MACHINE LEARNING APPROACH FOR THE PREDICTION OF THE STATUS OF TANZANIAN WELLS [COMP4030 CW2 - Data Science and Machine Learning]

*Thomas Cotter
Computer Science
University of Nottingham
Nottingham, England
psytc8@nottingham.ac.uk

*Loo Yang Shen Jason
Computer Science
University of Nottingham
Nottingham, England
hfyyl5@nottingham.ac.uk

***Abstract*—This paper details our approaches and results for the COMP4030 CW2. The goal was to predict the status ('Functional', 'Functional - Needs Repair' & 'Non-Functional') of water wells in Tanzania. This is important as not only does knowing the status of the well help keep the total percentage of functioning wells higher, but it also allows for more effective spending on repairs. The most important features included 'age', 'season_when_recorded', 'height_above_sealevel' & 'extraction_type'. The best performing models were XGBoost and BaggingClassifier.**

***Index Terms*—Machine Learning, Data Science, Classification, Water Pumps**

I. Introduction

A. Research Questions

What factors are most important for determining the status of a well, and how accurately can we classify wells based on these features?.

We choose this question because we are interested in the factors that determine the status of a well, and using ML to try to classify these wells. Follow-up questions to this question could include:

- How does the accuracy of the classification model vary with different feature sets and classification algorithms?
- Could we use our results to ensure that wells are built and repaired so that fewer wells are non-functional?

B. Dataset

The chosen dataset is from the Tanzanian Ministry of Water and contains information on the status of wells in Tanzania. This dataset has 59400 rows, with 40 different features. These 40 features could be broken down into three subgroups: a) Geographic Location of the Wells. b) Management of the wells. and c) Water Condition of the wells. The dataset is originally split into 2 different files, one for labels and one for the actual data. These can be merged easily with pandas through left join on the "ID" column.

Fig. 1 shows the distribution of the target variable, which is the status of the well. We can see from this that we might need to oversample the 'functional needs repair' class, due to class imbalance.

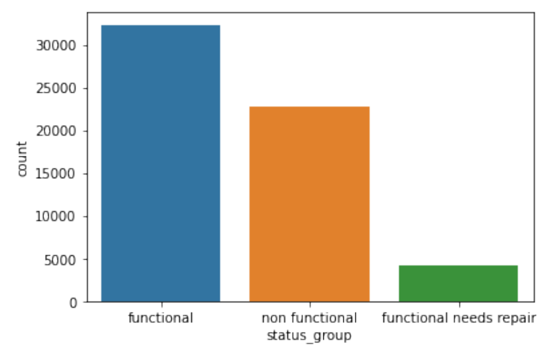


Fig. 1: Distribution of the target variable

C. Management Structure

A requirement of this project was to work separately within our pairs and try different techniques to solve the problem. To not lose important insights, we communicated our results throughout the project. We followed a 'Christmas Tree'-like approach for each stage: 'Data Analysis', 'Data Pre-processing' & 'Data Classification'. Each section was performed individually, before combining our results and moving out the next section. This process wasa performed cyclically, such as re-visting the Preprocessing step after the first Classificiation step. See Fig. 2

This results in two main benefits. Firstly, it allows us to research a large number of approaches, which provides a higher chance of finding a powerful approach. Secondly, the combination of approaches that resulted might have never happened when working individually, which may provide more optimal results.

