MACHINE LEARNING II: GROUP PROJECT

Group O-1-6
LOUIS DUBAERE
JUAN PABLO GARCÍA GÓMEZ
FRANZ-ANTON GRAF BASSELET VON LA ROSEE
SIYAO LU
ANDREW MARTINEZ
STEPHANIE NJERENGA
TOMAS TELLO GARCÍA

Table of Contents

Executive Summary

This report deals with the analysis of turnover of well functional status in the country of Tanzania with the aim of accurately predicting which wells are functional, non-functional, and still functioning but in need of repair. The primary model selected for this purpose will be a Random Forest utilizing stacked models (which will include Random Forest, Gradient Boosting Trees, and K Nearest Neighbors) in order to provide fine-tuned predictions on each category. With a rich set of data, various techniques with regards to data cleaning (e.g. imputing missing values, fixing invalid entries, etc.), feature engineering (e.g. custom feature creation, Genetic Programming, Polynomial Features, etc.), introduction of outside data, and hyperparameter tuning were attempted to achieve a maximum possible accuracy. This iterative testing process has yielded a current best accuracy of approx. 81% on the blind test set.

Data Exploration and Preparation

Numerical Values

Preliminary analysis reveals that there are a number of missing values among many of the variables (including construction year, population, longitude, latitude, among others), therefore imputation or variable dropping will be needed. With regards to construction year, it was found that imputing 0 (which the analysis has understood as a null value) with the overall mean was the most helpful path to take. As this methodology would be too crude for population as well as longitude, latitude, and GPS height (altitude) due to their geographic dimension, the mean with regards to the region and district for a given empty entry were imputed instead. Amount_tsh, id, and num_private were ultimately excluded as they were found to be not useful in the case of the latter two or had too many blanks that could be otherwise rectified in the case of the former. Before model training, all numerical values were standardized, fixed for skewness, and clipped for outliers

Categorical Values

A significant amount of time was spent contemplating how to handle the many rows and unique values within the categorical variables. It was first discovered that installer, wpt_name, date_recorded, lga, scheme_name, ward, funder, Name, and subvillage each had more than 50 unique values. In the case of date_recorded, this variable was parsed into Year, Month, Day, as well as Day of Week, with the original variable dropped. Other variables that were dropped from this list included wpt_name, scheme_name, ward, Name, and subvillage as they could be replaced by other variables in the case of the geographic related ones or were found to be irrelevant in the case of name related ones. Furthermore, payment, quality_group, quantity_group, source_type, extraction_group_type, and waterpoint_type_group were ultimately dropped as well due to the availability of similar variables that could provide better specificity.

LGA, funder, and installer were retained though the number of unique values were reduced. For LGA, entries that included the words “rural” or “urban” were categorized accordingly (as intuitively more urban regions might be better maintained due to a higher overall population) while any other value was labeled as “other”. Similar schemes were applied across the board to the other categorical variables whereby categories that represented less than 1.0% of the total data were grouped as “other”. Raising this threshold to a higher degree was found to reduce the unique values to a point that information was lost, and model performance suffered. A more fine-tuned approach was applied to extraction type such that a combination of lower recorded categories and those that explicitly contained the phrase “other” were bucketed together.

The variables permit and public meeting were found to have missing values which were imputed with False entries. Upon completion of these procedures, the remaining categorical variables were converted into dummy variables.

Introducing Outside Data

Relevant outside data was also included in this analysis. In particular data with regards to region area, population changes as well as average weather for each month of the year were found to be of some value. With these inclusions and the aforementioned variable removals, 184 variables were used for the baseline model.

