# **Water Pump- Functional Status Prediction**

**Krishna Teja,Neethu Sundar,Supraja Suresh**

# 

[Overview](#_2et92p0)

[Problem](#_tyjcwt)

[Data Mining Task](#_3dy6vkm)

[Data](#_1t3h5sf)

[Challenges](#_4d34og8)

[Feature Engineering](#_2s8eyo1)

[Overview of Work](#_17dp8vu)

[Sampling](#_3rdcrjn)

[Missing Value Handling](#_26in1rg)

[Outlier Handling](#_lnxbz9)

[Feature Transformation](#_35nkun2)

[Feature Aggregation/Reduction](#_1ksv4uv)

[Feature Splitting](#_44sinio)

[Feature Selection](#_2jxsxqh)

[Model Selection & Tuning](#_z337ya)

[Overview of Work](#_3j2qqm3)

[Model 1](#_1y810tw)

[Model 2](#_4i7ojhp)

[Model 9](#_2xcytpi)

[Lessons Learned](#_1ci93xb)

# 

# 

# Overview

## **Problem**

Using data from [Taarifa](http://taarifa.org/) and the [Tanzanian Ministry of Water](http://maji.go.tz/), the problem is to predict which pumps are functional, which need some repairs, and which don't work at all. Prediction of one of these three classes is based on a number of variables about what kind of pump is operating, when it was installed, and how is it managed. This can improve maintenance operations and ensure that clean, potable water is available to communities across Tanzania.

## Data Mining Task

It is a classification problem more specifically a Multi-class classification problem, as we try to group the water pumps into one of the 3 categories : Functional, Functional but Needs Repair and Non-Functional .

## Data

The Training Dataset and Test Dataset contains data about pumps in the Tanzania region in East Africa.The test dataset doesn’t have the ground truth and hence the models were applied on the train set after splitting it to train and test set. The features are as follows:

* **amount\_tsh** - Total static head (amount water available to waterpoint)
* **date\_recorded** - The date the row was entered
* **funder** - Who funded the well
* **gps\_height** - Altitude of the well
* **installer** - Organization that installed the well
* **longitude** - GPS coordinate
* l**atitude** - GPS coordinate
* **wpt\_name** - Name of the water point if there is one
* **num\_private** -
* **basin** - Geographic water basin
* **subvillage** - Geographic location
* **region** - Geographic location
* **region\_code** - Geographic location (coded)
* **district\_code** - Geographic location (coded)
* **lga** - Geographic location
* **ward** - Geographic location
* **population** - Population around the well
* **public\_meeting** - True/False
* **recorded\_by** - Group entering this row of data
* **scheme\_management** - Who operates the water point
* **scheme\_name** - Who operates the water point
* **permit** - If the waterpoint is permitted
* **construction\_year** - Year the water point was constructed
* **extraction\_type** - The kind of extraction the water point uses
* **extraction\_type\_group** - The kind of extraction the water point uses
* **extraction\_type\_class** - The kind of extraction the water point uses
* **management** - How the water point is managed
* **management\_group** - How the water point is managed
* **payment** - What the water costs
* **payment\_type** - What the water costs
* **water\_quality** - The quality of the water
* **quality\_group** - The quality of the water
* **quantity** - The quantity of water
* **quantity\_group** - The quantity of water
* **source** - The source of the water
* **source\_type** - The source of the water
* **source\_class** - The source of the water
* **waterpoint\_type** - The kind of water point
* **waterpoint\_type\_group** - The kind of water point

**Distribution of Labels:**

The labels in this dataset are simple. There are three possible values:

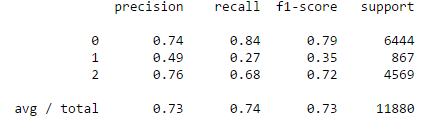
* **functional** - the waterpoint is operational and there are no repairs needed
* **functional needs repair** - the waterpoint is operational, but needs repairs
* **non functional** - the waterpoint is not operational

**Evaluation Metric**

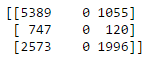
We will evaluate the model based on Accuracy and Recall , as the class functional but needs repair has low examples in training set and is hard to predict.Though Accuracy is a primary measure , Recall helps in identifying the pumps say that are functional but needs repair out of the total pumps that are functional. Precision can also included as it would help in letting the public know the correct status of the water pump and say out of those classified as functional the percentage ‘A’ is actually functional.

The Precision and Recall can be considered using classification report and confusion matrix.

Example of classification report:

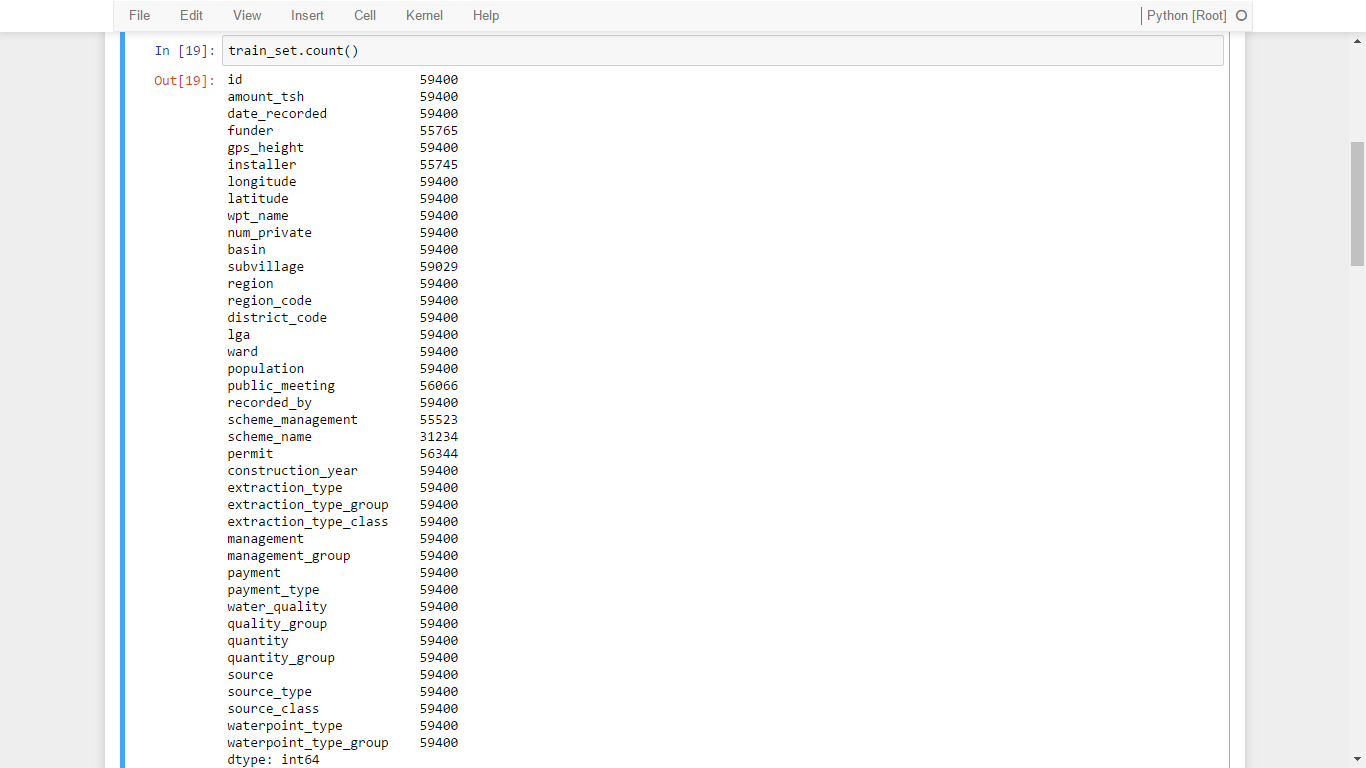


Example of confusion matrix :



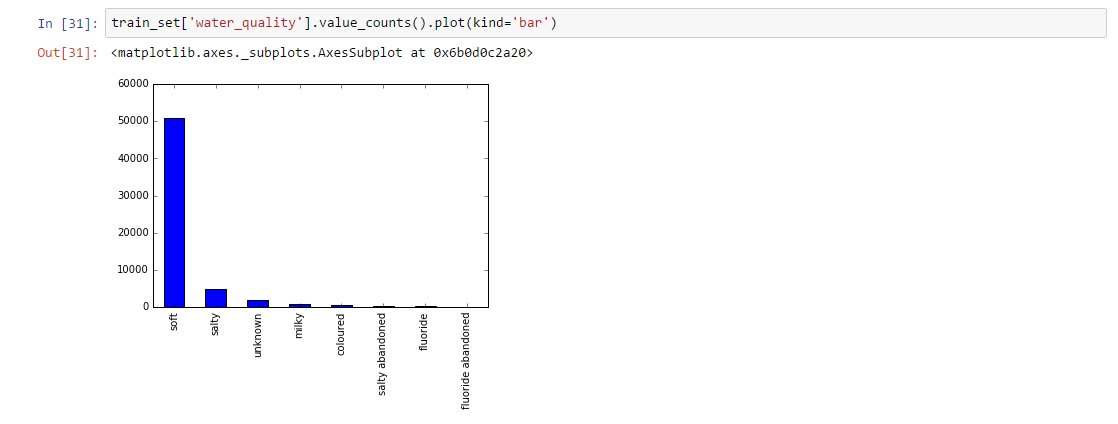
**Data Exploration**

Initial data analysis were performed using Pandas Dataframe. The following visualizations were made.

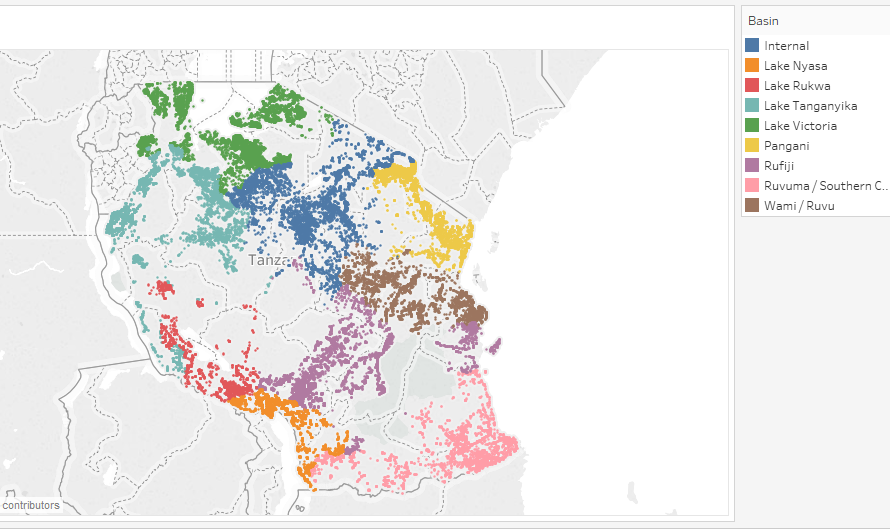




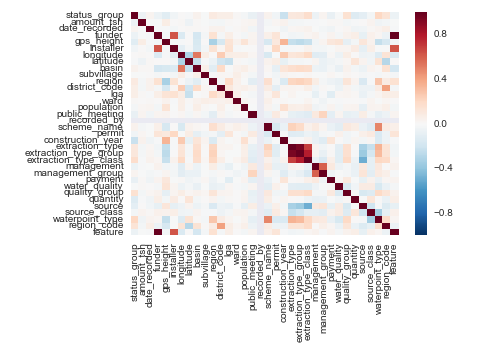
A bar plot gives the distribution of water pumps based on Water Quality.



The below map shows to which basin the water point belongs to



Correlation between the features are depicted in the below figure



From the above correlation matrix it can be seen that waterpoint type,quality, regioncode, lga, district code, region, date recorded, basin all play an important role in determining the status group.

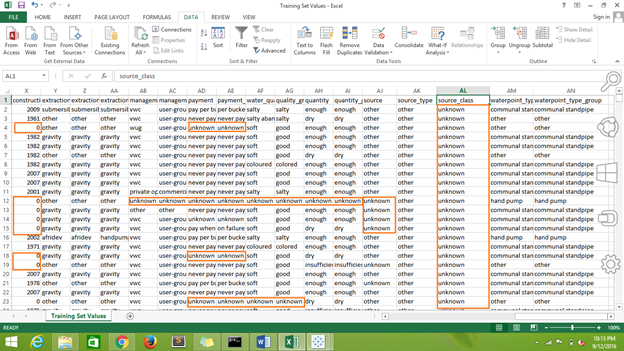
## Challenges

There about 59400 records and around 41 columns ,the dataset size can be described as small to medium. The features are mostly categorical with few continuous and numeric features.