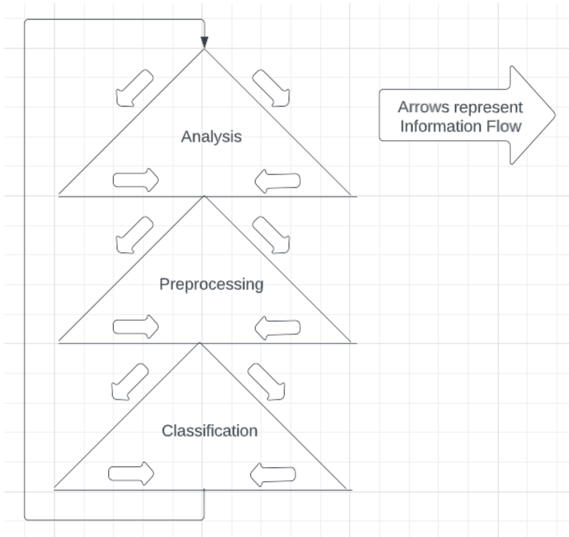


Fig. 2: Management Structure

II. Literature Review

One study by Pathak et al. (2023) [1] compared the performance of TabNet, a sequential attentive classification architecutre designed for tabular data, and tree-based approaches such as XBBoost. They found that TabNet outpeformed XGBoost, boasting an 83% accuracy compared to XGBoosts 78%. TabNet makes use of Transformers, a machine learning algorithm which uses self-attention to differentially weight the significance of each part of the input. A point of note is that TabNet does not require feature engineering to perform at these standards. We have not used TabNet in our report, as our primary goal was to showcase an end-to-end machine learning solution, and this includes feature enginering. However, it provides an interesting comparision.

Pham et al studied fully connected neural networks on the Pump it Up dataset [2], settling on a model with 7 hidden layers and cross-entropy loss as the loss function. Their trained model achieved a 78.6% accuracy rate on the test dataset. Although our project has prioritized tree-based methods for their speed, it would be interesting to compare our results with those of a neural network.

Jithin Paul also undertook a study on the Pump it Up dataset, experimenting with different models to test for the best results [3]. He proposed four methods, RandomForest, DeepLearning, LogisticRegression & AdaBoost. The results showed that RandomForest, was the most accurate model, achieving a mean accuracy of 81.18% on the test dataset. This finding demonstrates that tree-based models perform exceptionally well on this dataset, which motivated us to use more sophisticated tree-based methods in our own work.

III. Methodology

A. Data Analysis

When conducting data analysis, the python library Seaborn [4] was our primary data visualization tool. Seaborn offers more visually appealing graphs than MatPlotLib.

1) Thomas: Thomas visualized the correlation between 'construction_year'and other features in the dataset. 'construction_year' had a number of missing values, therefore these graphs could determine the possibility of using other features to impute the missing values.

2) Jason: Jason checked the number of missing and unique values across 38 features provided. The main reasoning behind it was to identify features that would require imputations due to missing or invalid values. In addition, this step allowed him to distinguish features which would have to be recategorised due to the large number of unique values present. He also generated Count Plots for each of the previously mentioned features based on water wells' status. These plots allowed him to visualise the distribution of the wells' status based on features. Thus, making it easier to identify features that could possibly influence the status of a water well.

B. Data Preprocessing

1) Thomas: Thomas performed imputation of the lat & lon missing values by applying grouped mean imputation. The data was grouped by 'ward', 'lga' and then 'region'. This ensured that any missing values in either 'ward' or 'lga' did not result in missing lat / lon values. This also provides more accurate results than just 'regular' mean imputation, as the imputed values are localised. The Funder & Installer columns contained 1000s of unique values. In Thomas' approach, he decided to group the column into categories: 'Charity', 'Government', 'Local Government', 'Private', 'Religious', 'Foreign', 'School' & 'Unknown'. This was done to simplify the dataset, making it easier for the model to learn the distribution. Since, the columns had crossover between the values, this can be done in 1 step. Thomas also used the SMOTE-ENN algorithm from imblearn [5], to balanced the dataset, specificall the 'Functional - Needs Repair' class. Imbalanced datasets could potentially affect model performance, especially in tree-based models. Finally, 'gps-height' was normalised using a custom MinMaxScaler and a custom Z-Score Scaler. This was due to the prevelence of outliers in the 'gps-height' feature.

2) Jason: Due to the presence of missing data in the Latitude & Longitude columns, Jason decided to apply mean imputation based on the Region column. The values of the Region column corresponds with the name of cities present in Tanzania. By calculating the mean latitude and longitude value of each region, a rough estimation of the geographical location of wells with missing value could be known. Therefore, allowing valuable insights to be learnt by ML models during training.