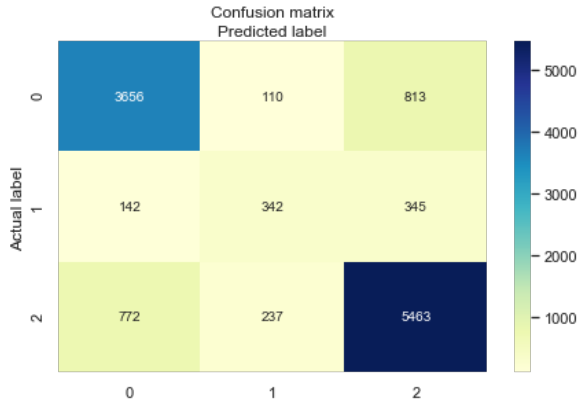
Baseline Model

A Random Forest model was ultimately chosen for this analysis due to its relatively strong performance (per the below cross validated scores) as well as lower time needed for execution (note: models like SVC were also tried but were found to be too computationally expensive to be practical).

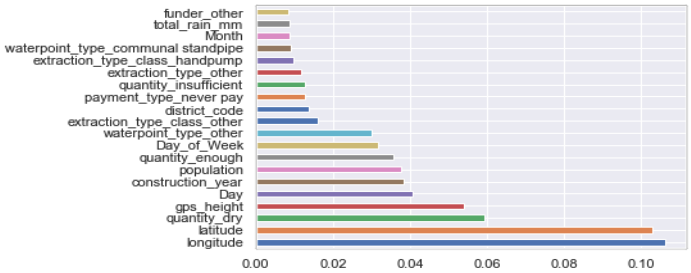
```
LR: 0.740522 (0.003617)
KNN: 0.772138 (0.004408)
NB: 0.440152 (0.018386)
SGD: 0.711347 (0.011296)
RFC: 0.788872 (0.003067)
GB: 0.757071 (0.004099)
XGB: 0.749141 (0.004469)
```

After applying cross validation, the Random Forest model presents the below deceptively high results:

```
BASE LINE:
Accuracy of Random Forest classifier on hold-out set: 0.796
Mean Accuracy after CV: 0.791 +/- 0.003
Best Accuracy after CV: 0.794

              precision    recall  f1-score   support

           0       0.80      0.80      0.80      4579
           1       0.50      0.41      0.45       829
           2       0.83      0.84      0.83      6472

   micro avg       0.80      0.80      0.80     11880
   macro avg       0.71      0.69      0.69     11880
weighted avg       0.79      0.80      0.79     11880
```



While this appears to be a very strong model, when tested against the blind test set this model is shown to overfit as it achieves an accuracy of approx. 75%. However, a benefit of this model is seeing which variables were ultimately the most important (something that will later be helpful). As it can be seen below, imputing latitude, longitude, GPS height, construction year, and population were critical due to their high importance in this model. In the context of Random Forest, variable importance is determined by a particular variable's gini importance (i.e. which takes into account the decrease in node impurity [a "pure node" would have a rule that would put all observations into a single bucket] taking into account the probability of reaching that node. Therefore, more important nodes would be reached with greater probability and more cleanly partitions observations.



Feature Engineering

Manual Feature Engineering

The two features that were manually created included the creation of a “distance from capital” and “operation years” variables that measure the distance from a given pump to the capital city Dodoma for the former and difference between the Year (extracted from the removed “date_recorded” variable) and Construction Year for the latter. Across different runs, distance proved to be consistently beneficial for the model and was retained, while operational years was eventually discarded.

```
Accuracy of Feature: 0.797
Mean Accuracy after CV: 0.790 +/- 0.002
Best Accuracy after CV: 0.793
```

```
Accuracy of Feature: 0.791
Mean Accuracy after CV: 0.790 +/- 0.004
Best Accuracy after CV: 0.793
```

Genetic Programming

Genetic programming is a technique through which random linear combinations of existing variables are created using a variety of operations (e.g. addition, subtraction, logarithms, cosine, etc.). By utilizing this technique, different combination of variables that the modeler might not otherwise think of, can be created in an automated fashion. Although in this instance it appears to slightly hurt the model, across different runs it was ultimately determined to be beneficial.

```
Accuracy of Feature: 0.793
Mean Accuracy after CV: 0.789 +/- 0.005
Best Accuracy after CV: 0.794
```