**Data Quality issues:**

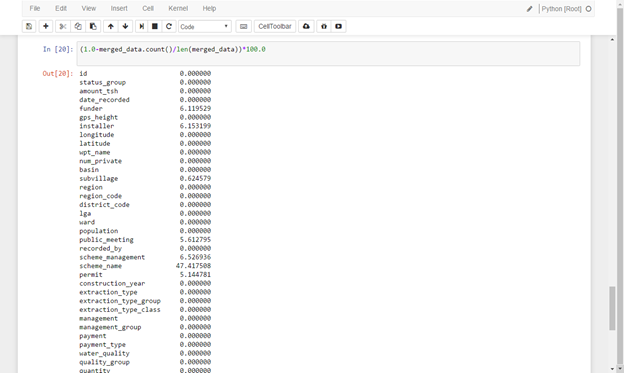
The dataset on the whole has few issues as discussed below:

There are missing values of certain features in most records.



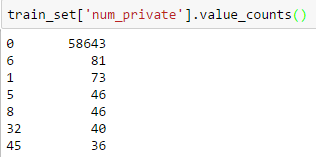
Using pandas dataframe, percentage of missing values were visualised:

Funder, installer ,scheme\_name,scheme\_management and public\_meeting has major missing values which could affect our prediction if not handled correctly or using a model that is robust in spite of noise.

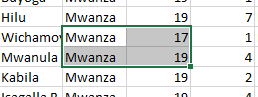


* num\_private field, though it doesn’t have missing values but has 0 as most of it’s values.

Out of 59400 records , 58643 are 0,as it doesn’t indicate anything it can be dropped.

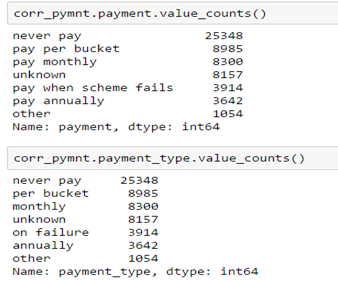


* The other issue is the same region has different region codes , this in real world is not possible and so it has to be fixed.Eg: Mawanza region has 2 different region codes like 19 and 17 as it’s region\_code.

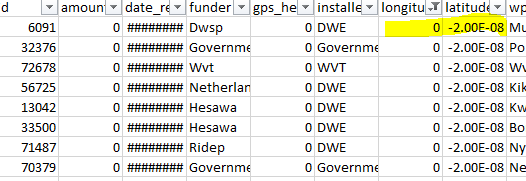


* Several columns have duplicate values , for example payment and payment\_type as indicated below. There are other columns such as source','source\_type','source\_class',Waterpoint\_type','waterpoint\_type\_group , payment','payment\_type', extraction\_type',’extraction\_type\_class’,'extraction\_type\_group'

Based on it’s correlation with the label we can handle it by combining or dropping it.



* There are few outliers in the latitude and longitude field where they correspond to regions outside Tanzania , and this has to be correctly handled and could affect the prediction



* Other issues are that funder , installer, wpt\_name have several unique values and this might tend to overfitting the data as few values occur for less than 10 of the entire records.
* The distribution of data among the different classes is slightly imbalanced but can be used as such to determine the model.
* Few of the values in features like watepoint\_type are underrepresented , like seen in the graph below dam has fewer records while the communal standpipe has a good deal of records.This might introduce a bias in the data

**Consolidated view of Challenges in the data are :**

1. Too many missing values - fields like funder, installer,scheme\_name,scheme\_management, construction\_year, has too many missing values
2. Duplicate columns - The below columns are redundant
   1. quantity','quantity\_group'
   2. source','source\_type','source\_class'
   3. Waterpoint\_type','waterpoint\_type\_group'
   4. payment','payment\_type'
   5. extraction\_type',’extraction\_type\_class’,'extraction\_type\_group'
3. Values not matching the real-world values - region\_code values are different for the same region , this must be addressed as the region plays an important role in determining the pumps status
4. Irrelevant features with null values
5. High Number of values in features - There are several categorical features with high number of unique values and this has to be reduced.For example funder, installer,..

# 

# 

# Feature Engineering

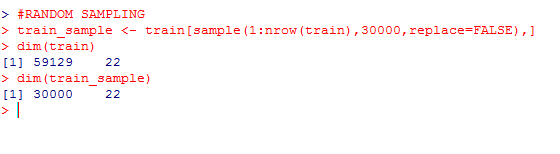
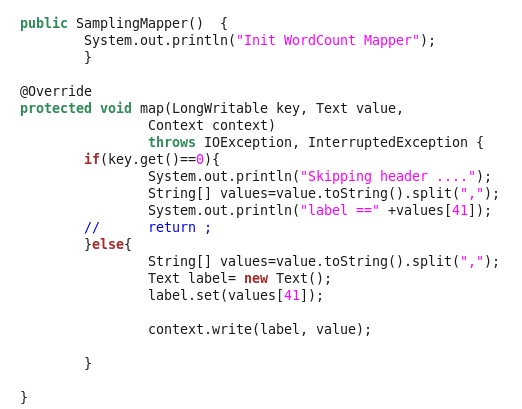
## Overview of Work

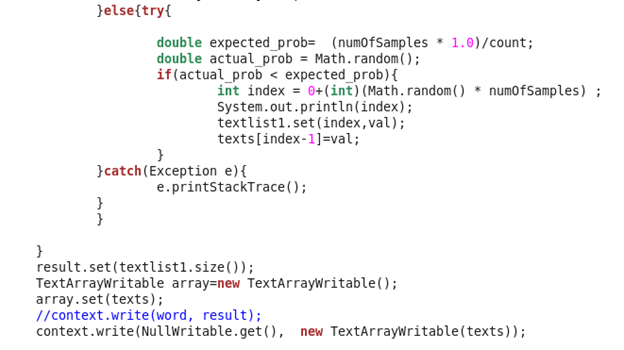
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Tasks** | **Sampling** | **Missing Value Handling** | **Outlier Handling** | **Feature aggregation/Reduction** | **Feature Transformation** | **Feature Selection** |
| **Krishna** | Hive | Python/MapR | Python | Python/MapR | Python | Hive |
| **Neethu** | R | R, Hive, Map Reduce | R | R | R | R |
| **Supraja** | MapReduce | Pig | Python | Pig | Pig or Python | Pig or Python |

## Sampling

**Sampling using Hive - Krishna Teja V**

The data was first loaded into a Hive Schema and then the first 45000 rows were used as training set and then the rest 14000 as testing set.

  
  
  
**Sampling using R - Neethu Sundar**Random sampling was done on the dataset, to obtain a random sample of 30k records  **Stratified Sampling using MapReduce - Supraja (Best):**Implemented Stratified Sampling using MapReduce , in the Map phase the dataset was grouped based on the Label and in the reduce phase the data from different strata was sampled using random sampling internally so that the distribution of data based on the labels remain the same .The output was a part-r-0000 file in the specified folder. Though Sampling was done as the dataset wasn’t large it was used to train and evaluate the model.  
Mapper Code :  
   
Reducer Code: 

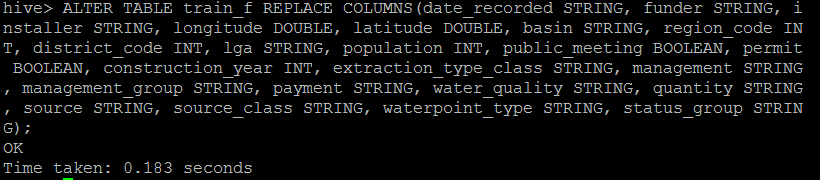


The stratified sampling method was chosen as it was intuitively the best approach for our dataset and it performed the best among all the others techniques. The base accuracy after sampling was 74% with Random Forests and we worked our way up from there. We chose Random Forests as it performed the best out of all the models owing to the mixed nature of our data and the noise in it.

## Feature Selection

**Hive - Krishna Teja V**

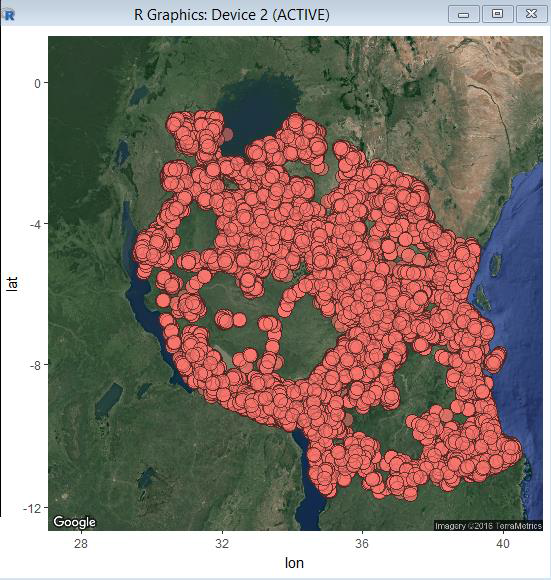
I removed some of the features using the HIVE ALTER TABLE command due to them being duplicate values, >70% missing values and some of them being the same value in the entire column.



After dropping these columns the accuracy increased to 75.4% because the model did not have to worry about all the missing values and inconsistent values present in the dataset. **Feature selection using R -Neethu Sundar**

A list of features were dropped from the dataset due to the following reasons.

1. All these features represent the geographical location of the water pump : Latitude, Longitude, Sub Village, Region code, District code, Lga, Ward

Latitude and Longitude – The location of the pumps differ only by small differences in the longitude and mostly confine to the same region.  
  
Subvillage – They represent a detailed location inside each region, which is too much of an information towards the problem.

## Region code and District code- contain numerous faulty data. For example, the same region has different region code.

## Lga and Ward - They represent the constituency of each region, which deviates from the problem.

## 2. “Num Private” is a numerical feature in the dataset with more than 95% of its value being “0” and has no metadata in the problem statement.

3. “Recorded by” has the same value repeated throughout the dataset.

## 

**Evaluation** : The feature selection removed highly correlated variables which improved the accuracy score when run on the Decision Tree Classifier model to **71%.**

.

**Python -Supraja Suresh**

Unimportant columns  were dropped using data frame drop method in python .These columns were intuitively removed as they represent name **wpt\_name** which is unique for each row and **scheme\_name** was same for all records as with **num\_private.**



## Missing Value Handling

**Python/MapR - Krishna Teja V**

Used the fillna() function of Pandas in Python notebook to fill the null values of permit and public\_meeting values to “Unknown” and the null values of construction\_year and population(the two features with the highest number of missing values among the features left) to 0.

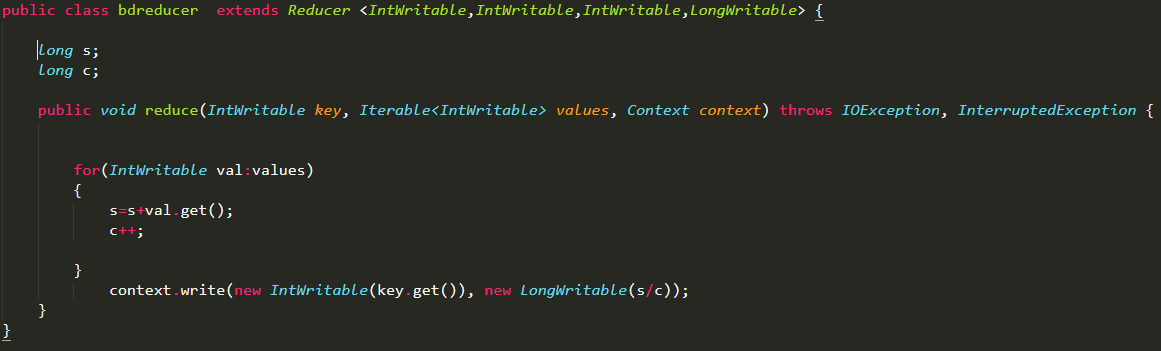


Intuitively, I thought the population will be related to a geographic feature and that the construction year will be dependent on the installer who installed it in the first place. Thus I wrote a MapReduce Program to group the construction year by it’s installer and the population by the district code and calculate their averages.

