phase3_project

Description

To build a model to that can predict the condition of a waterpump based on certain input parameters

Dataset

Dataset sourced from https://www.drivendata.org/competitions/7/pump-it-up-data-mining-the-water-table/page/23/

Methodololgy

Since this a classifiaction problem, baseline models were built using Logistic Regression, Decision Trees and K-Nearest Neigbors. Out the of the three base models, the model with the best scores was chosen for further optimization

Test Scores of baseline models

Logistic Regression

	precision	recall	f1-score	support
functional	0.56	0.90	0.69	7945
non functional	0.00	0.00	0.00	1091
functional needs repair	0.56	0.21	0.31	5779
accuracy			0.56	14815
macro avg	0.38	0.37	0.33	14815
weighted avg	0.52	0.56	0.49	14815

Decision Tree

	precision	recall	f1-score	support
functional	0.79	0.79	0.79	7945
non functional	0.38	0.38	0.38	1091
functional needs repair	0.76	0.76	0.76	5779
accuracy			0.75	14815
macro avg	0.64	0.64	0.64	14815
weighted avg	0.75	0.75	0.75	14815

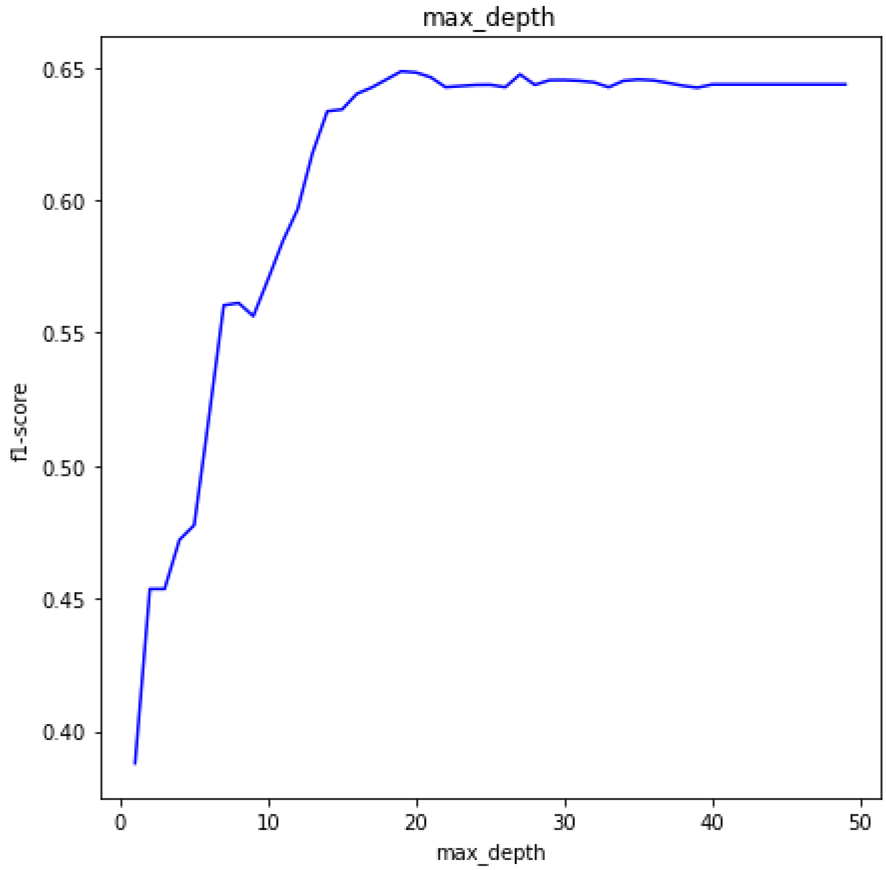
K-Nearest Neighbors

	precision	recall	f1-score	support
functional	0.65	0.76	0.70	7945
non functional	0.31	0.16	0.21	1091
functional needs repair	0.61	0.52	0.56	5779
accuracy			0.62	14815
macro avg	0.52	0.48	0.49	14815
weighted avg	0.61	0.62	0.61	14815