During the Data Analysis Phase, he had noticed that two columns Permit and Public Meeting had missing values. Upon further inspection, these two columns only contain the values of either 'True' or 'False'. Data Imputation was not done at this stage due to the lack of valuable data present in the data set. To preserve the distribution of these two columns, a unique values known as 'Unknown' was given to rows with missing data.

Jason had performed data recategorisation on both funder and installer columns which contained 1000 - 2000 unique values. He had first performed data cleaning to fix spelling mistakes that were present in each column. This was followed by identifying unique values with count above the threshold of 1000. Each unique value above the previously mentioned threshold would be classified as its own individual class while those below it would be classified under the 'Other' value. Data recategorisation would be able to reduce the number of unique values present in these two columns leading to better ML model training performance.

Although the number of unique values in the Region Column are minimal, Jason decided to reclassify them according to their geographical zones based on data that was provided by the Tanzania Water and Sanitation Network [6]. This reduced the number of unique values of the Region Column to 7 from 21. New insights could potentially be drawn upon based on the zone that the water wells is currently located in.

The data set provided had feature and target variables in the form of categorical form. Jason had converted these features into numerical form that is required of by ML models prior to training. He had encoded the values of the feature and target variables in the form of alphabetical order whereby a = 0, b = 1 etc.

C. *Data Classification*

Initially, we focused on testing the performance of different algorithms without any feature selection. This allows us to focus on the best performing ones. See Table. I. We can see that XGBoostClassifier, CatBoostClassifier, BaggingClassifier & HistGradientBoostingClassifier were the most effective for our problem. We decided to explore these models using feature selection & hyper-parameter tuning. Our feature selection process employed a Chi-Squared test and feature importances obtained from a RandomForest and a XGBoost.

TABLE I: Initial Classification Results

Algorithm	Accuracy	Precision	Recall
XGB	**0.799**	**0.751**	0.633
CatBoost	0.795	0.746	0.634
Bagging	0.793	0.704	**0.659**
HistGradientBoosting	0.791	0.743	0.624
DT	0.757	0.643	0.645
KNN	0.709	0.623	0.563

1) Thomas: Thomas trained the 4 top models (XGBoost, CatBoost, Bagging & HGBoost) on the SMOTE balanced dataset and compared the results to models trained on the imbalanced dataset. The results were also compared when optimizing for 'recall_macro' or 'accuracy'. Recall is usually a better metric for imbalanced datasets, as accuracy can often misrepresent the truth. 'recall_macro' takes an average of the recall for all 3 classes, which makes it a good metric to use.

Furthermore, he used Weights and Biases (WandB) [7] to analyse the subset of features produced by the Chi-Squared Test and RandomForest. WandB served as an accesible MLOps platform for experiment tracking. Multiple classifiers were trained on different feature subsets (ranging from sizes 0.75N to N, where N is the size of the original feature set), and tracked this information in WandB. WandB allows us to quickly filter for the best performing 'runs', and the feature subset used. He also used WandB to further analyse the results of his cross-validation hyper-parameter tuning, in much a similar process to feature selection.

2) Jason: From the Random Forest feature importance and Chi-Squared Test, Jason had shortlisted 16 features that influenced the classification of Tanzanian Water Wells. Prior to model training, the dataset was subjected to a 80%:20% train test split. This was followed by normalising the testing and training set features with z-score normalisation.

For this study, he considered three different machine learning models (Random Forrest, XGBoost & CatBoost). Hyper-parameter tuning was conducted via GridSearchCV [8] from the sklearn library. Upon completion of the hyperparameter tuning process, the most optimal parameters of each model were recorded before being passed for model training.

To evaluate the performance of each model, Jason used three different evaluation methods. K-fold cross validation with the value of K being 5 , confusion matrix and receiver operating characteristic (ROC) curve. K-fold cross validations was used to evaluate the performance of the model on unseen data. Meanwhile, confusion matrix showcases the predictions which are labelled correctly and incorrectly by class. Although ROC Curve is usually meant for binary classification, it could still be extended to multiclass classification through pairwise comparison i.e., one class vs all other classes.

