Alexia Marcous
Dec 2019

Predicting Water Pump Status
Milestone Report 2

Choosing Algorithms

Algorithms were chosen based on the properties of the data and target variable. The data from the Tanzanian Ministry of Water prepared by Driven Data was formed into two datasets that were labeled and provided the target variable, pump status, making this a supervised learning problem. The target variable has three categories, 'functional', 'functional needs maintenance' and 'non functional. With more than two classes, this is a multi-class classification problem. Five algorithms were chosen for this problem type, including K-Nearest Neighbors (KNN), Logistic Regression, Random Forest Classifier, Adaptive Boosting (AdaBoost) and Extreme Gradient Boosting (XGBoost).

Choosing Evaluation Metrics

Evaluation metrics were chosen based on the distribution of the data over the target, and the implication of false predictions. In this case, the data is imbalanced, with 54% of the pumps in 'functional' status, 39% in 'non functional' status and only 7% in 'functional needs maintenance' status. For this reason, F1 score was chosen over accuracy.

The F1 score was also appropriate due to a difference in the cost of false positives (predicting a functional status when the pump is not working) and false negatives (predicting a non functional status when the pump is working). A false positive could mean that the pump is ruled out for maintenance, resulting in it remaining in a non functional status, potentially causing pain and suffering in the local population. For this reason, the appropriate evaluation metrics are the true positive rate, also known as sensitivity or recall, and precision, or how "precise" the classifier is when predicting positive instances. Since the F1 score is the weighted average of precision and recall, if the F1 score is high, both precision and recall of the classifier indicate good results.

Preprocessing

To prepare the data for the machine learning algorithms which can only analyze numeric data, categorical values for all features of type object were converted to numeric values with the pandas factorize method. Since the categorical values in the dataset were ordinal, meaning they had no ranking, the data needed to then be one-hot-encoded. This was done with the pandas get_dummies method, specifying that n-1 features be created for n values in order to avoid creating problems of collinearity. The number of features in the low-variance dataset went from 11 to 53. The number of features in the high-variance dataset went from 14 to 56.

Once the data was encoded, it was split into training and test sets, using an 80/20 split. Each set was then scaled with the StandardScaler from scikit learn.

Model Creation and Prediction

There were 4 steps to the model creation and prediction process.

1. The 5 models were run on the low-variance dataset. Feature importances were evaluated for performance improvement.
2. The 5 models were run on the high-variance dataset. Feature importances were evaluated for performance improvement.
3. Using the dataset that produced the best results, the 5 models were optimized with cross-validation and hyperparameter tuning.
4. Using the best dataset and tuned models, principal component analysis was performed for performance improvement.

Step 1: Low-Variance dataset

Model results

The algorithms were run with default parameters on the low-variance dataset . The resulting F1 scores were:

F1 scores	0 - Functional	1 - Non Functional	2 - Functional Needs Repair
KNN	0.78	0.65	0.13
Logistic Regression	0.77	0.65	0.05
Random Forest	0.82	0.77	0.31
AdaBoost	0.78	0.66	0.04
XGBoost	0.80	0.68	0.12

Random Forest was the top performer on the low-variance dataset with default model parameters.