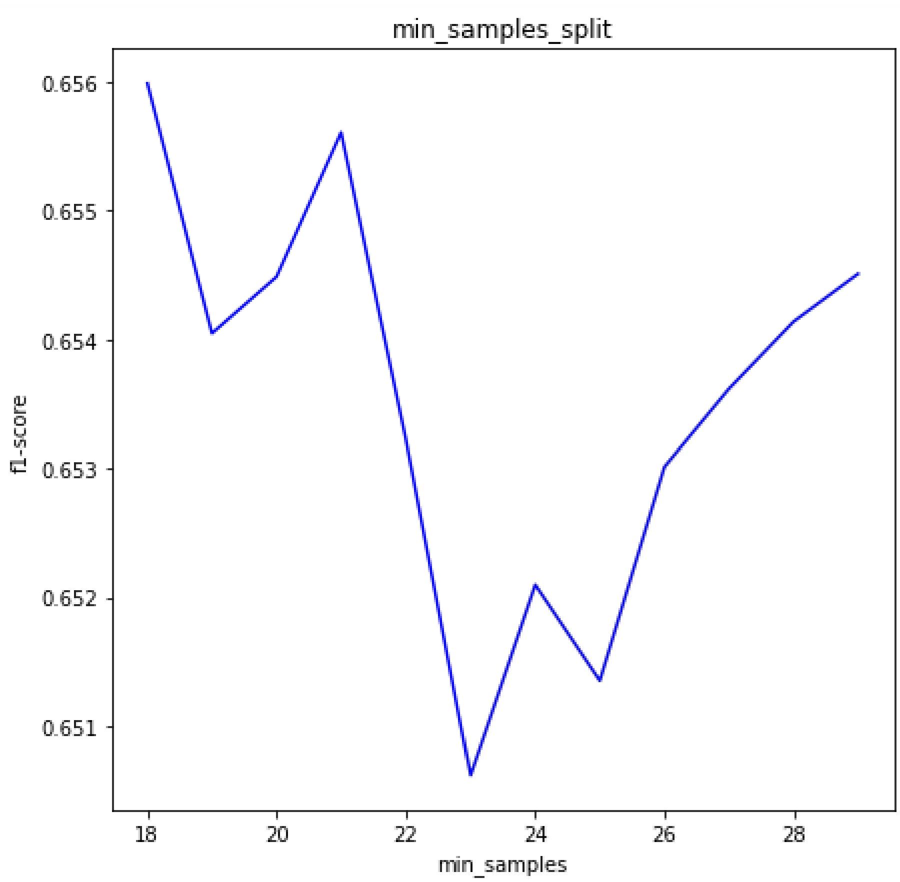
Since the Deccision Tree has the best recall score, we will use that for modelling and optimization.