IV. RESULTS

A. *Data Analysis*

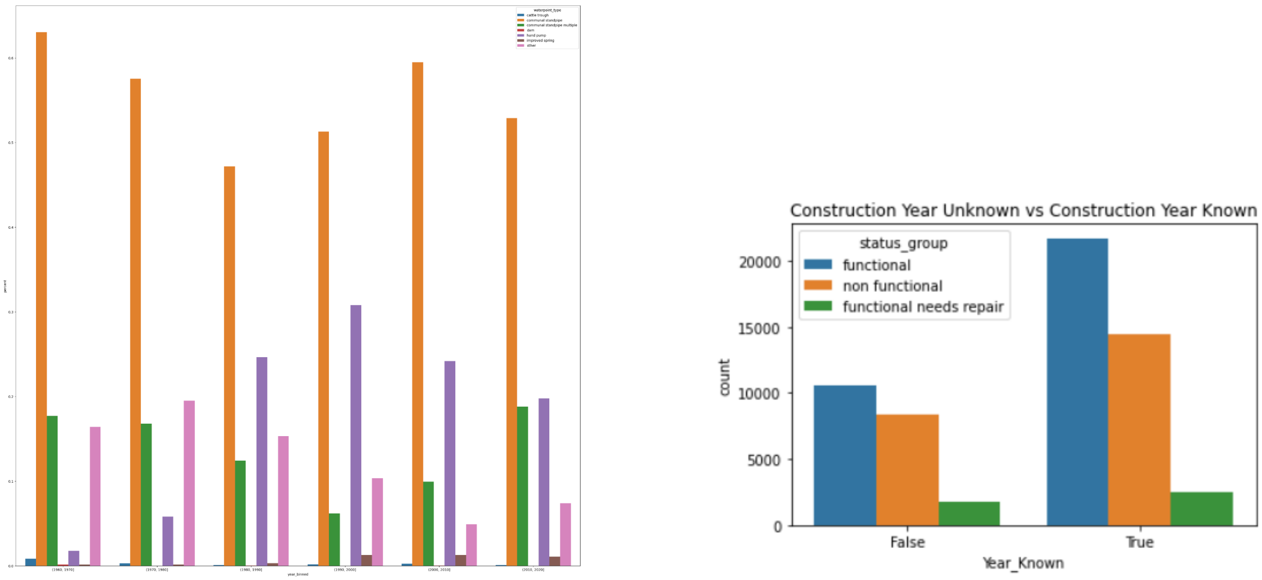
1) Thomas: An example of the construction year visulisations can be see in Fig. 3. From this, we can see that there is a slight variation in Waterpoint Type usage across decades, but nothing of significance. The same is true across the other visualisations produced (see the accompanying ipynb for more details). For this reason, the missing values in the Construction year were kept as 'Unknown'. The distribution of the status group values in 'Unknown' or 'Not Unknown' construction year values can be seen in Fig 3.

B. *Data Preprocessing*

1) Thomas: Firstly, The results of the latitude and longitude can be seen in Fig. 4.

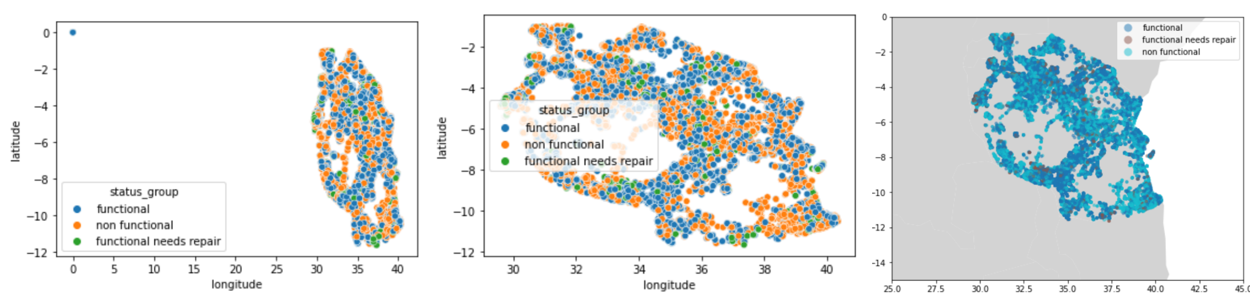
The installer & funder columns were categorized manually using Thomas' own knowledge and online research. The result was 8 categories, the counts for these are presented in Table II. These versions of the features showed some benefit to the model when testing feature subsets in WandB