Feature Importance

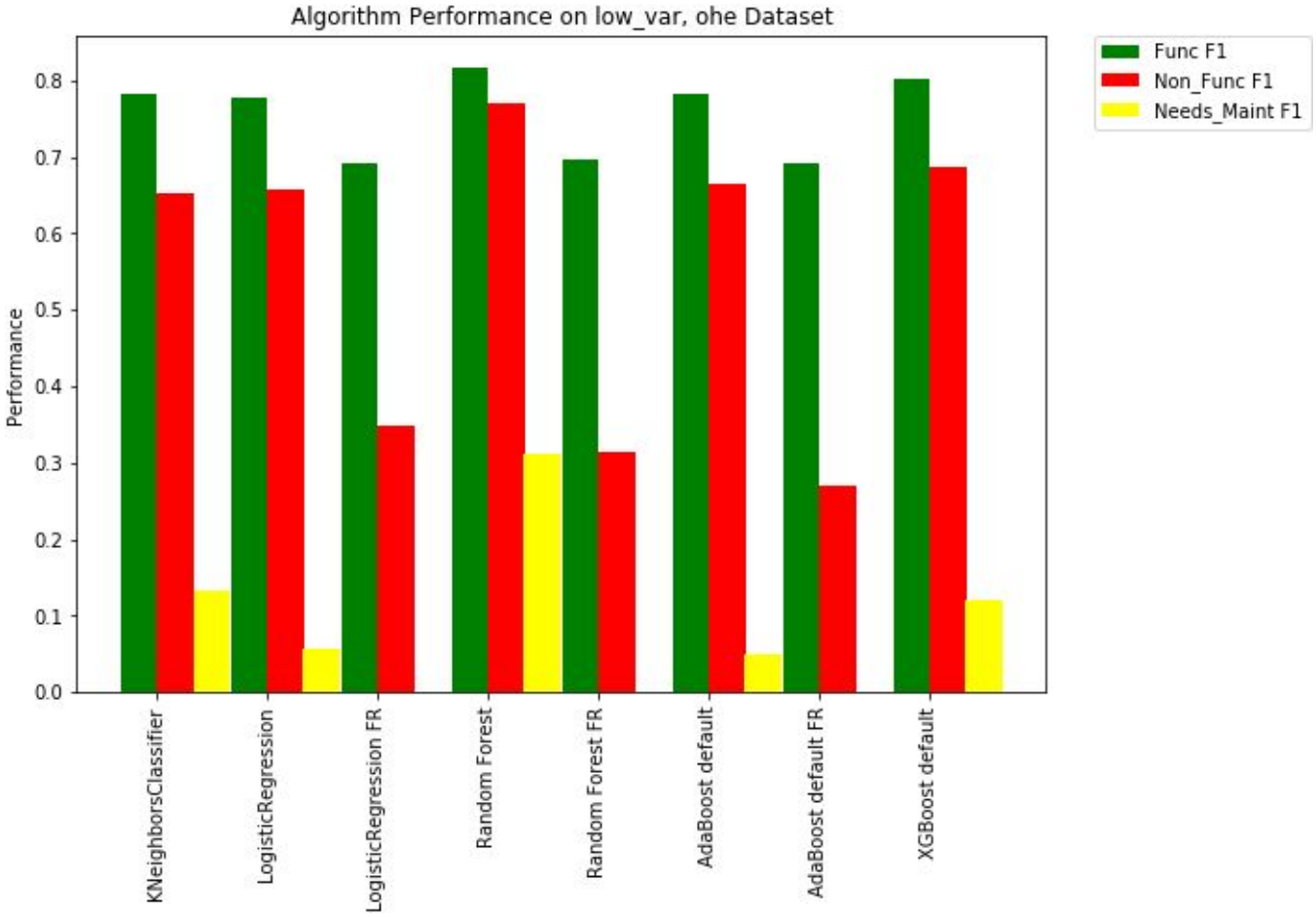
In an attempt to improve these results, feature importances were evaluated where models provided them. Features with coefficients less than a model-specific threshold were dropped from the data and the models were re-run. Of note, different features were dropped for different models. The resulting F1 scores were as follows, with changes from the full-featured dataset noted:

	Feature drop threshold	# Features dropped	0 - Functional	1 - Non Functional	*2 - Functional Needs Repair
Logistic Regression	< 0.05	14	0.69 -0.08	0.34 -0.31	0
Random Forest	< 0.005	17	0.69 -0.12	0.31 -0.46	0
AdaBoost	= 0	14	0.69 -0.09	0.26 -0.40	0

* each model produced an undefined metric warning indicating there were no predicted samples for the 'Functional Needs Repair' class.

All of the models had a drop in performance following feature reduction. This could be because the thresholds were too low, something that can be evaluated in future work on this and other projects.

Model performance on low-variance, one-hot-encoded dataset summary



Step 2: High-Variance dataset

Model results

The same algorithms were run with default parameters on the high-variance dataset. The resulting F1 scores were as follows, with changes from the low-variance dataset results noted:

F1 scores	0 - Functional	1 - Non Functional	2 - Functional Needs Repair
KNN	0.78	0.63 -0.02	0.15 **+0.18**
Logistic Regression	0.78 **+0.01**	0.66 **+0.01**	0.04 **+0.01**
Random Forest	0.83 **+0.02**	0.79 **+0.02**	0.38 **+0.07**
AdaBoost	0.78	0.67 **+0.01**	0.09 **+0.05**
XGBoost	0.80	0.69 **+0.01**	0.14 **+0.02**

Random Forest was also the top performer on the high-variance dataset with default model parameters. Of note, all of the models improved on predicting the 'Functional Needs Repair' status.