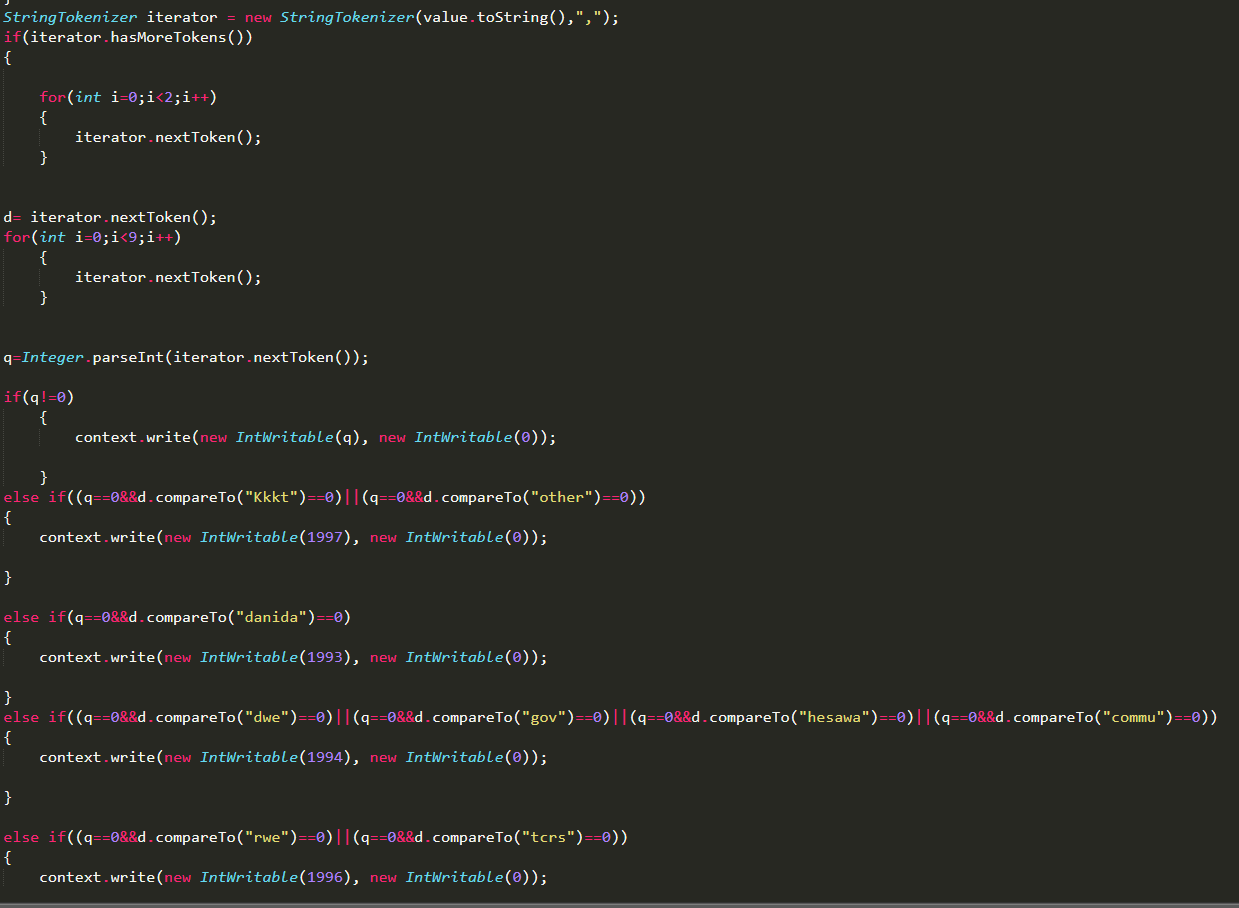
Mapper program:-

### 

Reducer program:-

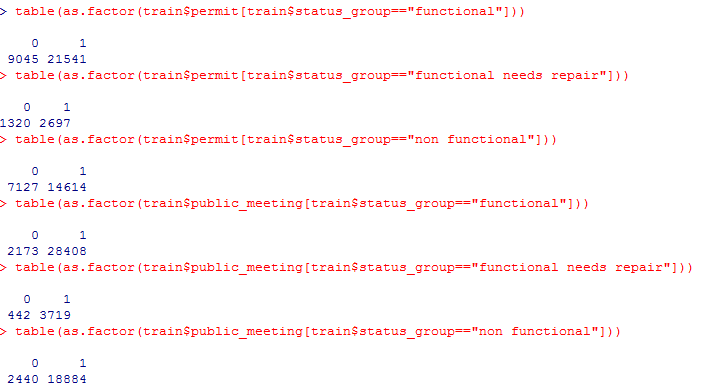


I then inserted these averages into the dataset in place of the missing values using a Mapper program



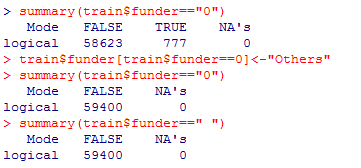
**Handling missing values - Neethu Sundar**

There are two features namely “Permit” and “Public Meeting” which have Boolean values. The number of TRUE and FALSE instances for each of these features were computed against the target class. And the maximum was used to replace the missing value.



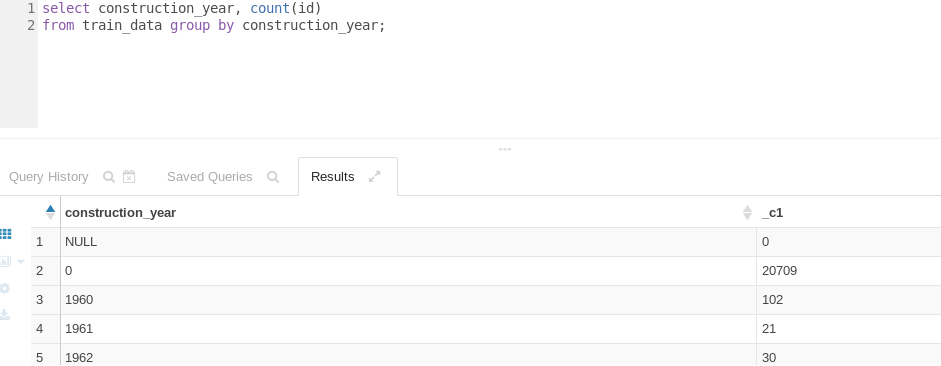
**Handling missing value using the R Language**

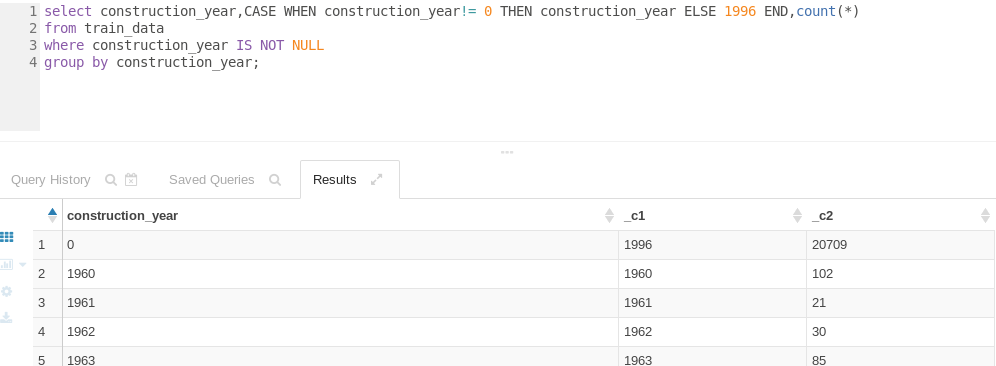
The feature “Funder” is a categorical field containing names. A summary of number of 0’s is taken and then the 0’s are replaced with the category “Others”.



**Handling missing values using Apache Hive.**

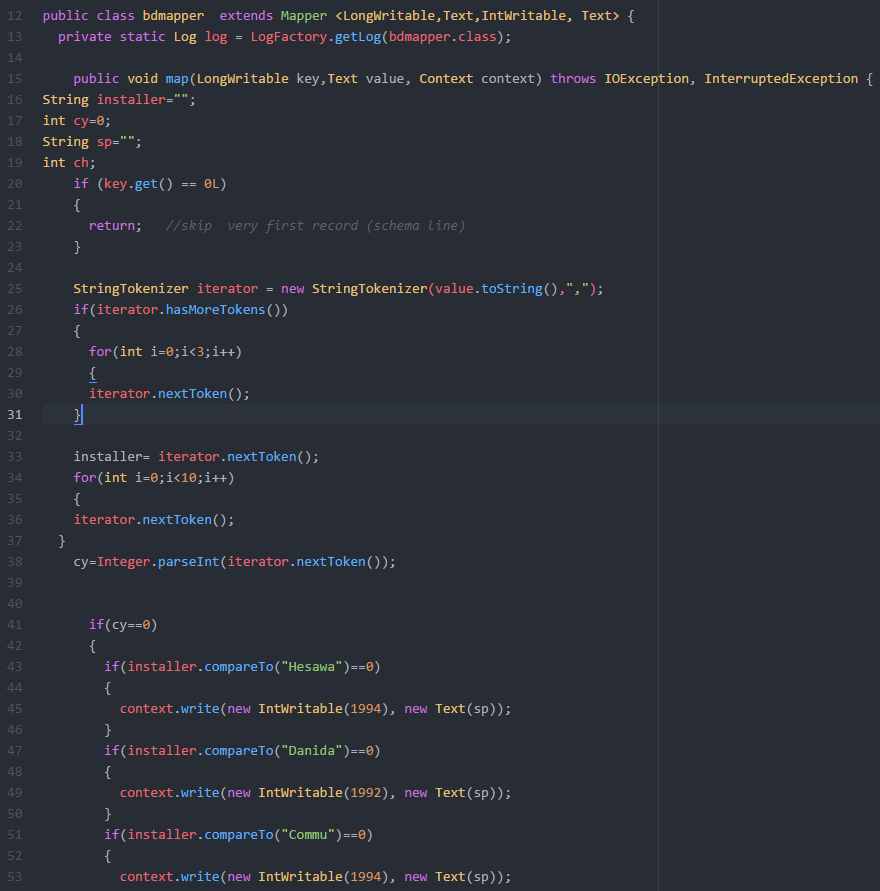
Construction Year is a numerical field containing the value of a year. Since year cannot be 0, it is considered as a missing value. 0’s are replaced with the average value of construction year. Similarly, Population is an integer field ranging from 0 to 30500. Since the population cannot be 0, it is considered as a missing value. The missing values are replaced with the average value computed grouping “district” wise.





**Handling missing value using MapR**

Construction Year is a numerical field containing the value of a year. Since year cannot be 0, it is considered as a missing value. The average value are computed grouping “funder” wise. A mapper-reducer program replaces the 0 values with the averages based on the “funder” value.

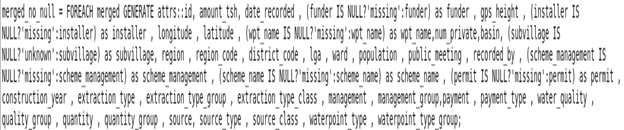


**Evaluation**: After the missing values in the dataset were handled, the Decision Tree Classifier model had a moderate increase in its accuracy score to **72%**

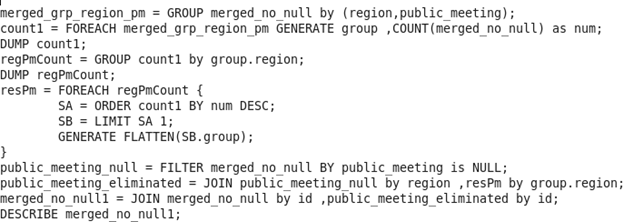


**Handling mising values using Pig - Supraja Suresh**

The missing values in the fields Funder ,installer, wpt\_name, subvillage, scheme\_management, scheme\_name ,  permit   - are replaced with the term “Missing ” using Pig



The blanks in Public\_meeting column is filled by grouping the records based on region and then the mode of the Public\_meeting for that region is used.



This series of modifications provided the **best results** and the accuracy of Random Forest model increased from **75% to 78%** after applying the changes construction year , population, permit and public meeting by value after grouping by region code and hence these was chosen for our dataset.

## Feature Aggregation/Reduction

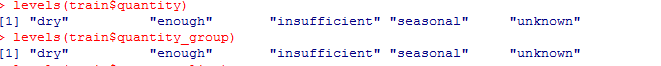
**Python - Krishna Teja V**

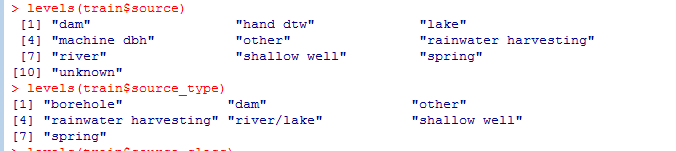
The installer and funder columns were studied with the Python function value\_counts() which lists the various values along with their counts. There were hundreds of values for each column but they were all top heavy. Thus I loaded the dataset into a Python pandas dataframe and converted the string values of installer for the top 10 values and all the other values were mapped into “other”.

Similarly, the funder column was reduced from thousands of values to 10. The accuracy went up from 78% to 79% after this step hence was included in the final dataset.

**Feature Aggregation - Neethu Sundarprasad:**

* The features “waterpoint type group” and “waterpoint type” are categorical input variables representing the same set of categories.
* The number of records under each category matched exactly for the two features, hence it is aggregated. Similarly, the features “source” and “source type” are aggregated. Also, the features “water quality” and “quality group” , “payment type” and “payment”, “management group” and “management”, “extraction”, “extraction type group”, “scheme name” and “scheme management” are aggregated.



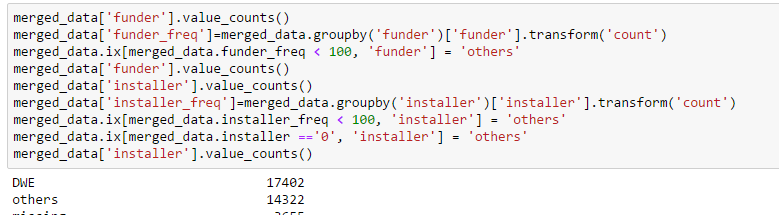


**Evaluation**: The feature reduction and aggregation steps of data pre-processing removed most duplicate features in the dataset. There was not any significant change in the accuracy score of the model.