Thomas used the SMOTE-ENN algorithm to oversample the minority class. The results of this can be seen in Fig. 5. On the x-axis, O is 'Functional', '1' is 'Functional - Needs Repair' and 2 is 'Non-Functional'.



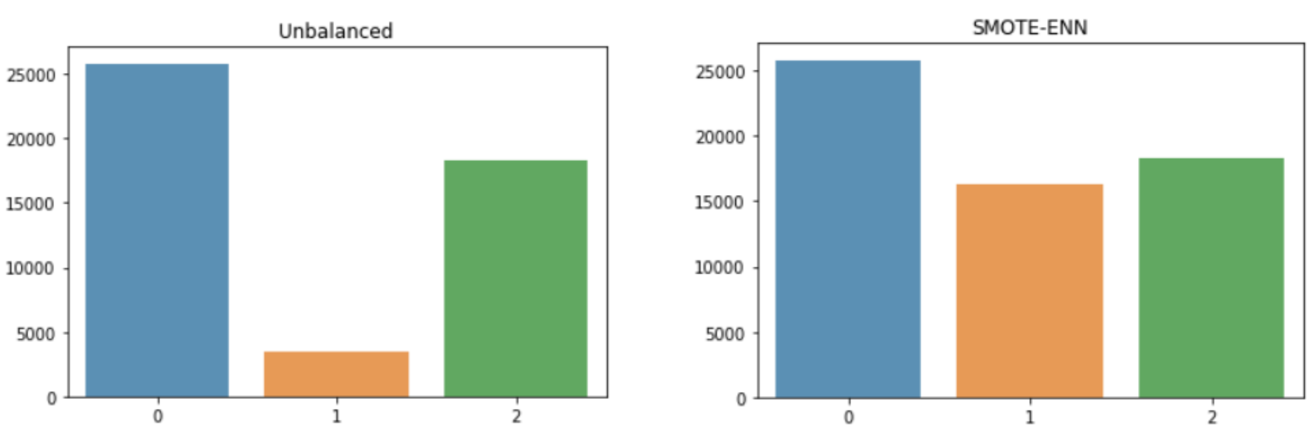
(a) Waterpoint Type vs Construction Year (b) Construction Year Unknown vs Construction Year Known

Fig. 3: Construction Year Data Analysis



(a) Pre-Imputation Latitude & Longitude (b) Post-Imputation Latitude & Longitude (c) Imputated Latitude & Longitude Overlayed onto a map of Tanzania

Fig. 4: Latitude & Longitude Imputation



(a) Pre-Oversampling Class Distribution (b) Post-Oversampling Class Distribution

Fig. 5: SMOTE-ENN Oversampling

C. *Classification*

1) Thomas: Post hyper-parameter tuning,accuracy, precision ('macro'), recall ('macro') and f1-score were calculated

TABLE II: Funder & Installer Categories

Column	Category	Count
Funder	Government	20199
	Charity	11066
	Unknown	8216
	Foreign Aid	8131
	Religious	4087
	Private	3889
	Local Government	3774
	School	38
Installer	Local Government	22515
	Government	10327
	Unknown	8800
	Charity	7487
	Private	3853
	Foreign Aid	3346
	Religious	3046
	School	26

for each of the models. The models were evaluated on the imbalanced and SMOTE-ENN balanced dataset. The results can be seen in Table III

The results show the BaggingClassifiers performs the best, although the difference is minimal. The balanced dataset provides a slight increase in F1-Score, which accuracy is higher with the imbalanced dataset. This makes sense, as incorrectly classifying the minority class has a small impact on accuracy but a large impact on F1-Score.