Feature Importance

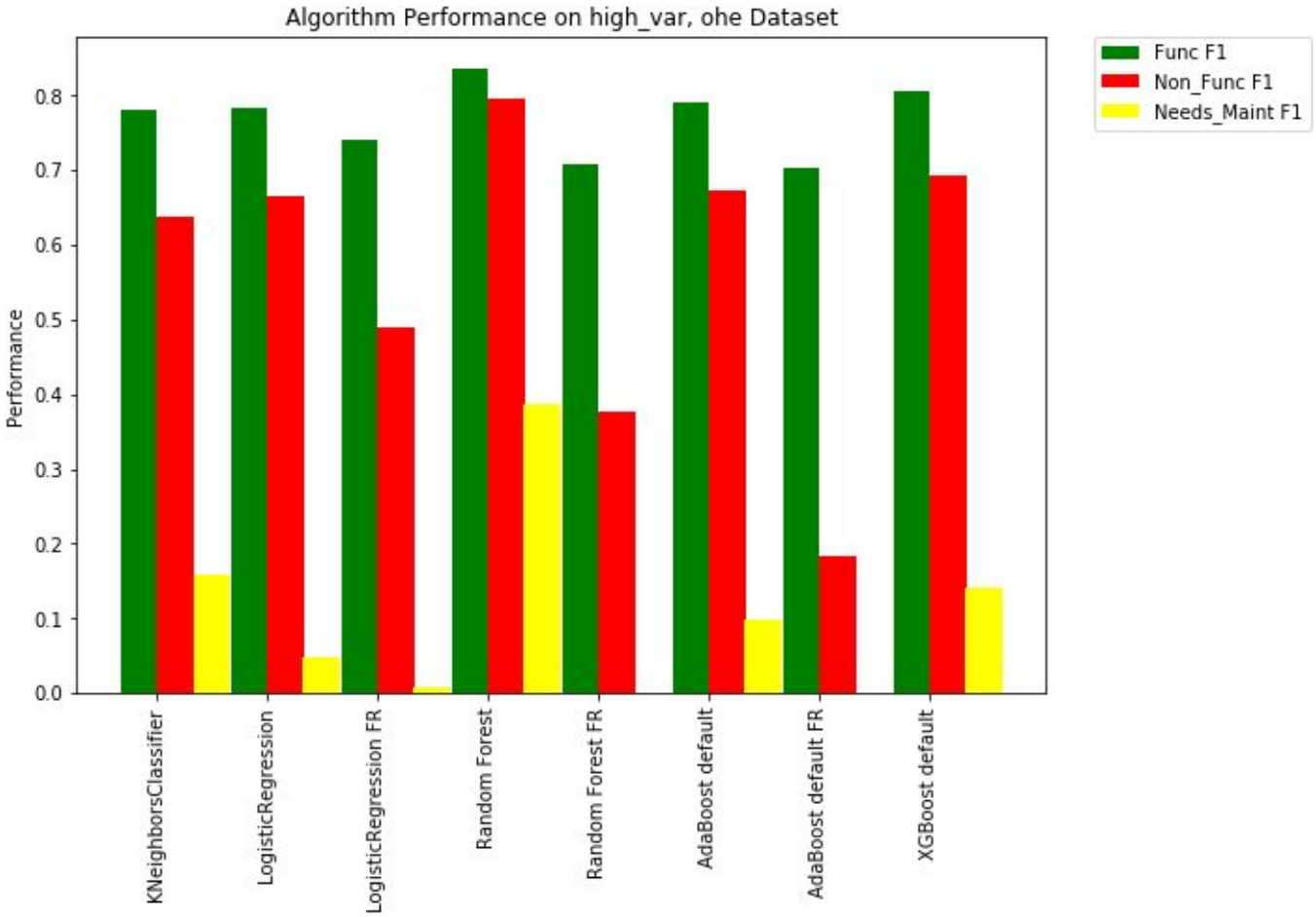
As with the low-variance dataset, an attempt was made to improve these results by evaluating feature importances where models provided them. Features with coefficients less than a model-specific threshold were dropped from the data and the models were re-run. Of note, different features were dropped for different models. The resulting F1 scores were as follows, with changes from the full-featured high-variance dataset noted:

