_split=5, max_features=20' for un-normalised data to train the model, the validation score for this model is 81.58%, the execution time is 136.0794s which depends on the status of the computer as well. Then, we plot the confusion matrix figure for the true labels and the predicted labels. The figures for normalised and un-normalised confusion matrix are shown in Figure 4,5.

From the result, we can find the well of 'functional needs repair' does not have a good prediction. Most of the real ones end up in 'functional'. Finally, we fit the model in the test set and obtain a score of 81.08% in accuracy.

3.5 XGBoost

XGBoost is an open source library [3] which provides a Gradient boosting framework. For XG-

Min Samples Split	Max Features	Mean of Score	Std. of score
2	5	0.7980361	0.002465
2	10	0.7978303	0.0025093
2	20	0.7981857	0.003114
5	5	0.7959972	0.0037087
5	10	0.7979052	0.0035229
5	20	0.7982979	0.0028764
10	5	0.7895063	0.0027442
10	10	0.7920502	0.002554
10	20	0.7939769	0.0031733

Table 2: The results of grid search for un-normalised data

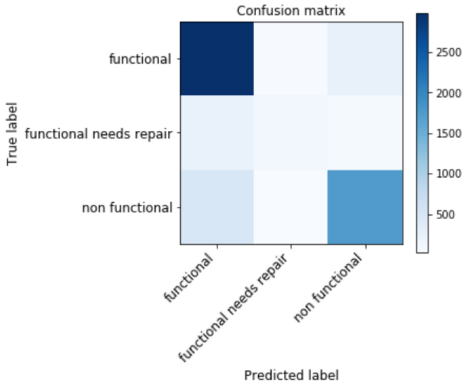


Figure 5: The un-normalised confusion matrix

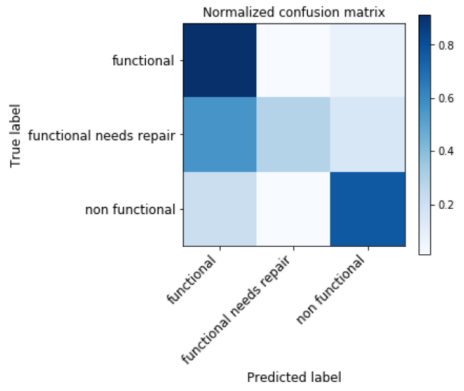


Figure 6: The normalised confusion matrix

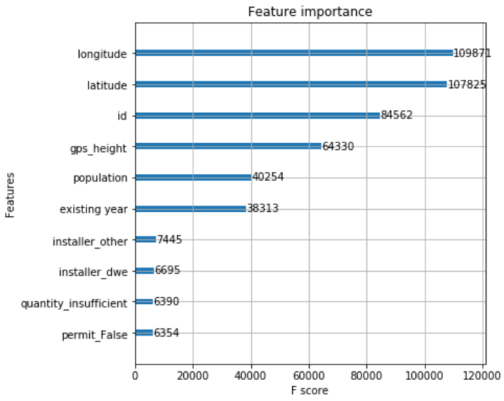


Figure 7: Feature importance result from XGBoost

Boost, we set 'multi:softmax' for the 'objective' parameter, the learning rate is 0.01, the max depth of trees is 12, the L2 regularization factor is 2, the factor of controlling whether to prune is 0.1, the percentage of random subsample is 0.7, the percentage of column sampling when generating tree is 0.5.

Because the XGBoost supports paralleling computing for gain calculation of each feature when determining the optimal split point, we set ''nthread': 4' for multi-thread computation. The execution time is 351.03s for 500 rounds. We get the score plot of the most ten important features influencing the result as follows, which shows the geographical information longitude and latitude are the most important features. The validation accuracy score of the XGBoost model is 80.94%, and the testing score is 80.22%. The feature importance result from XGBoost is shown in Figure 6. We can find out that the geographical data is still very important in prediction.

Overall, comparing the three ensemble methods, the accuracy is similar via parameters tuning, however, in terms of execution time, the Random Forest has the best performance. Hence, we choose the Random Forest as a solution.

4 Conclusion

In "Pump it Up: Data Mining the Water Table", our group tried to predict whether a pump functional, need to be repaired or not working. We explored the dataset to find important variables

in the dataset and found out that some columns have too much missing data. Besides, there are some columns represent the same meaning. We also found some variables were important but skew in predicting the status of a pump such as the "population" and "amount_tsh". We transformed the data from these columns.

Then we cleaned the dataset such as dropping some useless columns, transferring data types and fulfilling the missing data. In data cleaning, we also tried to categorised values from some variables such as the funder and producer of a pumps because they are important variables for our prediction but have too many different values. Finally we retained only 22 variables from 40 variables for prediction.

In the prediction, we tried neural network, support vector machine (SVM) and 3 algorithms based on the decision tree. Though SVM can reach about 70% accuracy in the classification, we found that tree methods can get even higher performance. Gradient boosting reached 80.72% in accuracy, while the XGBoost get 80.22%. In our project, the random forest is the best classifier, reaching 81.08% in accuracy for the test set.

Our best classifier's performance(81.08%) is close to the best classifier's accuracy(82.90%) on this task in the world, according to the recent data from drivendata.org. We are happy to see this result because it is the first for most of us in the group to join in the competition. In the future, we might improve our classifier by doing more in cleaning the dataset and introducing different datasets that can help us to improve the performance in prediction.

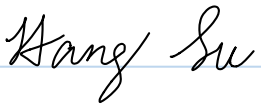
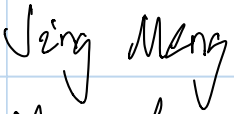
References

[1] Pump it up: Data mining the water table. https://www.drivendata.org/competitions/7/pump-it-up-data-mining-the-water-table/. Accessed April 13, 2019.

[2] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.

[3] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794. ACM, 2016.

Mark Distribution

Member Id	Member name	Mark	Sign
hs1a18@soton.ac.uk	Hang Su	25%	
zw3u18@soton.ac.uk	Zhihan Wang	25%	
jm4y17@soton.ac.uk	Jing Meng	25%	
xd1u18@soton.ac.uk	Xiaochuan Deng	25%