Tuning Hyperparameters

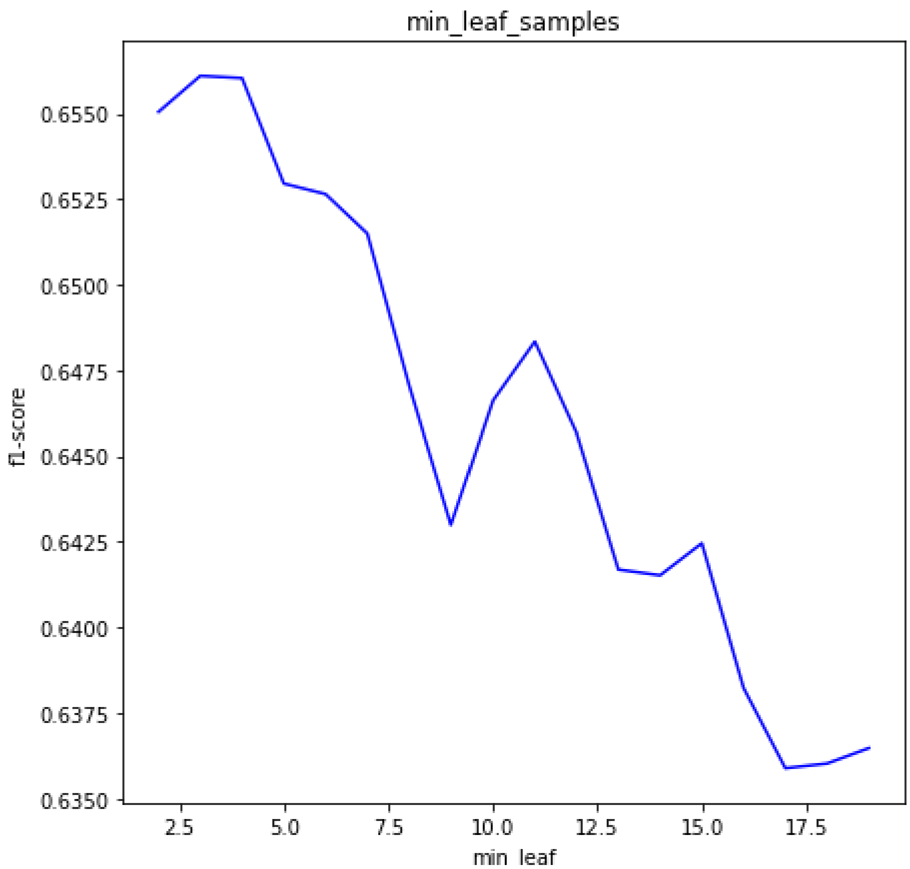
max_depth



min_samples_split



min_samples_leaf



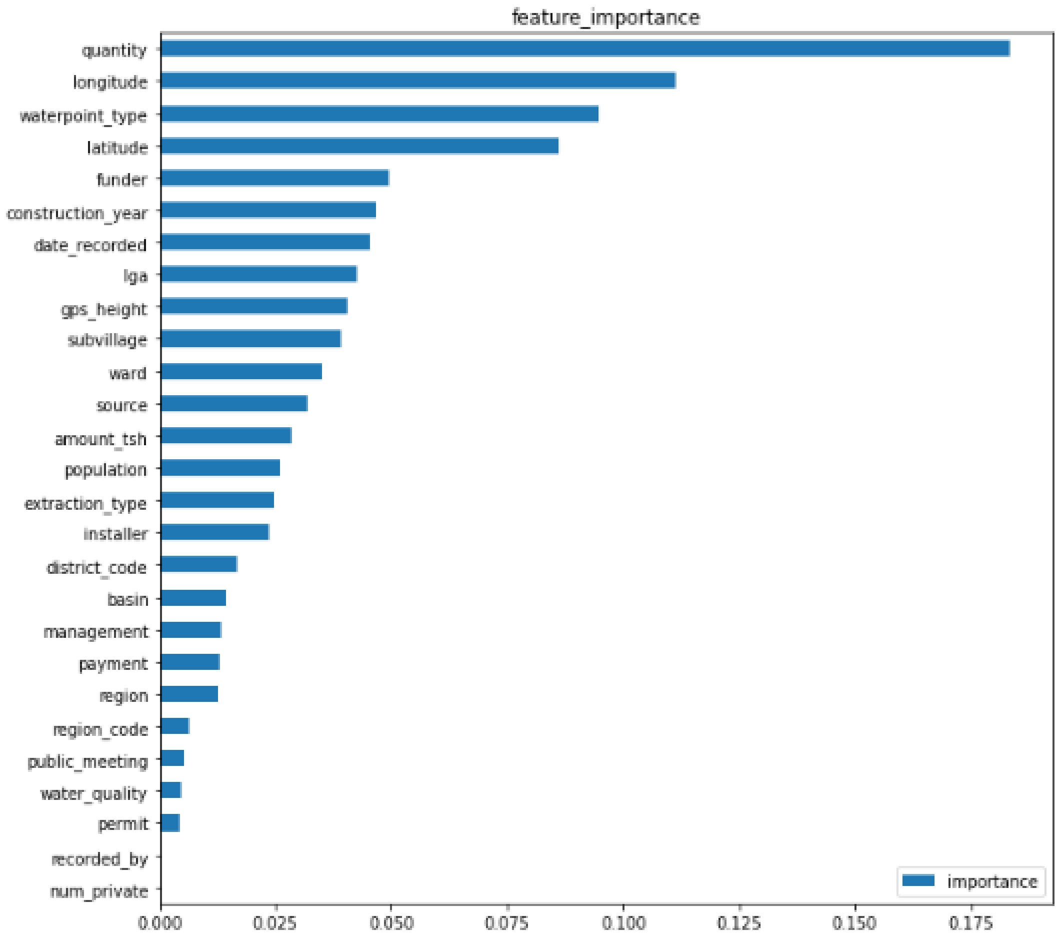
Building the model with the peak values:

```
max_depth:20   min_samples_split:30   min_samples_leaf:3
```

Result with the optimized parameters

```
TEST SCORES
----------------------------------------------------------
                         precision    recall  f1-score   support

             functional       0.78      0.85      0.81      7945
         non functional       0.49      0.32      0.39      1091
functional needs repair       0.79      0.74      0.76      5779

               accuracy                           0.77     14815
              macro avg       0.69      0.64      0.66     14815
           weighted avg       0.76      0.77      0.76     14815
```