	Feature drop threshold	# Features dropped	0 - Functional	1 - Non Functional	*2 - Functional Needs Repair
Logistic Regression	< 0.05		0.74 -0.08	0.48 -0.24	0.006
Random Forest	< 0.005		0.70 -0.13	0.37 -0.42	0
AdaBoost	= 0		0.70 -0.08	0.18 -0.49	0

* The Random Forest and AdaBoost models produced an undefined metric warning indicating there were no predicted samples for the 'Functional Needs Repair' class.

As with the low-variance dataset, all of the models had a drop in performance following feature reduction. This could also be because the thresholds were too low, something that can be evaluated in future work on this and other projects.

Model performance on high-variance, one-hot-encoded dataset summary



Step 3: Cross validation and hyperparameter tuning

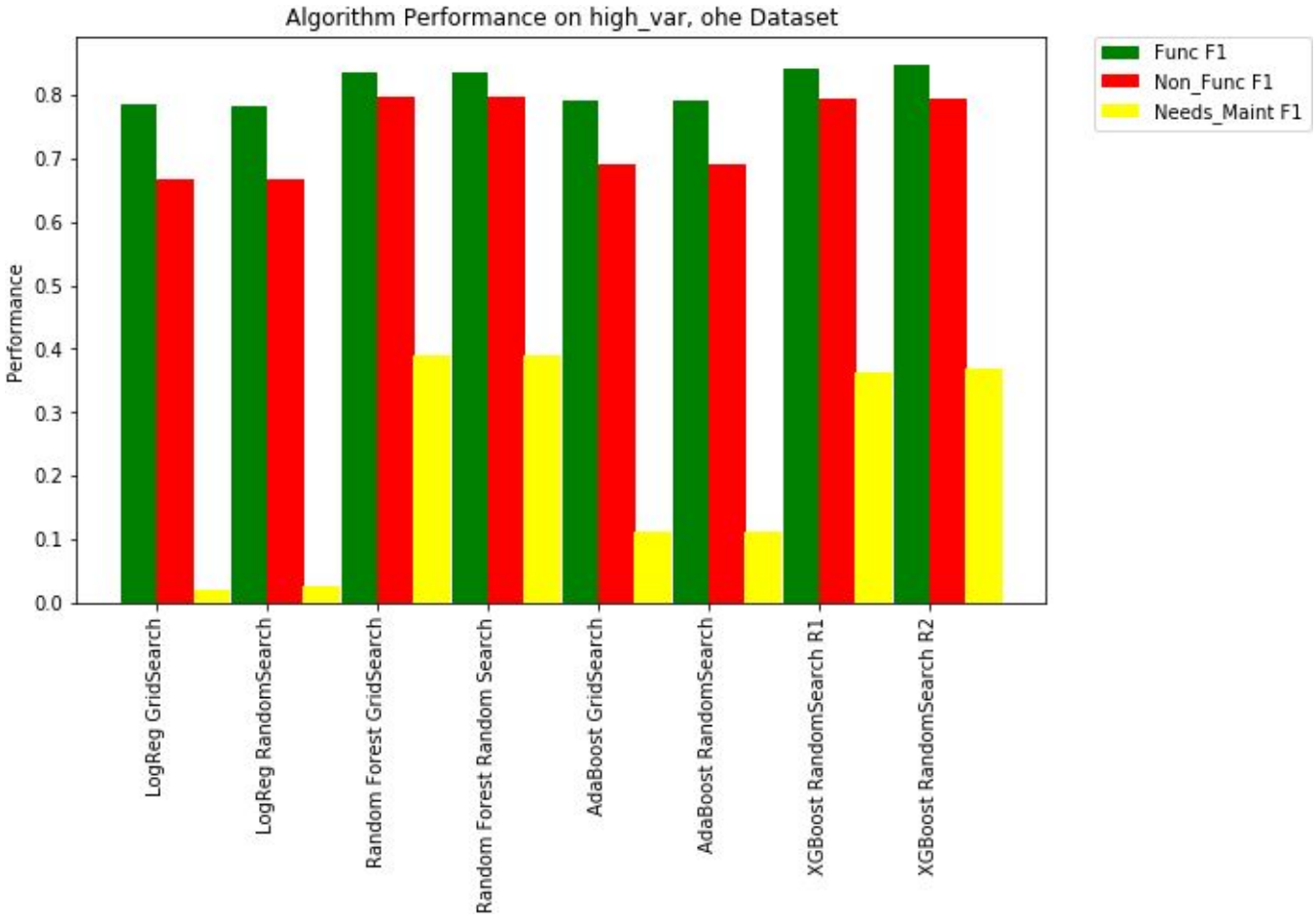
With the better of the two datasets identified, the next step towards improving the model results was to perform cross validation and hyperparameter tuning on the top 4 highest performing models. 5-fold cross validation ran the models through more training and testing scenarios by subsetting the existing data into new test and training sets. Hyperparameter tuning tested multiple scenarios of inputs to the learning process. Both grid search, an exhaustive exploration, and random search, a partial exploration of the matrix of possible hyperparameters were run on the logistic regression, random forest and Adaboost models. Given that hyperparameter tuning on XGBoost required multiple rounds due to the number of parameters available for tuning, only random search was performed for expediency. The results were as follows:

F1 scores	Search Type	0 - Functional	1 - Non Functional	2 - Functional Needs Repair
Logistic Regression	Grid	0.78	0.66	0.02 -0.02
Logistic Regression	Random	0.78	0.66	0.02 -0.02

Random Forest	Grid	0.83	0.79	0.38
Random Forest	Random	0.83	0.79	0.38
AdaBoost	Grid	0.79 **+0.01**	0.68 **+0.01**	0.11 **+0.02**
AdaBoost	Random	0.79 **+0.01**	0.68 **+0.01**	0.11 **+0.02**
XGBoost Round 1	Random	0.84 **+0.04**	0.79 **+0.10**	0.36 **+0.22**
XGBoost Round 2	Random	0.85 **+0.05**	0.80 **+0.11**	0.37 **+0.23**

Three rounds of tuning was planned for XGBoost, but the second round took 38 hours, so additional tuning was left for future work. XGBoost produced the highest-scoring model for the first two classes, and Random Forest remained in the lead for the 'functional needs maintenance' class.

Model performance on high-variance, one-hot-encoded dataset with hyperparameter tuning and cross validation summary

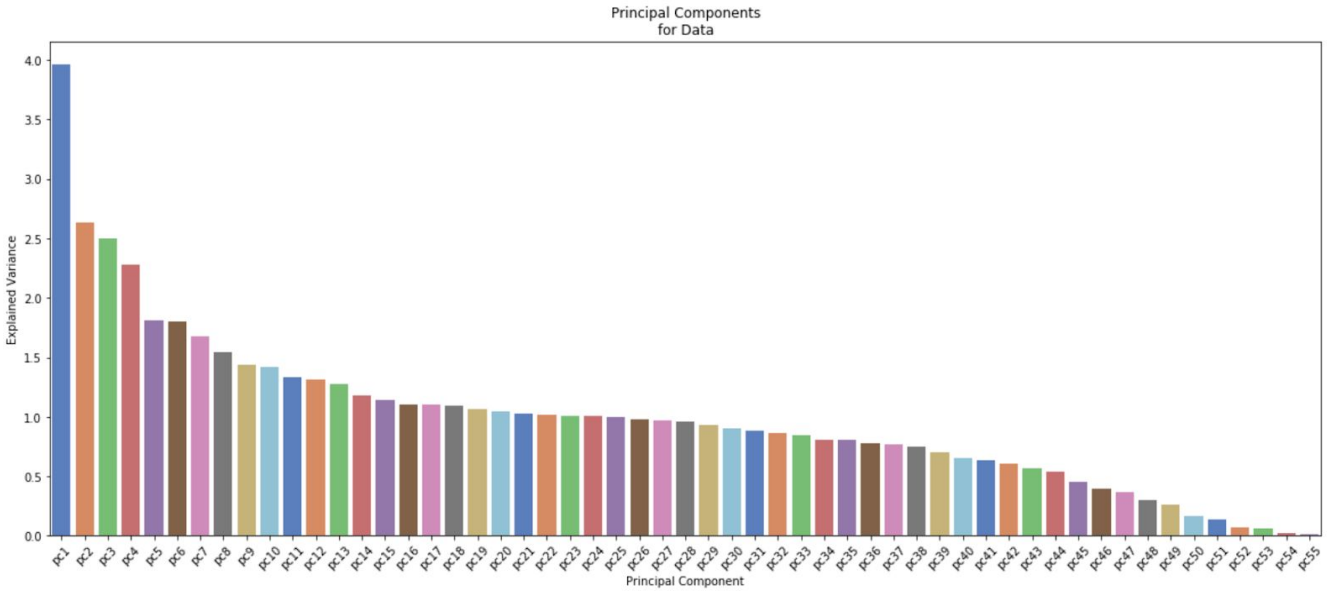


Step 4: Principal Component Analysis

With the better of the two datasets and the best of the tuned models identified, the next and final step towards improving the model results in this project was to perform principal component analysis (PCA). This is a dimension-reduction technique, aimed at including fewer relationships between variables in an attempt to prevent overfitting by capturing the variance of the data in as few components as possible.

The drawback to PCA is that it reduces the interpretability of the results. We no longer have an understanding of which features are relevant. This is not in alignment with the original goals of the project, but if more water pumps are kept operable by a better predictive outcome, that still achieves the overall goal.

A chart of the principle components ordered by explained variance shows the results from fitting the PCA instance on the training data:



Test and training sets were reduced to 13 components, the first leveling-off point, indicating diminishing returns with additional components. Given the 38-hour duration of the best-performing model, the next-best performing classifier, the tuned RandomForest model, was used first to determine if there was any improvement to be gained using PCA.

The model ran quickly on the reduced data but the results are not as good as the model produced before, for F1 scores of all classes. This could be because the cutoff of principal components was too aggressive. The model was run again using 38 components, but produced results worse than with 13 components.