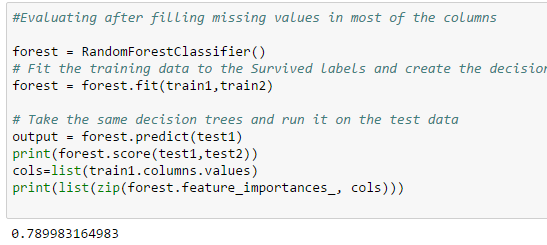
**Python - Supraja Suresh:**

**Feature Reduction using Python -Supraja Suresh**

The **funder and installer** had a variety of categorical attribute , so the number of unique values was reduced by taking those values that **occur less than 100 times** and replacing it with the term ‘others’.



But we chose the method where only 10 of the values was used as this decreased the accuracy value and the number of unique values were more .The accuracy obtained as is below



**Feature Aggregation using Python -Supraja Suresh**

There are many features groups that can be reduced / combined to one feature as they are highly correlated and provide no new value and these feature groups and the feature chosen out of it as mentioned above are :

'quantity','quantity\_group' -> quantity

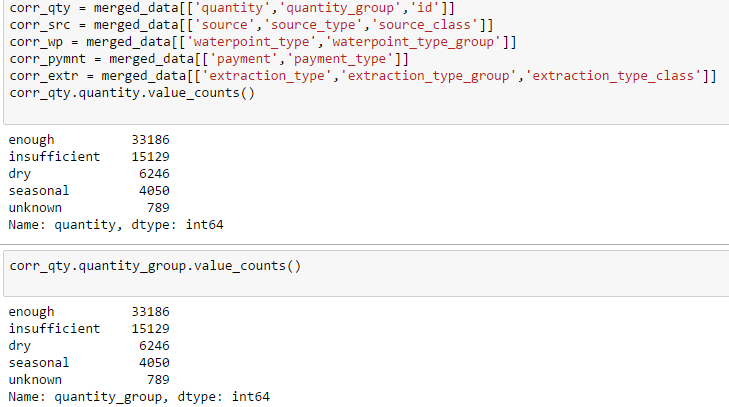
'source','source\_type','source\_class' -> source

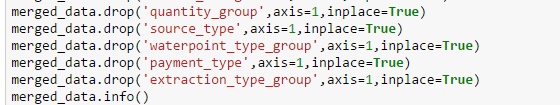
'waterpoint\_type','waterpoint\_type\_group' ->'waterpoint\_type'

'payment','payment\_type' -> 'payment

'extraction\_type','extraction\_type\_group','extraction\_type\_class' -> extraction\_type', extraction\_type\_group

For Example , quantity and quantity\_group has the same values and count .Say 33186 rows had enough as the quantity and same number had enough as quantity group. So took the lowest feature , in this case it is quantity.





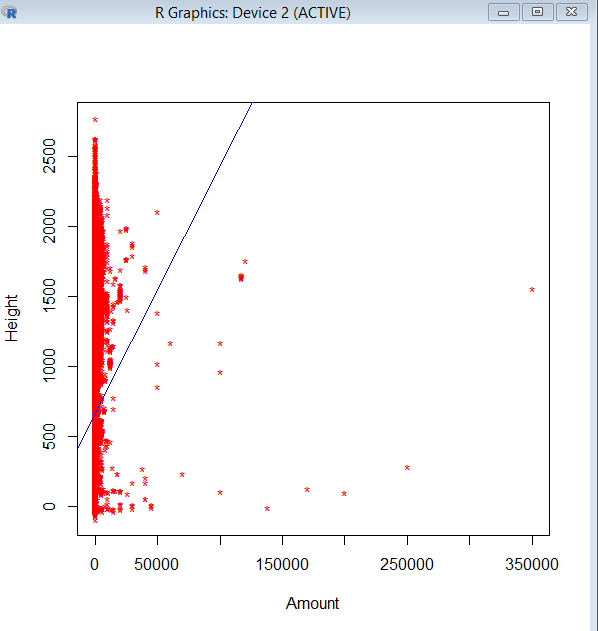
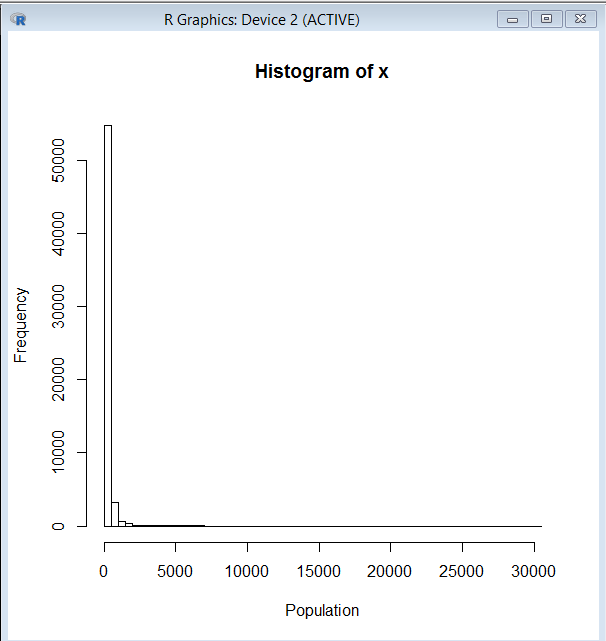
This step maintained the accuracy at 79% so this was included in the combined dataset.

## Outlier Handling

**Using R - Neethu Sundarprasad**

Population had outliers with few data points being sparsely distributed and not in a continuous range. These records contributed only to a small percentage of the dataset and thus were deleted.

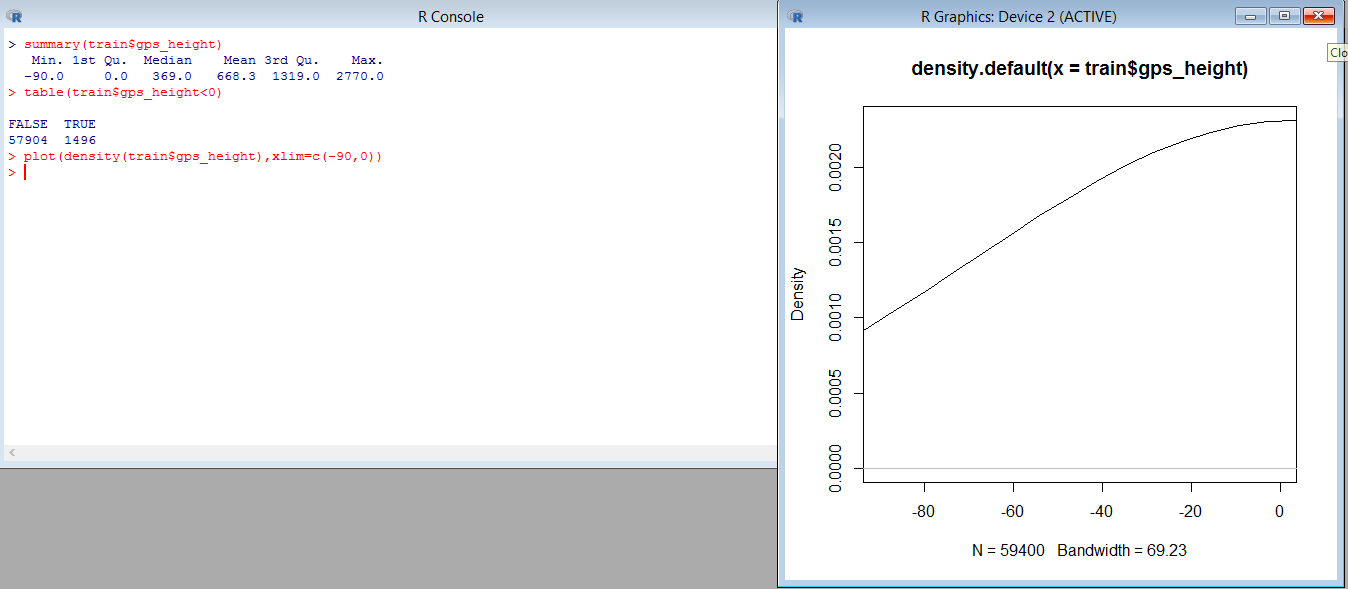
Amount TSH had values far beyond the mean value and those data points were outside the boundary region. These records were deleted.



“GPS Height” is a feature that represents the height of the well. However, it contained negative values.

These negative values contributed to only 2% of the dataset.

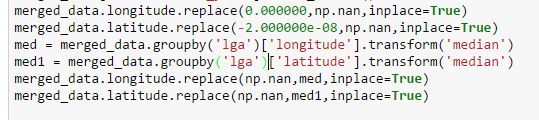
Hence all negative values were levelled to the value “0”.



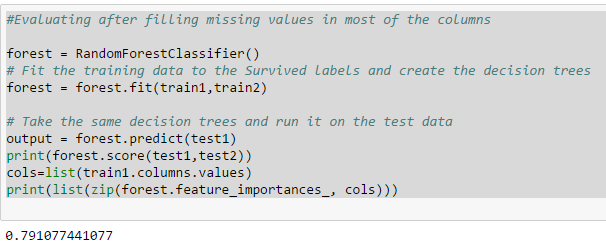
**Evaluation**: Because there were very few records holding outlier values, there was no consequential change observed in the accuracy score of the model.

**Python - Supraja**

**Latitude and Longitude** values pointed to other region outside Tanzania and in the Atlantic Ocean , these as an alternative approach can be intuitively considered as outliers .Many rows had such data and it wasn’t feasible to remove those records , so these outliers were handled by finding the **median** of the Latitude and Longitude after grouping it based on **lga** and then it was used to replace the outliers



After handling outliers when the model was evaluated using Random Forests the accuracy improved to 79.1%



## Feature Transformation

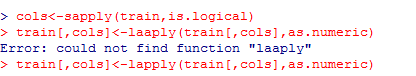
**Using R- Neethu Sundarprasad**

Construction Year represents the year the water pump was constructed.

In order to provide a meaningful correlation, using construction year the “Age” of the pump could be derived. Therefore, the feature “construction year” was transformed into “Age” with numerical input values.

Similarly, Date Recorded is a value containing the date when the observation was made. Since these dates stretch across various years and months, a new feature called “days since recorded” is derived. Days since recorded contains numerical values of how many days was it since the observation was made. Also, The features “Permit” and “Public meeting” are categorical variables in the dataset containing TRUE, FALSE and OTHER as values. These features are better processed if transformed as binary values.

Therefore they are transformed into numerical values containing 0’s and 1’s.

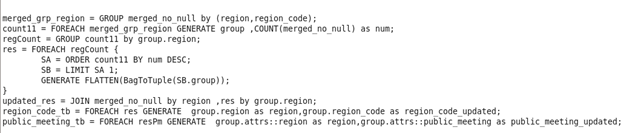


**Evaluation** : Feature transformation provided to the dataset more meaningful features. This process improved the accuracy close to **74%** in the Decision Tree Classifier model.



**Pig - Supraja Suresh**

Same **region had different region-codes** , so this was rectified by **grouping the regions** and then using the **most frequent region code** as the correct one ,it was identified as an incorrect value by cross verifying the region\_code from a reliable websource .



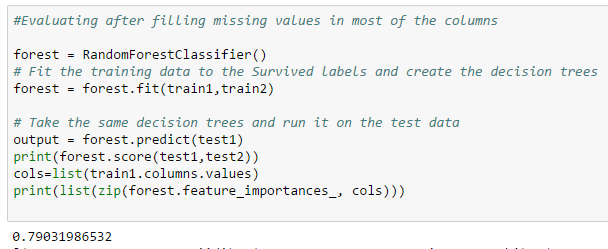
**Python - Supraja Suresh**

The other transformation that was done by finding the **median of the construction year** and used it instead of 0 and splitted into **bins** using python as below :

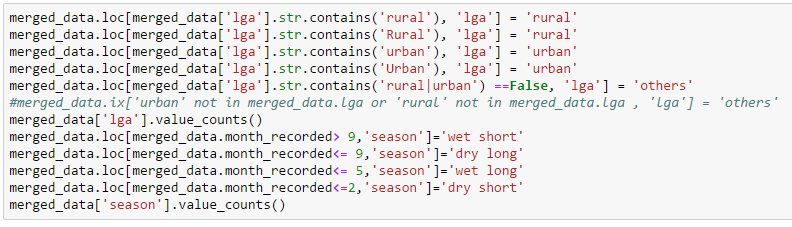
*merged\_data['construction\_year'].replace(to\_replace=0, value=merged\_data['construction\_year'].median(),inplace=True)*

*merged\_data['construction\_year'] = pd.cut(merged\_data.construction\_year, bins=10,labels=False)*

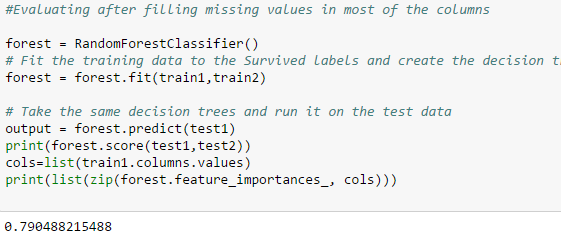
When this change was evaluated we found that this slightly decreased the accuracy to 79.03% .