Checking feature_importance



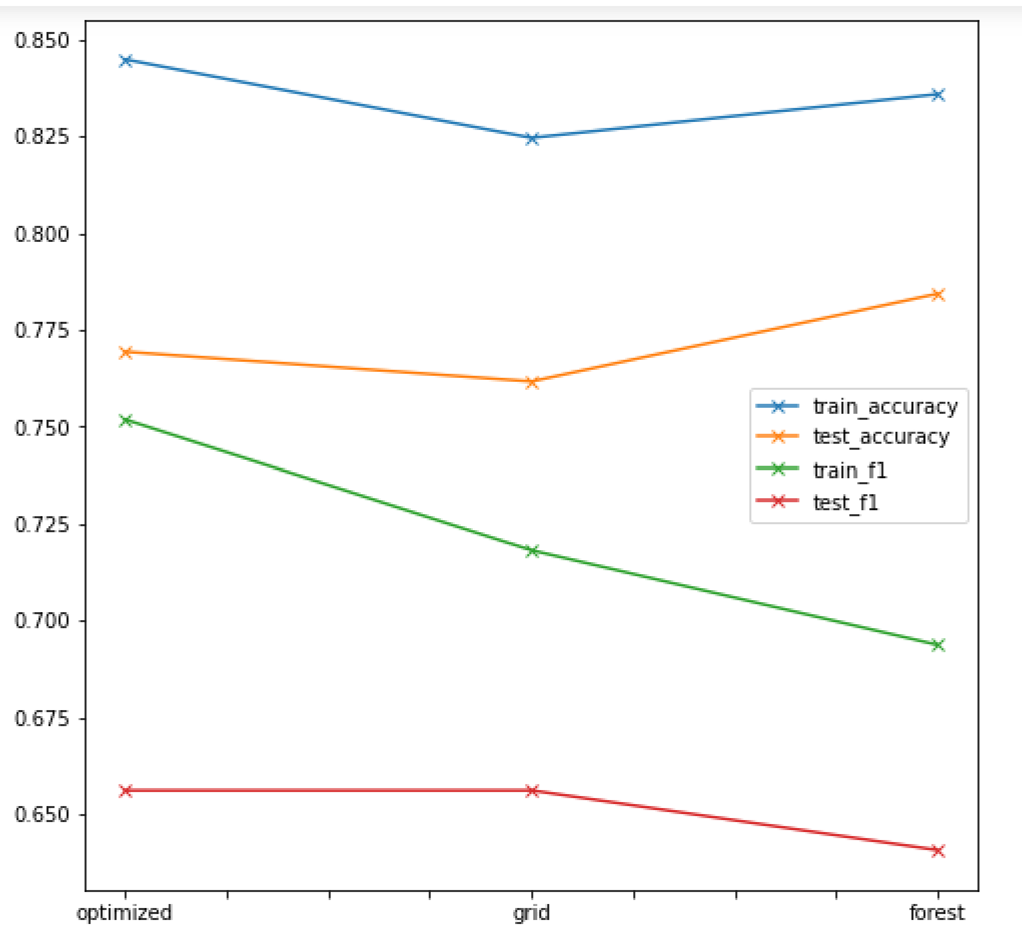
Using GridSearch on the model using only top10 features

```
TEST SCORES
--------------------------------------------------
                         precision    recall  f1-score   support

              functional       0.77      0.86      0.81      7945
functional needs repair       0.51      0.30      0.38      1091
          non functional       0.79      0.72      0.75      5779

                accuracy                           0.76     14815
               macro avg       0.69      0.62      0.65     14815
            weighted avg       0.75      0.76      0.75     14815
```