Since performing dimension reduction with feature extraction via PCA did not improve the results on the next-best model, we will not attempt it on the top-performing model, XGBoost. Given the impractical duration of tuning that model further, we will leave that for future improvements.

Summary

This 4-step approach revealed that the high-variance dataset produced better results, which were not improved with feature reduction via exclusion by importance threshold. The results were improved with

hyperparameter tuning and cross validation, although with drastic increase in run time. Principal Component Analysis improved run time, but sacrificed results.

Of note, the XGBoost model took 38 hours to run in the second round of hyperparameter tuning. Utilizing PCA for feature extraction did improve performance on the Random Forest model, but lead to sub par results.

No one model performed the best across all three target classes on measures of F1 scores. The following is a summary of the two top-performing models:

	Random Forest			XGBoost		
	Precision	Recall	F1	Precision	Recall	F1
'Functional'	0.81	0.87	0.84	0.80	0.89	0.85
'Non Functional'	0.82	0.77	0.80	0.83	0.77	0.80
'Functional needs maintenance'	0.48	0.33	0.39	0.55	0.28	0.37

With the goal of helping to improve the management of Tanzania's water resources and ensure access to potable water for its citizens, we can evaluate these models by reviewing the recall scores to determine how helpful they are in terms of knowing when a pump is broken or needs maintenance. Of all the pumps that were actually non functional, both models predicted that correctly 77% of the time. Of all the pumps that needed maintenance, the Random forest model predicted that correctly 33% of the time, while the XGBoost predicted that correctly 28% of the time. Based on this evaluation criteria, we can declare the Random Forest model to be the most useful.

Future Work

Future work on this project could include bringing in data from other sources to fill in missing data, such as population, or augment the data, such as annual rainfall. It would be interesting to review other preventive maintenance programs to learn best practices. Work could also be done to help address the imbalance in the target classes, such as bootstrapping the data, using weights in model creation, or evaluating the impact of combining classes. Additional tuning could also be done to attempt to improve results further.