**Lga** had geographical regions and this was split as **rural , urban and others** .If the value contained the term rural it was considered as rural , if it was urban then it was replaced by urban , if none of the above then it was replaced by ‘others’

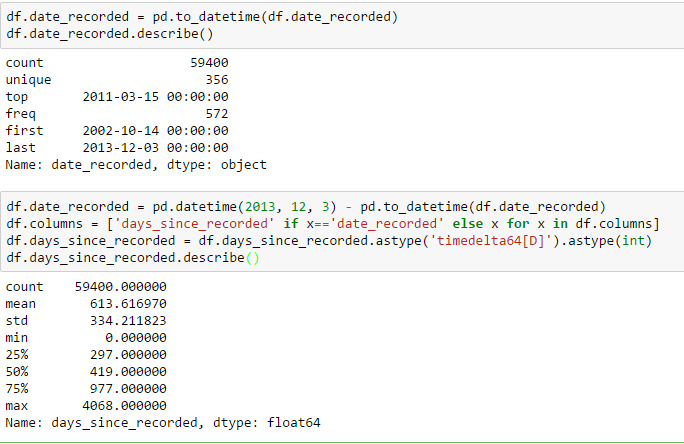


Based on the month recorded I created a new feature ‘**season**’ which would decide the season during that month as one of **‘wet short’,’dry long’,’wet long’ and ‘dry short’**.

This increased the accuracy by .01% from the previous.

**Python - Krishna Teja V**

The date\_recorded column was not in the standard date-time format. So I used Python Notebook environment to transform it into the proper format. To make it more relevant to the data set I subtracted the latest date recorded from the rest to result in a new column days\_since\_recorded as I believe it shows a better and relevant meaning than the previous feature, the idea being that the latest recorded waterpoint is more likely to be functional. Thus the date\_recorded column was transformed into days\_since\_recorded.



Overall Evaluation:

The Random Forest model reached **80.3% after this final step** on the final dataset that contained the best feature engineering results obtained by each of the members. Thus we are finished with the pre- processing part and the dataset obtained from all these contributed transformations by all the members of the group was used to test and tune the various Machine Learning Models using Python’s SKLearn libraries.

# 

# 

# **Model Selection & Tuning**

## Overview of Work

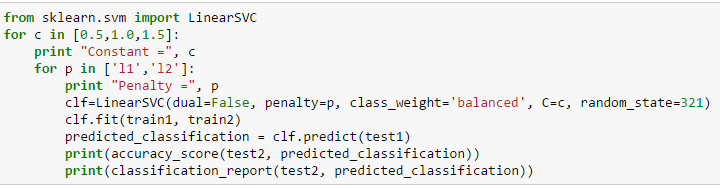
|  |  |  |
| --- | --- | --- |
| **Member** | **Algorithms Implemented** | **Breadth Covered** |
| Krishna Teja V | Gradient Boosted Trees, Adaboost, Linear SVC with OvO | Multi-class, Linear model with regularization, Boosting |
| Neethu Sundarprasad | Naive Bayes Decision Tree Logistic Regression | Gaussian model,Multinomial model,Smoothing, Bagging,One versus All, Classification tree, Regressor tree Solver, Multi class |
| Supraja Suresh | Random Forests, KNN, SGD,  One vs Rest | Model averaging, Lazy learning,Linear model with regularization,Multi-class |

## Model 1 - Linear Model with Regularization - SVC with OvO - Krishna Teja V

For the first model, I chose a simple linear model in SVM because among the linear models this is effective in high dimensional spaces. I knew it would not perform that good as our data is a mix of numerical and categorical features and are not continuous. Nevertheless, LinearSVC() will provide a baseline to compare the other models.

Firstly, I applied the model by varying the Constant C, The C parameter tells the SVM optimization how much you want to avoid misclassifying each training example. For large values of C, the optimization will choose a smaller-margin hyperplane if that hyperplane does a better job of getting all the training points classified correctly. Conversely, a very small value of C will cause the optimizer to look for a larger-margin separating hyperplane, even if that hyperplane misclassifies more points.

Next, the Penalty - l1 and l2 regularization are the available options. is the first moment norm |x1-x2| (|w| for regularization case) that is simply the absolute dıstance between two points where L2 is second moment norm corresponding to Eucledian Distance that is |x1-x2|^2 (|w|^2 for regularization case). Hence I run 2 loops on the Linear SVC model.



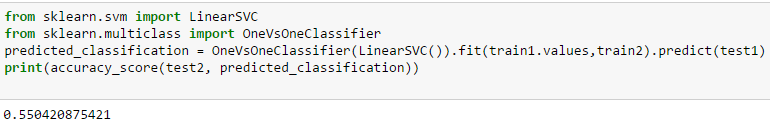
Running that, we get the following results:-

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | C = 0.5 | | C = 1.0 | | C = 1.5 | |
| Penalty = l1 | Penalty =l2 | Penalty = l1 | Penalty = l2 | Penalty = l1 | Penalty = l2 |
| Accuracy | 0.593 | 0.591 | 0.593 | 0.592 | 0.593 | 0.591 |

We observe that there isn’t much change in the accuracy when we change the hyperparameters. It also means that our dataset is fairly balanced and not too sparse or less sparse else there would have been a bigger difference between the l1 and l2 regularization.

Next, I added the One vs One Multiclass classifier to run on the best performing parameterized Linear SVC. One vs One Classifier uses a single binary classifier for every pair of classes to predict. So for our case the number of classifiers is 3.

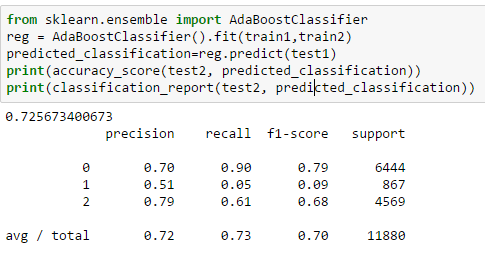
A surprising thing I noticed was that the accuracy actually dropped down to 55% from 59%



A possible explanation for this maybe that according to the classification report, the accuracy and precision for Class 1 (functional but needs repair) is very low (0.18). Thus when the classifier pairs Class 0 and 2 with it, the accuracy gets decreased overall.

## Model 2 - Adaboost - Krishna Teja V

The core principle of AdaBoost is to fit a sequence of weak learners (i.e., models that are only slightly better than random guessing, such as small decision trees) on repeatedly modified versions of the data. The predictions from all of them are then combined through a weighted majority vote (or sum) to produce the final prediction.   
I expected this model to perform better on our dataset than the previous model as our data itself is not continuous and has a fair bit of noise.



Right off the bat, the Adaboost Classifier with default parameters gives 72.5% accuracy which is way better than Linear Models.

Now for the tweaking part, the parameters that are available are n\_estimators and learning\_rate. N\_estimators is the number of weak learners that the model uses to train on and learning\_rate is a float value that decides how much to shrink the value of each classifier. Thus I ran the models with a range of values for each of the parameters.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | n\_estimators=250 | | n\_estimators=350 | | n\_estimators=450 | |
| learning\_rate=0.5 | learning\_rate=1.5 | learning\_rate=0.5 | learning\_rate=1.5 | learning\_rate=0.5 | learning\_rate=1.5 |
| Accuracy | 0.732 | 0.739 | 0.734 | 0.740 | 0.737 | 0.742 |

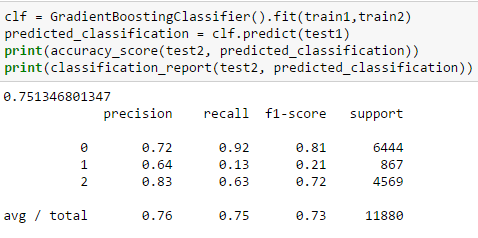
Thus we observe that with more number of estimators we get a higher accuracy as more the number of classifiers, more the accuracy of the model. However, learning\_rate follows an inverse relationship with accuracy.

The reason I think is that Adaboost overfits the data into the model. Thus when we put higher values in learning\_rate, it shrinks the importance of each classifier. Thus each classifier does not have the parameter to read and process more data and overfit it, thus resulting in higher accuracy.

## Model 3 - Gradient Boosted Classifier - Krishna Teja V

GBC is an ensemble method which produces a prediction model in the form of an [ensemble](https://en.wikipedia.org/wiki/Ensemble_learning) of weak prediction models, typically [decision trees](https://en.wikipedia.org/wiki/Decision_tree_learning). It builds the model in a stage-wise fashion like other [boosting](https://en.wikipedia.org/wiki/Boosting_(meta-algorithm)) methods do, and it generalizes them by allowing optimization of an arbitrary [differentiable](https://en.wikipedia.org/wiki/Differentiable_function) [loss function](https://en.wikipedia.org/wiki/Loss_function).

It handles mixed type of data very well (such as ours) and it is fairly robust to noise and outliers because of it’s loss functions. Thus I predicted this model to run the best on my dataset.



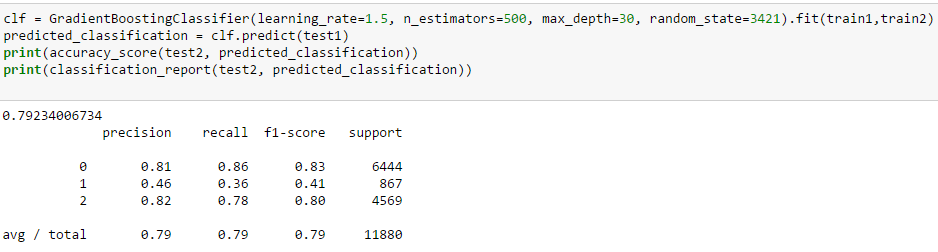
With the basic parameters we get an accuracy of 75.13%. The important thing to note here is that the precision for the 2nd class is higher compared to the other models. This is the hardest class to predict for the dataset and fine tuning the parameters I wish to improve it.

The parameters available are learning\_rate, n\_estimators (which we have previously seen in adaboost), max\_depth, warm\_start, max\_features, min\_sample\_split. Maximum depth of the individual estimators. The maximum depth limits the number of nodes in the tree. Thus this was the most critical variable, and it’s value depended on the number of inputs in the dataset. Warm\_start, when set to True, reuses the solution of the previous call to fit and add more estimators to the ensemble. Turning this to True provided a higher accuracy.

Max\_features is the number of columns to consider. The optimum value is found to be about 50% of the total number of columns. Min\_sample\_split is the minimum no. of values needed to make a split in a node. Higher values prevent overfitting so it was set to 2.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | max\_depth=10 | max\_depth=20 | max\_depth=30 | |
| n\_estimators=400, learning\_rate=1.5 | n\_estimators=500, learning\_rate=1.5 | n\_estimators=500,learning\_rate=1.5 | n\_estimators=500 ,learning\_rate=0.5 |
| Accuracy | 0.780 | 0.787 | 0.791 | 0.792 |

Thus, we observe that increasing the max\_depth value increases the accuracy greatly. Also n\_estimators also give a better performance but the tradeoff is the time it requires to run the mode, i.e, the computation time. The learning\_rate in this case did not really have an effect which means that the model is fairly safe from overfitting problem that occurred in Adaboost.



Thus Gradient Boosted Classifier with tuning performed the best out of all my models with accuracy of 79.23% and the highest precision/recall for Class 1.