Polynomial Features

Another option explored for automated feature engineering was to make use of Polynomial Features, which generates new variables by creating a matrix of new features based on all polynomial combinations (i.e. given a pair of features a and b, features like a^n or b^n s.t. n is the number of degrees entered as an input). This methodology can help the final model create non-linear partitions between the different categories that would otherwise have been difficult to classify with linear partitions. However, it was found that this actually consistently decreased the model’s performance and therefore was excluded.

```
Accuracy of Feature: 0.786
Mean Accuracy after CV: 0.784 +/- 0.003
Best Accuracy after CV: 0.787
```

Feature Combination

After analyzing each of the aforementioned feature engineering techniques in a vacuum, different combinations were attempted to see which had a superior effect. Experimentation determined that combining genetic programming and the “distance from the capital” variable were the most beneficial on average.

```
Accuracy of Feature: 0.791
Mean Accuracy after CV: 0.790 +/- 0.002
Best Accuracy after CV: 0.793
```

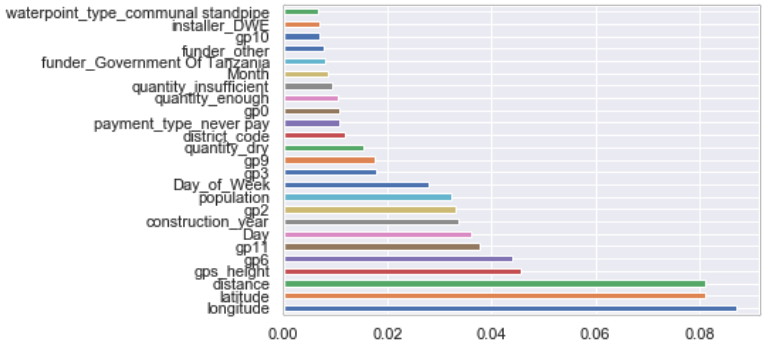
Feature Reduction

As the above techniques had created many additional variables (which significantly slowed model runtime and was found to affect the model’s ability to generalize in subsequent steps), it was necessary to reduce the total number of features to an extent that maintained a comparable accuracy.

Techniques such as Principal Component Analysis (PCA) and Recursive Feature Elimination with Cross Validation (RFECV) were attempted with inferior results. In the case of PCA, it was found that numerical variables could theoretically be consolidated to a single component that captured approximately 99% of the total variance, however this performed poorly in testing in many scenarios. Furthermore, RFECV was applied first to the Boolean variables in conjunction with PCA (however, this only reduced the number of Booleans by a minimal degree without much benefit) and later on the entire set of variables (this was done without PCA and was found to be too computationally expensive to be practical).

The settled-on technique was to take the n variables of highest importance according to the Random Forest Model. Experimentation found that approximately 120 of the top variables was sufficient to maintain (and in even some runs slightly improve) the model performance. Unfortunately, without the aforementioned ability to use RFECV, whether or not this number is optimal when making other changes to the model or ETL.

```
Accuracy of Feature Reduction: 0.791
Mean Accuracy after CV: 0.790 +/- 0.003
Best Accuracy after CV: 0.793
```



Tuning Hyperparameters

Before making use of stacking and ensembles of different models, finding the best possible single model given the above procedures was explored. Using GridSearchCV to determine the optimal number of estimators (albeit on a limited range due to computational constraints), it was determined that using 350 estimators vs the default 100, improved the model. This yielded the below results as well as an accuracy of approx. 80% when used on the blind holdout.

```
====================
best params: RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=None, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=350, n_jobs=None,
            oob_score=False, random_state=123456, verbose=0,
            warm_start=False)
best params: {'n_estimators': 350}
best score: 0.8
====================
====================
Accuracy of Tuned classifier on hold-out set: 0.809
Mean Accuracy after CV: 0.807 +/- 0.003
Best Accuracy after CV: 0.812
```