Thomas produced the two confusion matrices using an XGBoost, see Fig. 6. They show the output of training on both the balanced and imbalanced datasets. Balancing the dataset results in less False Positives on the minority class. If the goal of your model was to do specifically this, perhaps using the balanced dataset would be the best idea. This was not the case for us, so we continued with using the imbalanced dataset.

Since the difference between different models and datasets is neglible, Thomas decided to combine all the models to create a VotingClassifier to possibly squeeze out some extra performance. These results can be seen in Table IV

TABLE IV: Voting Classifier Results

Accuracy	Precision	Recall	F1-Score
0.810	0.759	0.650	0.679

Using both the results from model based feature selection, and feature selection after looking at the results from Weights & Biases, Thomas decided to use the following features in his final model: **'age'**, **'latitude_imputation'**, **'longitude_imputation'**, **'construction_decade'**, **'quality_group'**, **'basin'**, **'extraction_type'**, **'cat_installer'**, **'population'**, **'gps_height_zscorenormalise'**, **'cat_funder'**, **'quantity'**, **'consistent_water'**, **'source_class'**, **'zones'**, **'waterpoint_type'**, **'season'**, **'extraction_type_class'**, **'payment'** (Please consult the ipynb for more details on what information these features contain).

To relate this back to our research questions, we can say that these features are the most important when looking at whether wells are functional or not.We can also say we can classify whether a well is function or not with 81% accuracy.

V. Discussion

In this section, we discuss the results of our analysis. We compare each of our separate approaches to each other, as well as comparing them to the performance of models in the literature.

A. Pre-Processing

1) Tom: Jason and I used many different approaches throughout the project. For instance, Jason utilized a StandardScaler to scale every feature by removing the mean and scaling to a unit variance. I did not scale any features except 'gps_height', and even then I used a MinMaxScaler. Despite the presence of categorical data, Jason's method still produced high quality results. Another difference in pre-processing was the handling of the funder and installer columns. We both took different approaches to this, and in my opinion Jason's technique proved slightly better. By categorizing the entire dataset, valuable information was lost by not using the specific funders or installers. Jason's approached lost less information as only values with small counts were removed from the dataset and classified as 'Other'.

B. Modelling

1) Tom: We also used different hyper-parameter tuning methods, specifically GridSearch (Jason) and RandomSearch (Tom). RandomSearch enabled a wider exploration of hyperparameter ranges across various models, making it faster. On the other hand, GridSearch focused on specific potential hyperparameters, providing a more in-depth search. I think this is reflected in our results, as Jason's XGBoost slightly outperformed my XGBoost on the test set, however the breadth of search allowed me to find the BaggingClassifier, which performed very well.

C. Literature Discussion

When comparing to the TabNet implementation by Pathak et al [1], we can see that our best classifiers were outperformed by TabNet by at least 2%. This is quite significant. It's understandable that this is the case due to the capabilities of Transformers, which are arguably the most popular model right now.

We also reviewed Pham et al [2] and found that their produced Neural Network (NN) was not as high performing as some more lightweight models. XGBoost / HistGradientBoost and BaggingClassifier all out-perform the NN. These lightweight models also are much faster to train than an NN, so it is much more efficient to use these models on a dataset such as Pump It Up.