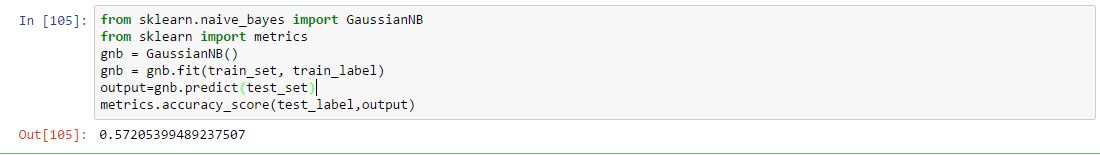
**SUMMARY OF EVALUATIONS**

|  |  |  |
| --- | --- | --- |
| Model | Accuracy | Recall |
| Linear SVC | 59% | 59% |
| Adaboost | 74.2% | 74% |
| Gradient Boosting Classifier | 79.23% | 79% |

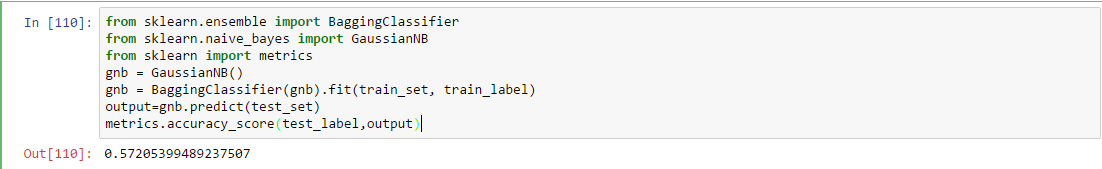
**Model 4: Naive Bayes classifier model**

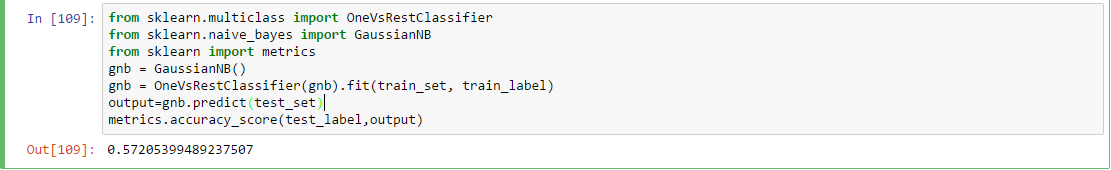
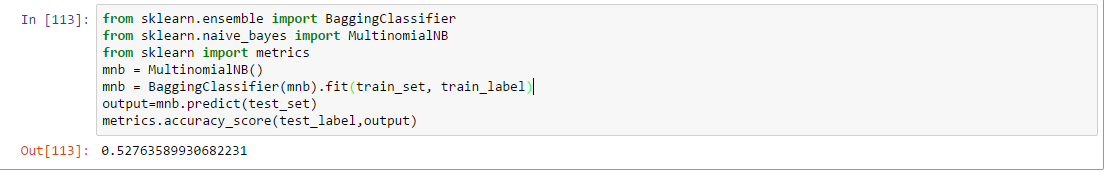
It works on Bayes theorem of probability to predict the class of unknown data set.Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature.There are three types of Naive Bayes model under scikit learn library:

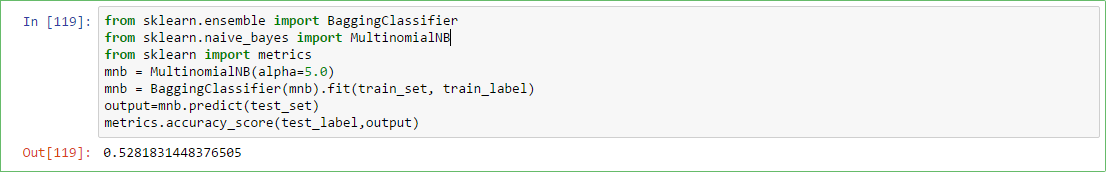
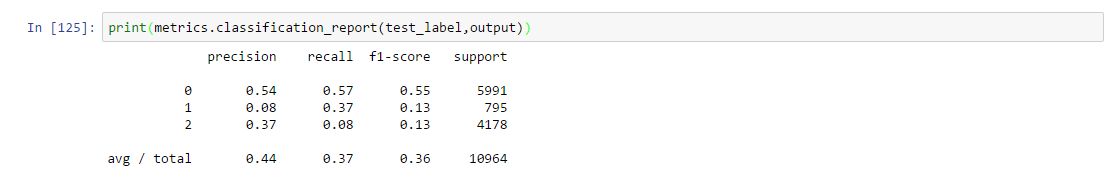
* [Gaussian](http://scikit-learn.org/stable/modules/naive_bayes.html)
* [Multinomial](http://scikit-learn.org/stable/modules/naive_bayes.html)
* [Bernoull](http://scikit-learn.org/stable/modules/naive_bayes.html)i

Applying the gaussian model without any tuning - ACCURACY SCORE : 57%  
Naive Bayes classifiers has limited options for parameter tuning like

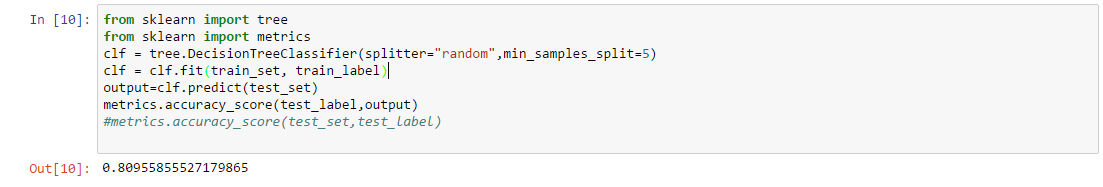
* alpha=1 for smoothing,
* fit\_prior=[True|False] to learn class prior probabilities

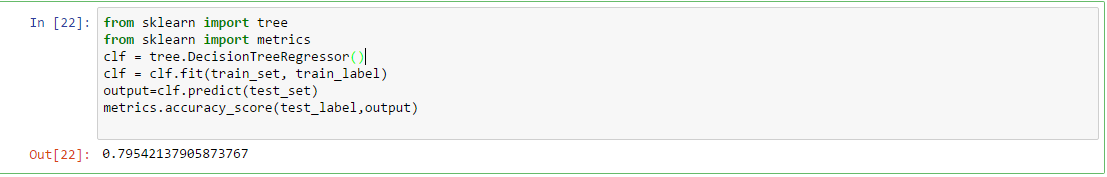
Applying some classifier combination technique like bagging we try to improve the model’s performance. However bagging won’t help since their purpose is to reduce variance and naive Bayes has no variance to minimize. Therefore the score remains the same.  
Applying Gaussian Naive Bayes model with bagging. ACCURACY SCORE : 57%

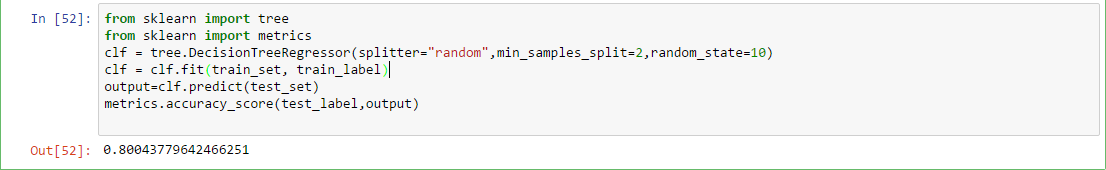
Binary Relevance methods such as the OneVersusRest classifier also does not improve the performance of the Gaussian Naive Bayes model.  
Applying the OneVsRestClassifier. ACCURACY SCORE: 57%  
i Multinomial Naive Bayes model It is used for discrete counts which observes the “number of times outcome number is observed over the n trials”.  
Applying Multinomial Naive Bayes mode. ACCURACY SCORE: 52%  
  
Trying to increase the value of smoothing, the accuracy score still remains the same.  
Applying the Multinomial Naive Bayes model with smoothing. ACCURACY SCORE: 52%

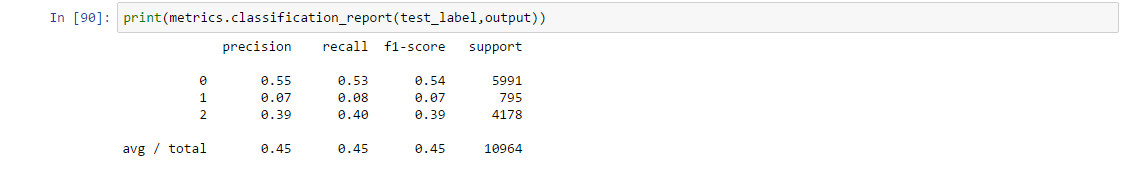
  
Evaluation of the Naive Bayes classifier:  
  
  
**Model 5: Decision Trees Classifier**Decision Tree Classifier has the following parameters to tune with:

* Splitter - The strategy used to choose the split at each node. “Best” to choose the best split and “random” to choose the best random split.
* Max\_features - The number of features to consider when looking for the best split
* Max\_depth - The maximum depth of the tree
* Min\_samples\_split-The minimum number of samples required to split an internal node
* Random\_state-The seed used by the random number generator
* Presort -Whether to presort the data to speed up the finding of best splits in fitting

Applying the decision tree model on the dataset- ACCURACY SCORE:73%  
   
Applying with tuning : ACCURACY SCORE : 80%  
Splitter - Random  
Max\_features -sqrt(n\_features)  
Presort- False/True  
  


Using a decision tree regressor classifier, improves the score.ACCURACY SCORE: 79%  


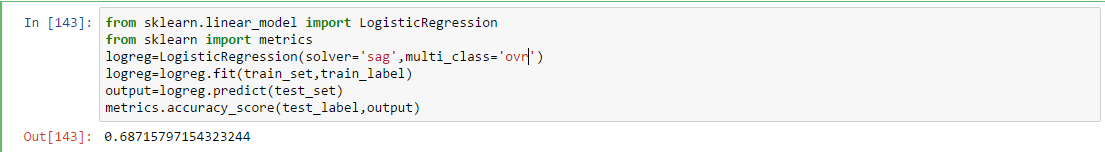
Tuning the parameters of decision tree regression model, ACCURACY SCORE: 80%  


Evaluation of the Decision Tree Classifier:  
  
  


**Model 6: Logistic Regression**Logistic regression is a [regression](https://en.wikipedia.org/wiki/Regression_analysis) model where the [dependent variable](https://en.wikipedia.org/wiki/Dependent_and_independent_variables) is [categorical](https://en.wikipedia.org/wiki/Categorical_variable).  
Applying Logistic Regression model - ACCURACY SCORE: 64%  
  
Some important tuning parameters of the Logistic Regression Model are:

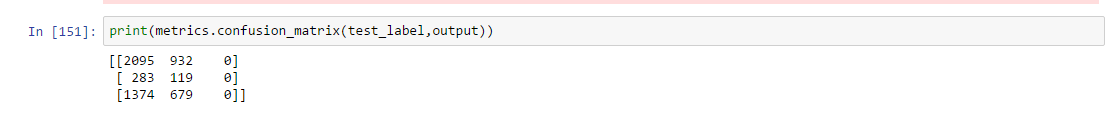
* Solver -Algorithm to use in the optimization problem.For small datasets, ‘liblinear’ is a good choice, whereas ‘sag’ is faster for large ones.
* Multi\_class -Multiclass option can be either ‘ovr’ or ‘multinomial’.

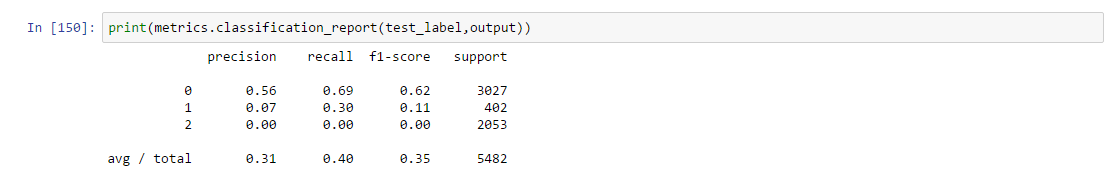
In the multiclass case, the training algorithm uses the one-vs-rest (OvR) scheme if the ‘multi\_class’ option is set to ‘ovr’  
Applying with tuning parameters solver =sag andS multi\_class= ovr - ACCURACY SCORE:68%



Evaluating the Logistic Regression model:







**SUMMARY OF EVALUATIONS**

|  |  |  |
| --- | --- | --- |
| Model | Accuracy | Recall |
| Naive Bayes | 57% | 57% |
| Decision Tree | 79.5% | 45% |
| Logistic Regression | 68% | 40% |

**Model 7-KNN**

K-Nearest Neighbour is an Instance based algorithm where we classify a datapoint as a particular class based on the nearest neighbour out of the k-nearest neighbors and summarizes the output for those k instances, in classification the mode of the class labels . The parameter k plays a major role if the right value of k is not chosen then it affects the class label. We can even give assign high weights to the instances that are nearby and less to those that are far away while predicting the class type of an unknown datapoint.

The various parameters that can be used are :

**n\_neighbors** : The number of nearest neighbours used to classify the query point

Default -5

**weights**:Determines the influence of the data points

Default-uniform

All points have equal weights, with ’distance’ as value the nearest points have greater influence

**algorithm**: the algorithm used to determine the neighbours

Default-auto

The other values that can be used are ‘ball\_tree’,’kd\_tree’ and ‘brute\_force’ .The latter causes memory error as it takes too much space to identify the nearest neighbours.

**Leaf\_size**:

Default-30

Used when the one of the tree algorithm is used , determines the speed and also the memory needed to determine the neighbours

**p:** Power parameter for the Minkowski metric

Default-2

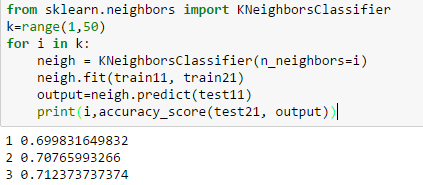
When p = 1, this is equivalent to using manhattan\_distance (l1), and euclidean\_distance (l2) for p = 2. For arbitrary p, minkowski\_distance (l\_p) is used.