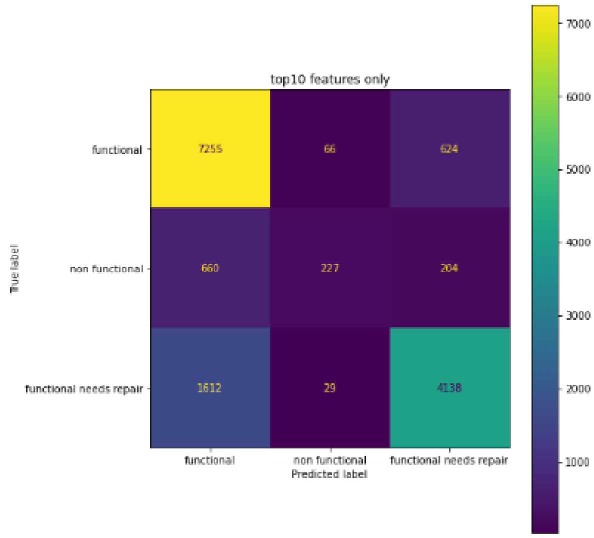
Random Forest

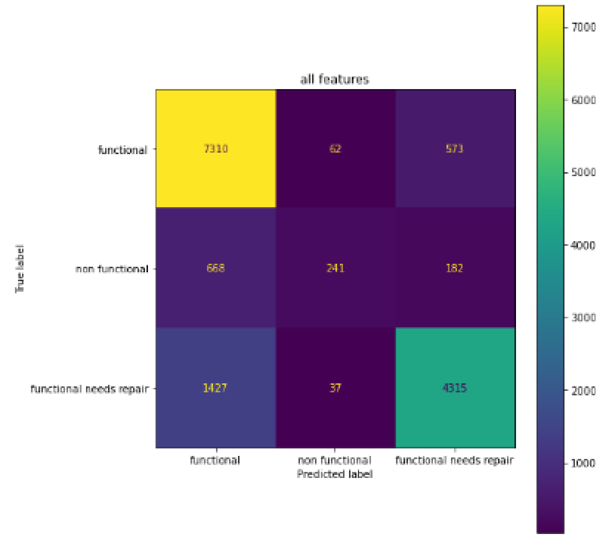
```
TEST SCORES
--------------------------------------------------
                         precision    recall  f1-score   support

             functional       0.76      0.91      0.83      7945
functional needs repair       0.70      0.21      0.32      1091
         non functional       0.83      0.72      0.77      5779

               accuracy                           0.78     14815
              macro avg       0.77      0.61      0.64     14815
           weighted avg       0.79      0.78      0.77     14815
```

Visualizing Scores of the model with optimized parameters, GridSearch and RandomForest



Checking the confusion matrix of model with top10 features Vs all_features





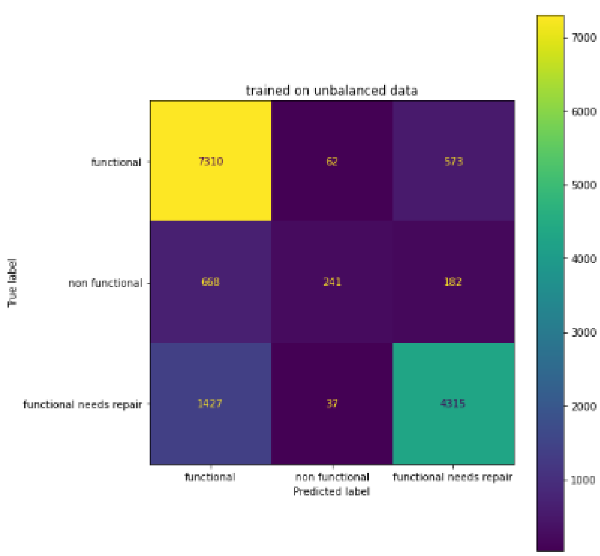
We can see that the features make a negligible difference.

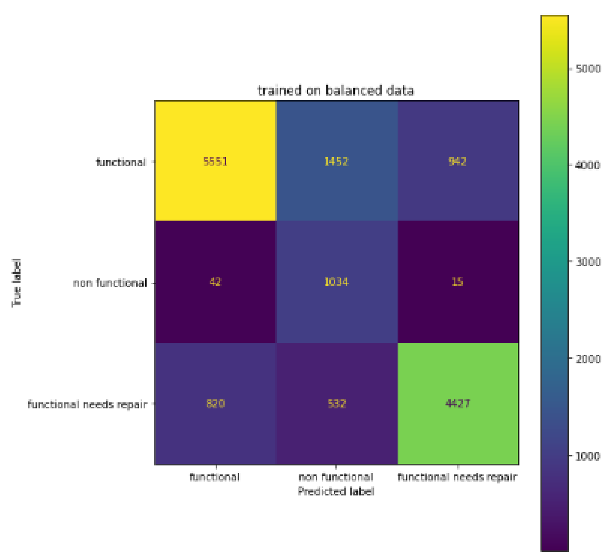
Examining target feature

```
functional                 32186
non functional             22765
functional needs repair     4308
Name: status_group, dtype: int64
```

We can see clearly that there is an imbalance in the different classes. We will now train a model on a balanced dataset and test it on the validation data to see check for model performance.

Confusion Matrix between balanced and unbalanced data





Next Steps

1. Possibly re-frame this as a binary classification problem i.e functional vs non-functional and see if we can build a better model.
2. Re-create the model with equal number of data points between functional and non-functional. Optimize parameters on this balanced dataset and test it on validation data to check for performance.

Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

Languages

- **Jupyter Notebook** 100.0%