VI. Conclusion

References

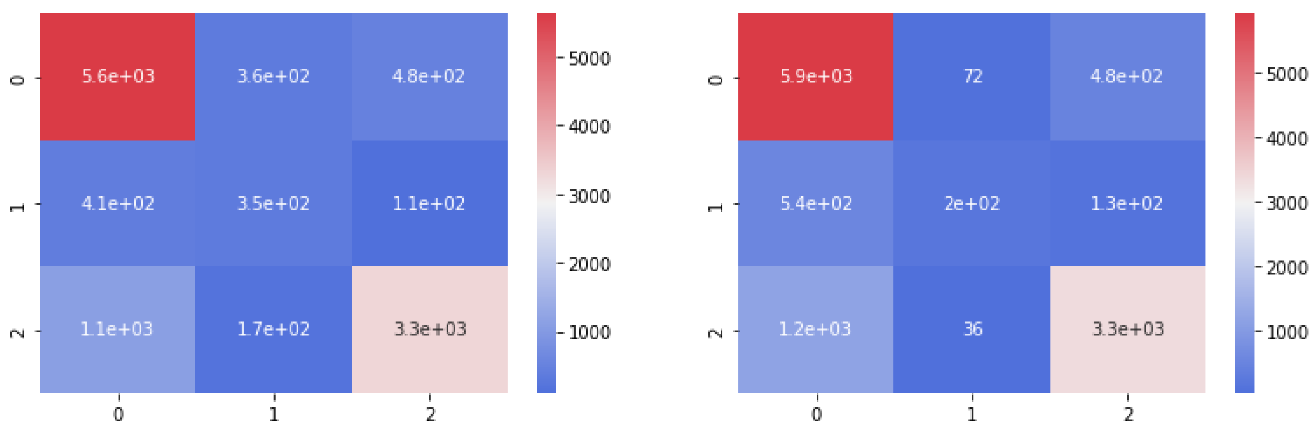
[1] K. Pathak and L. Shalini, "Pump it up: Predict water pump status using attentive tabular learning," 2023.

[2] A. Pham, B. Backus, and L. Zhu, "Pump it up: Mining the water table," *Stanford Machine Learning Projects 2018*, 2018.

[3] J. Paul, "Pump it up - data mining the water table," https://jitpaul.blog/2017/07/12/pump-it-up/, accessed: 2023-04-28.

TABLE III: Classification Results

Scoring	**Dataset**	**Model**	**Accuracy**	**Precision**	**Recall**	**F1-Score**
Recall Macro	Balanced	XGBoost	0.802	0.701	0.668	0.683
		CatBoost	0.783	0.669	0.660	0.664
		HistGradientBoosting	0.796	0.693	0.665	0.677
		BaggingClassifier	0.803	0.703	**0.676**	**0.687**
	Imbalanced	XGBoost	0.804	0.720	0.662	0.683
		CatBoost	0.791	0.697	0.652	0.668
		HistGradientBoosting	0.806	0.730	0.658	0.681
		BaggingClassifier	0.803	0.722	0.660	0.681
Accuracy	Balanced	XGBoost	0.800	0.700	0.662	0.677
		CatBoost	0.787	0.678	0.665	0.670
		HistGradientBoosting	0.797	0.699	0.666	0.680
		BaggingClassifier	0.801	0.701	0.671	0.684
	Imbalanced	XGBoost	0.804	**0.748**	0.648	0.676
		CatBoost	0.797	0.753	0.635	0.665
		HistGradientBoosting	0.806	0.733	0.662	0.686
		BaggingClassifier	**0.811**	0.741	0.656	0.681



(a) Balanced Dataset

(b) Imbalanced Dataset

Fig. 6: Balanced vs Imbalanced Confusion Matrices (XGBoost)

[4] M. Waskom, "Seaborn: statistical data visualization," https://seaborn.pydata.org/, accessed: 2023-04-24.

[5] G. Batista, R. C. Prati, and M. C. Monard, "a study of the behavior of several methods for balancing machine learning training data," *ACM Sigkdd Explorations Newsletter 6 (1), 20-29*, 2004.

[6] "Tanzania water and sanitation network," http://www.tawasanet.or.tz/index.php, accessed: 2023-04-25.

[7] "Weights and biases," https://wandb.ai/, accessed: 2023-04-24.

[8] "Gridsearchcv," https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html, accessed: 2023-04-25.