**Metric:** Minkowski

**Minkowski distance** is a [metric](https://en.wikipedia.org/wiki/Metric_(mathematics)) in a [normed vector space](https://en.wikipedia.org/wiki/Normed_vector_space) which can be considered as a generalization of both the [Euclidean distance](https://en.wikipedia.org/wiki/Euclidean_distance) and the [Manhattan distance](https://en.wikipedia.org/wiki/Manhattan_distance), where 1 indicates manhattan distance and 2 indicates Eucledian distance

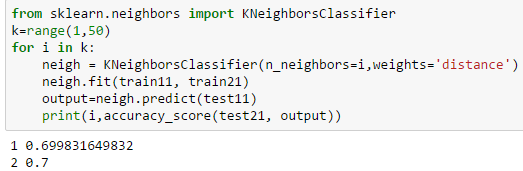
K neighbors1:

**1.Model with default setting except different k values**



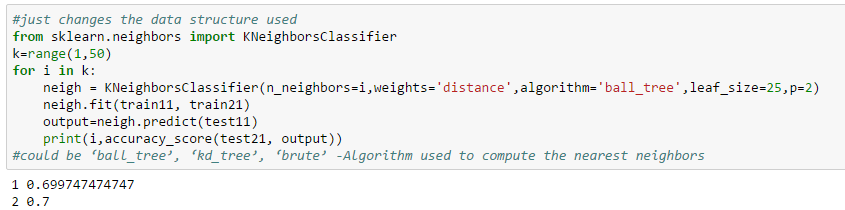
K neightbors2

**2.Tuning k value and using weights as distance**



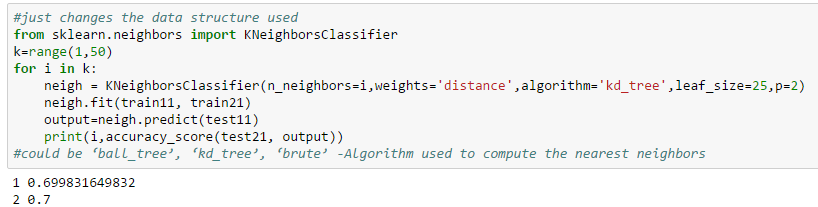
K neightbors3

**3.Tuning k,weights as distance, ball\_tree algorithm, leaf\_size as 25 and using p=2**



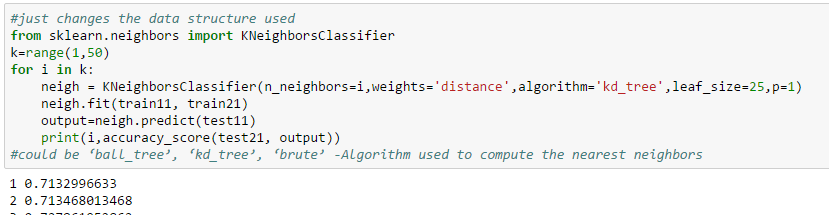
K neightbors4

**4.Tuning k,weights as distance, kd\_tree algorithm, leaf\_size as 25 and using p=2**

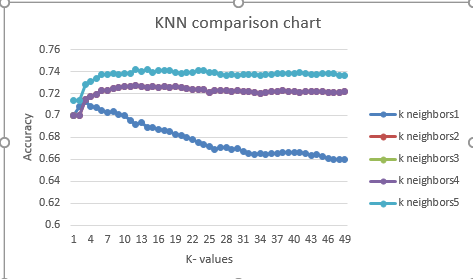


K neightbors5

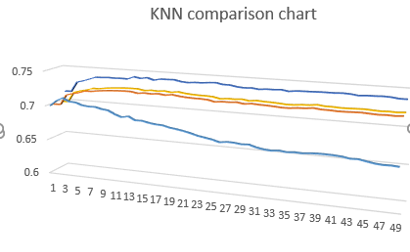
**5.Tuning k,weights as distance, kd\_tree algorithm, leaf\_size as 25 and using p=1**



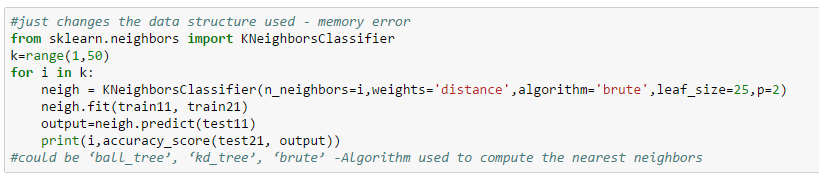
The results from the above 5 models are depicted in the graphs below:



To see without overlap the same chart in side view



It can be seen that the setting with K-neighbors 1 performs worse as k increases for the default setting with different settings for k.The series with kneighbors 2,3,4 are almost the same with little to no changes as the nearest neighbors are made to have more influence and Eucledian distance is used. The performance for the K-neighbors 5 with p=1 using Manhattan distance performance much better when compared to the others as it uses absolute difference than the Eucledian one.Thus this asserts us that the role of k is very important in determining the class label.

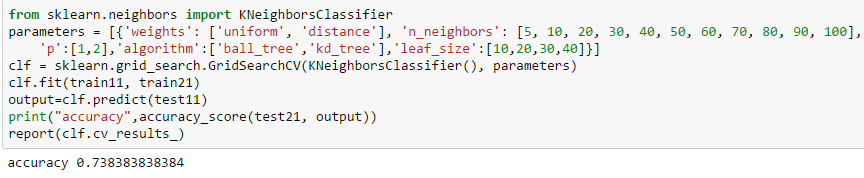
**6. With brute force algorithm** 

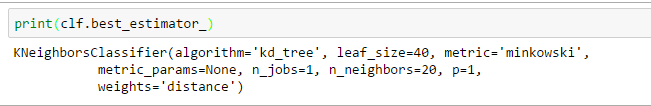
Got memory error as it uses trial and error method to determine the nearest neighbors

**7.Grid search :**

As it’s not practically feasible to go through all possible combinations we resort to Gridsearch which allows us to specify the hyperparameters we want to evaluate and gives a model with the best parameters based on crossvalidation results.

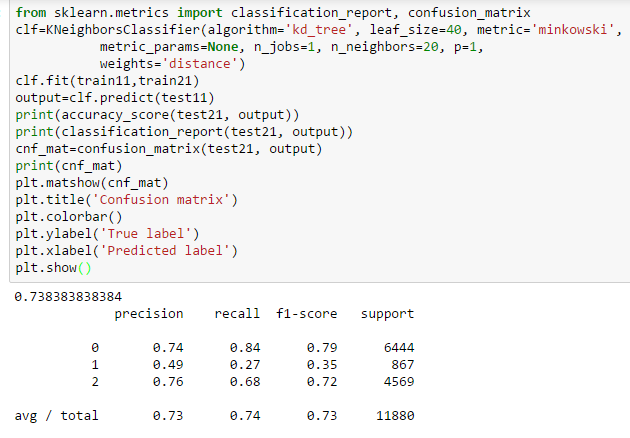
Grid search shows that the assumption p=1,algorithm=kd\_tree (from the graph)and n\_neighbors =20 we made through random evaluation from the above models is correct and it can be seen from the result in grid search.





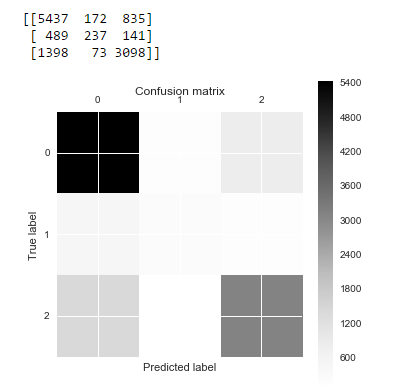
**Best Model Metrics:**

The best model obtained from Grid Search is as below and the accuracy obtained is 73.8% as shown below .The classification report is also depicted in the below figure.



The recall for the class functional but needs repair is low and below 50% but it does well for predicting functional class and mediocrely for predicting non-functional class.

**Confusion Matrix**:The confusion Matrix shows that the model predicts most of the functional but needs repair and non functional pumps as functional. This is a major drawback , but atleast it can be used for identifying if the datapoints are functional or non functional in general.



KNN works well with a small number of input variables (p), but struggles when the number of inputs is very large.Since the input is not large and number of features are low k-nn performs averagely.But doesn’t yield the best results.

**Model 8 - Random Forests**

A random forest is an ensemble algorithm that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to determine the output class labels. The sub-sample size is always the same as the original input sample size but the samples are drawn with replacement if bootstrap=True.

Parameters used are:

**n\_estimators :** The number of trees in the forest

default=10

**criterion** : measure the quality of a split

default=”gini”

criteria are “gini” for the Gini impurity and “entropy” for the information gain. Note: this parameter is tree-specific.

**max\_features** : number of features to consider when looking for the best split

default=”auto”.

If “auto”, then max\_features=sqrt(n\_features).

If “sqrt”, then max\_features=sqrt(n\_features) (same as “auto”).

If “log2”, then max\_features=log2(n\_features).

If None, then max\_features=n\_features.

*To experiment the model was tested on all features from 1 to total number of features to compare the results.*

**max\_depth** : The maximum depth of the tree

default=None- nodes are expanded until all leaves are pure or until all leaves contain less than min\_samples\_split samples.

**min\_samples\_split** : minimum number of samples required to split an internal node

default=2

**min\_samples\_leaf** : The minimum number of samples required to be at a leaf node

default=1

**max\_leaf\_nodes** : int or None, optional (default=None)

Grow trees with max\_leaf\_nodes in best-first fashion. Best nodes are defined as relative reduction in impurity. If None then unlimited number of leaf nodes.

**bootstrap** : boolean, optional (default=True)

Whether bootstrap samples are used when building trees.

**oob\_score** : Whether to use out-of-bag samples to estimate the generalization accuracy default=False

**n\_jobs** : The number of jobs to run in parallel for both fit and predict

default=1

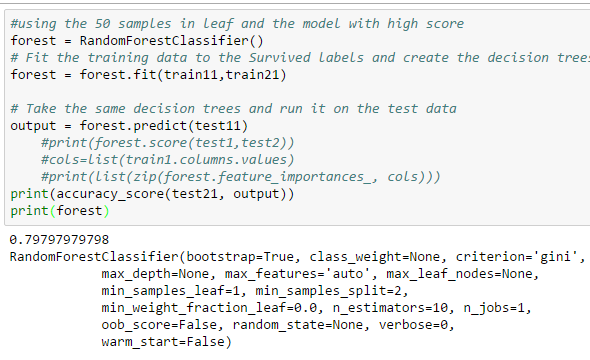
-1, then the number of jobs is set to the number of cores.

**class\_weight** Weights associated with classes in the form {class\_label: weight}. If not given, all classes are supposed to have weight one.

default=None

The “balanced” mode uses the values of y to automatically adjust weights inversely proportional to class frequencies in the input data as n\_samples / (n\_classes \* np.bincount(y))

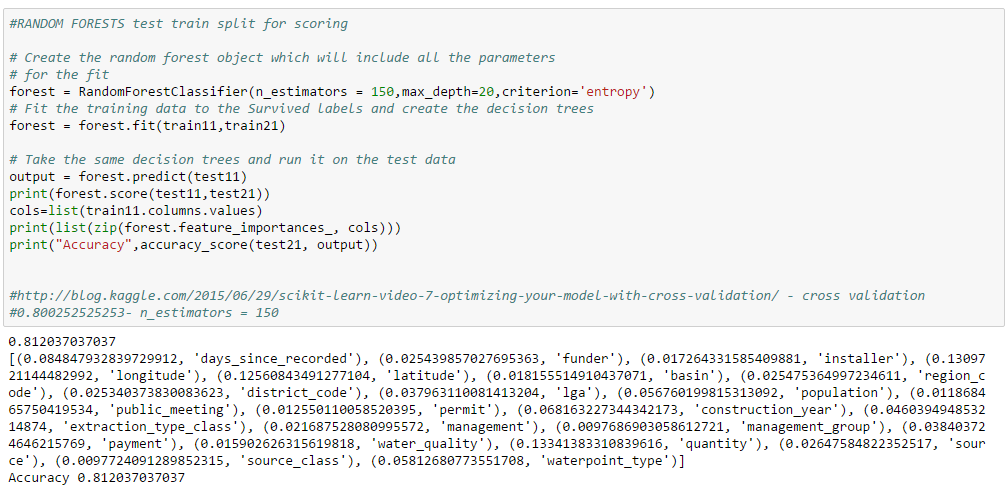
**Default classifier**



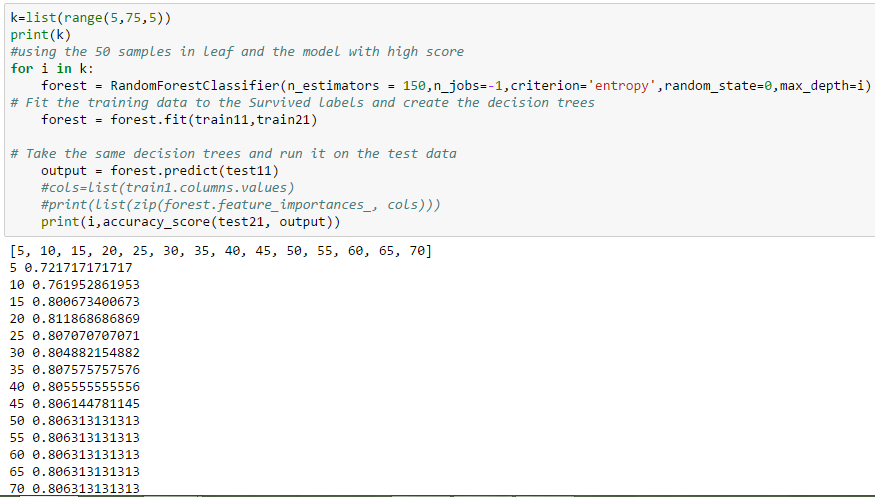
The classifier as such without parameter tuning produces accuracy of around 79.7%

1**. Changing N estimators,max\_depth and criterion:**

When the number of tree is set to 150 , and the depth of the tree is set to 20 and the criterion is entropy rather than the default gini impurity , the accuracy of the model improves and this is because as the number of trees increases the prediction is better as the average of those trees are considered as the final output for the datapoint.The depth indicates the height of the tree and at a level there can be more than one feature based decision and thus as the depth increases the accuracy improves.Gini measurement tries to improve the probability of a random sample being classified correctly if we randomly pick a label according to the distribution in a branch where as Entropy is a measurement of uncertainity or randomness and it tries to reduce the entropy and increase the information gain from a split .Hence Entropy performs better .

