This capstone project has been really interesting. At first, there were a lot of missing values and I was able to fill those missing values using appropriate methods covered on the Milestone Report.

The target variable (status_group) consists of three classes of water pumps: Functional, Non-function, and the ones that require repair work. This is a classic classification problem of how to accurately predict the classes. The challenge is if we can predict which pumps are functional.

In the Github Jupyter Notebook for Machine Learning part, you will see:

1. Multiple lists containing the catg_feature_names, num_feature_names and date_feature_names
2. Reduction in the levels of categorical data (keeping the top 10 and replacing the rest with 'OTHER'
3. compose.ColumnTransformer encoding of categorical variables
4. Attaching names and levels to each transformed column
5. Replacing (status_group) values with 0s and 1s -functional = 1 - broken,needs repair = 0
6. Trying Decision tree model and also using GridSearchCV to find the best params. Also added ROC Curve and AUC score to check the accuracy
7. Random forests model and also using GridSearchCV to find the best params. Also added ROC Curve and AUC score to check the accuracy
8. Comparing the model scores at the end

On line 3, you can see that I've used three lists that hold names on my categorical, numerical and date features. This helps me later on to add/remove any of these features from my training and test sets.

On lines 4 and 5, I've divided date_recorded into three features. Later, I will use them to see if any of these features help with any of my models.

After cleaning our original dataset, we still have a lot of values in some of our categorical variables. In order to help our models and to make the fitting process a bit faster, I've decided to use a function on line 6 that only keeps the top 10 (based on value_counts()) in each categorical feature. Then, I set the rest to "OTHER". This helps with having less transformed columns when we do the ColumnTransformer in later steps.

On line 12, I'm using compose.ColumnTransformer which helps transform (encode) all the categorical features at once! I've defined a pipeline (transformers) that gets the categorical feature names from the list I defined on line 3, and then performs the OneHotEncoder on them. OneHotEncoder, encodes categorical features as a one-hot numeric array. By default, the encoder derives the categories based on the unique values in each feature and this is the reason we reduced the feature sizes to the top 10.

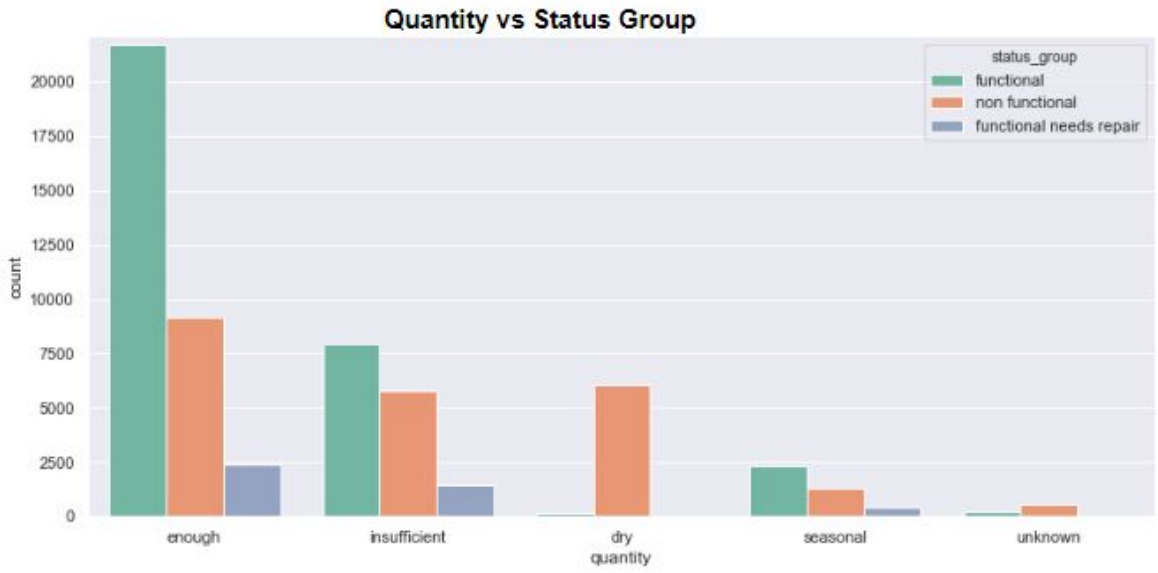
The one issue that I ran into was that the columns would be labeled with digits and I didn't know which ones would be representing the values from each original column. I did a bit of research and found a useful function that would attach the name of the original columns to the values and add them as the new column names. You can see this function on line 15. On this line, I also concatenate all the transformed categorical features with my numerical features and the date features.

On line 41, I decided to binarize the target (status_group) values. I assigned all of the functional pumps to 1 and all the broken, needs repair ones to 0.

On line 19, I split the data into training and test sets. I split the data using 70% to training and 30% for testing. I also set the random_state=42 to make sure my split is always the same.

On line 20, I start using my first model which is the Decision Tree Classifier. I also set a few hyperparameters for this model but later on I will use the GridSearchCV to find the best_params_ for this model. The accuracy score for my Decision Tree model is 71% which is not bad. In order to make sure that the model is not guessing the output, I use the ROC Curve as well as the AUC score on lines 24 and 25. The AUC score is 0.7 which is greater than 0.5 and this means the model is better than random guessing. Always a good sign!

On line 26, I used my model's feature_importances_ parameter and plotted the most important features of my model. It looks like quantity-dry plays a very important role in my Decision Tree model. This is something that we have discovered previously that if the quantity is dry, then the pump is most likely non-functional.



This is a classification problem and I think the Random Forest Classifier is the best model to make predictions on this problem. I will also use a few mode models and then compare their scores at the end on my notebook.

On line 27, I start training my Random Forest model using the n_estimators=200 as well as setting the criterion='gini'. The accuracy score of my model is 82% which is great. I also use the GridSearchCV to find the best_params_ for this model. I also use the ROC Curve as well as the AUC score on lines 31 and 32. The AUC score is 0.898. We can see that the ROC Curve and AUC score for Random Forest Classifier is much better.

Plotting the most important features shows that longitude, latitude and quantity-dry are playing important roles in this model.

I also used the Extra Tree Classifier and XGBoost Classifier. The results are pretty close with the Random Forest model. You can see the score comparison on the last line of my notebook.

```
Decision Tree 0.7122895622895623
Random Forest 0.8182379349046016
Extra Trees 0.8182379349046016
XGBoost 0.8173400673400674
```