Stacking Models

In order to increase model accuracy, the next step taken was to stack models whereby certain groups of models would become “experts” at specific predictions in the hopes that their combined capabilities will be superior to any individual model. A number of different setups were tested including a single 3 model ensemble that predicted solely functional wells, which was fed into a final meta model among other possibilities. However, it was eventually found

that having 3 separate groups of models predict each of the 3 categories provided superior accuracy despite the greater runtime. Additionally, different combinations of models were tested based on the prior model comparison. XG Boost was originally considered, however the computational expense of this model (particularly when tuning) proved to be greater than the benefit of using it. Therefore, Random Forest, K Nearest Neighbors, and Gradient Boosting Trees were ultimately used for this purpose.

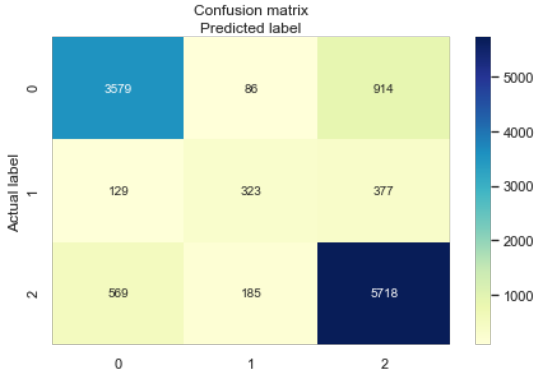
Tuning these models also proved to be difficult, due to increased computational expense of tuning 9 separate models. RandomizedSearchCV was eventually utilized to reduce runtime as a number of different parameter scenarios were considered. After the models were tuned to their specific target predictions (i.e. functional, non-functional, and function with repair), the type of result they would produce was experimented with. Initially, they would be simple classification models and provide a binary output with regards to their target. However, it was found that outputting probabilities yielded superior results due to the more robust degree of possibilities. These results were further cross-validated to ensure that each individual model and ensemble would provide accurate results. Utilizing these outputs as inputs for a final meta model yielded the following results:

```
Accuracy of Stacking: 0.809
Mean Accuracy after CV: 0.884 +/- 0.052
Best Accuracy after CV: 0.913
```

After applying GridSearchCV again to tune hyperparameters (in particular the number of estimators) for the meta model, the following improvement could be seen which yielded a final accuracy on the blind test set of approx. 81%.

Accuracy of Tuned Stacked classifier on hold-out set: 0.810
Mean Accuracy after CV: 0.959 +/- 0.001
Best Accuracy after CV: 0.962

	precision	recall	f1-score	support
0	0.84	0.78	0.81	4579
1	0.54	0.39	0.45	829
2	0.82	0.88	0.85	6472
micro avg	0.81	0.81	0.81	11880
macro avg	0.73	0.68	0.70	11880
weighted avg	0.80	0.81	0.80	11880



Conclusion

From the iterative results, it was clear that the data preparation stages had a large impact on model performance. Variables such as latitude, longitude, and population proved to be very significant and imputing them properly would have a material effect. Likewise, proper handling of the excessive number of categories had to be balanced between reducing the number of categories to improve model performance (from the perspective of removing noise and reducing computational expense) as well as maintaining the proper amount of information to train the model.

The final stacked model provided acceptable results that proved to be superior than a single Random Forest model, even when tuned. However, it should be noted that the tuning parameters were fairly conservative in order to minimize runtime and overfitting. Ultimately, the final model should seek to generalize well across different sets of data, even if that leads to a slightly lower accuracy in this particular instance. With greater computing power, more aggressive estimator and depth parameters could be used to further improve accuracy, where parameters such as min_sample_split, min_samples_leaf, and others, could have been utilized to avoid risk of overfitting. Additionally, techniques such as LDA/QDA could possibly be explored to find more effective ways to separate the three target categories. Although further testing will be needed to improve the accuracy, it can be seen that the positive effect of any individual change to the ETL or model at this stage would be marginal.