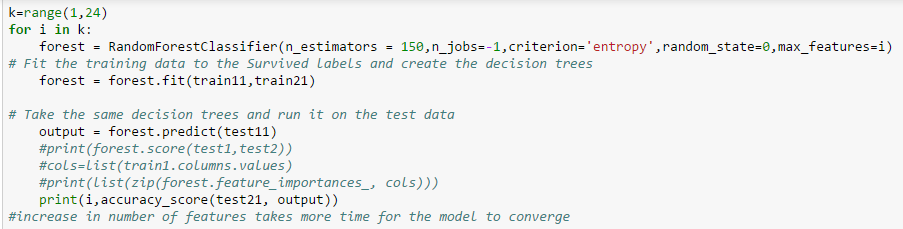


**2. Random forests with entropy and varying max\_depth and max\_features**



It can be seen that with max depth 20 we get higher accuracy for the above given parameters.Specifying the right depth is important otherwise the data might be underfitted if the depth is too low and overfitted if the depth is too high. The trees need to grow inorder to predict a correct classification.

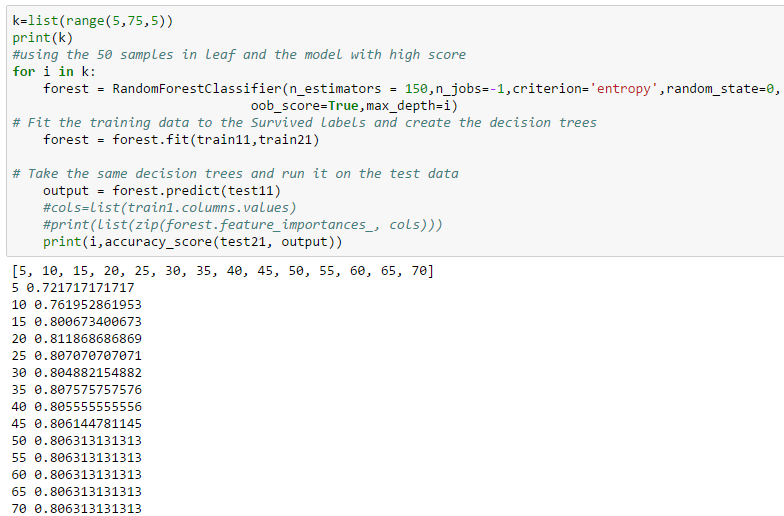
For varying max\_features :



|  |  |
| --- | --- |
| max\_features | accuracy |
| 1 | 0.799832 |
| 2 | 0.804714 |
| 3 | 0.802862 |
| 4 | 0.806313 |
| 5 | 0.804209 |
| 6 | 0.804377 |
| 7 | 0.80463 |
| 8 | 0.804966 |
| 9 | 0.805135 |
| 10 | 0.804882 |
| 11 | 0.803956 |
| 12 | 0.804125 |
| 13 | 0.803114 |
| 14 | 0.805724 |
| 15 | 0.803872 |
| 16 | 0.803788 |
| 17 | 0.80303 |
| 18 | 0.802946 |
| 19 | 0.804714 |
| 20 | 0.802525 |
| 21 | 0.805051 |
| 22 | 0.802778 |

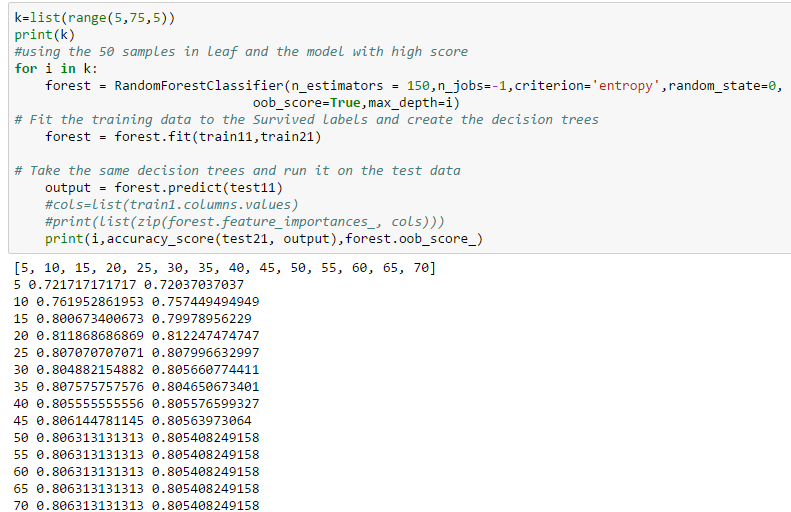
With as low as 4 features we get better accuracy , which can be used for further analysis

**3.Random forests with entropy and varying max\_depth and oob(out of bag samples set to true)**

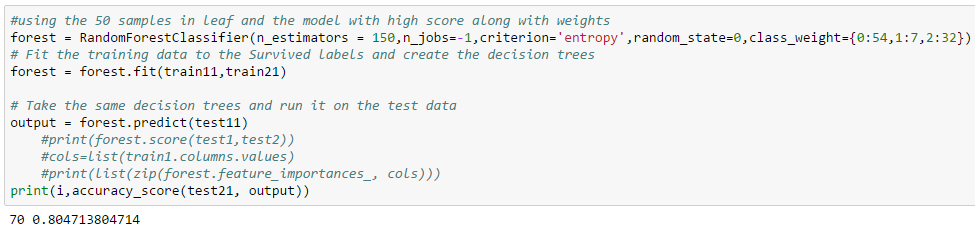


The accuracy remains the same when out of bag samples are considered , instead of measuring accuracy considering the oob\_error or score is a better estimate as it estimates the error rate of the out-of-bag classifier on the training set (compare it with known labels).

**out-of-bag examples** - After creating the classifiers (S trees), for each (Xi,yi) in the original training set i.e. T, select all Tk which does not include (Xi,yi). This subset, is a set of boostrap datasets which does not contain a particular record from the original dataset. This set is called out-of-bag examples. OOB classifier is the aggregation of votes ONLY over Tk such that it does not contain (xi,yi). the out-of-bag estimate is as accurate as using a test set of the same size as the training set. Therefore, using the out-of-bag error estimate removes the need to set aside test set.

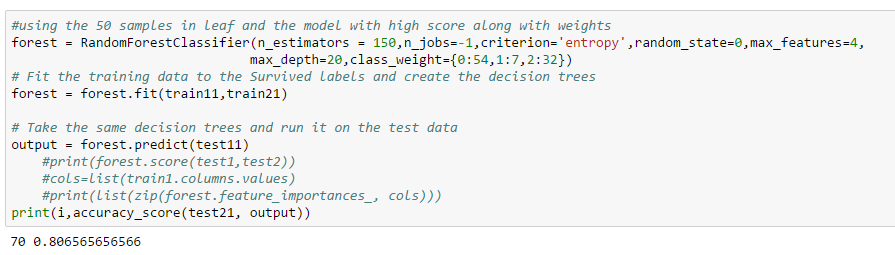


**5.Using Class weight**



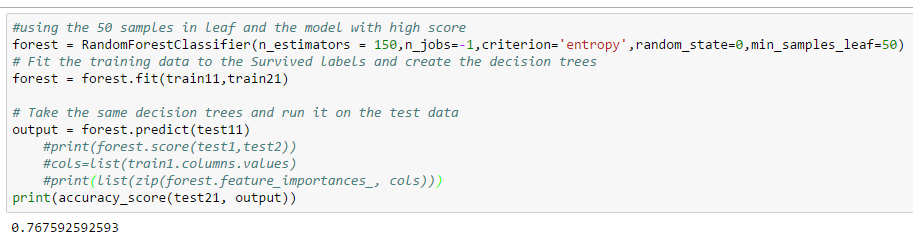
If the class weights are included then the weightage of the results changes, the class that has low frequency of occurance has higher weightage , thus functional but needs repair class has higher weightage but many of the records with this label is misclassified and hence the accuracy decreases as the importance changes.

**6.Using 4 features(almost equal to the sqrt(no of features)) ,class weight and max depth as 20 (as it gave better results in the above runs)**



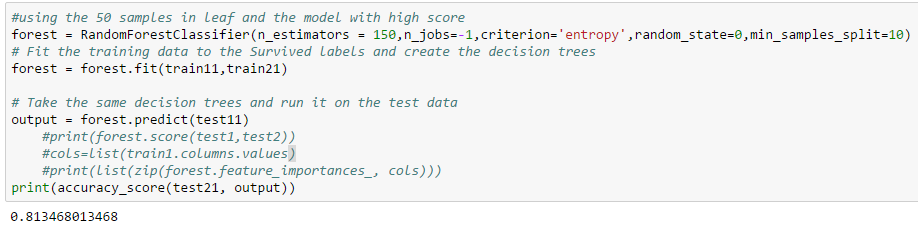
Class weights are given based on the ratio between the classes functional, functional but needs repair and non- functional in the original dataset. If the class weights are included then the weightage of the results changes and hence the accuracy decreases by around 0.4 % from 81%

**7. Tuning the min\_samples\_leaf**



As the min\_samples\_leaf parameter is set to 50 from default 3 , it expects atleast 50 nodes to be the leaf and hence it loses fine grained information and can underfit the data. As this is changed the tree construction is changed from the default and hence the accuracy is reduced to 76%.

**8. Tuning min\_samples\_split**



Here changing the min\_samples\_split from 2 to 10 helps in generalizing as random forests tend to overfit the data and hence changing the minimum samples required to split the internal node helps in slightly generalizing and avoids overfitting to some extent.

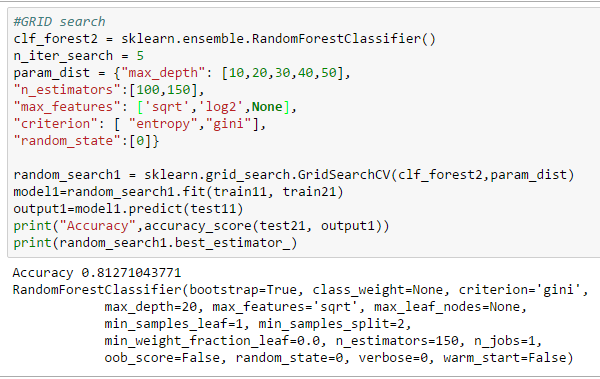
**9.Random search**



**10.Grid Search**

The results from random and grid search are different , because Grid search tries every combination of a preset list of values of the hyper-parameters and chooses the best combination based on the cross validation score, whereas Random search checks for random combination of hyperparameters and the latter doesn’t guarantee a combination of best parameters.

Thus Random search gives a model with accuracy 80.45% and Grid Search gives a model with Accuracy 81.2% as shown above.Also shows that Gini index is the better criterion for splitting , the rest of our assumption that n\_estimators =150, max\_depth=20 with oob\_score false is correct.Also the assumption that max\_features that is around 4 which is correct for best results from the below as it chooses the option ‘sqrt’.

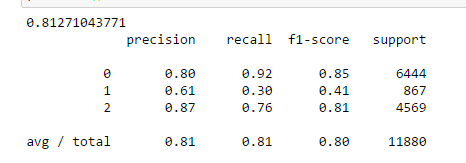


**Best Model metrics :**

The best model according to grid search is as shown above and the metrics of the model are

Accuracy - 81.27%

Classification Report:



It can be seen that the recall for the functional class is as high as 92% but that of functional but needs repair is 30%, the model identifies the nonfunctional class with recall 76%. This is much better when compared to that of KNN.

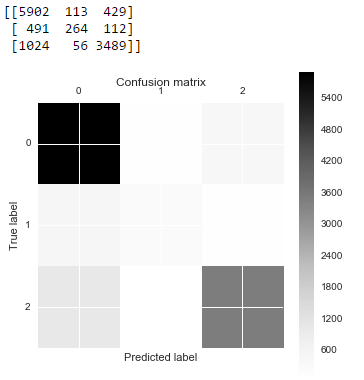
**Confusion Matrix:**

It can be seen that Random Forest predicts all classes correctly to some extent, it lacks correctness in predicting functional but needs repair class majorly and non functional as it classifies them to be functional in many cases.

0- functional

1- functional but needs repair

2-non functional



## Model 9-SGD classifier

SGD classifier is a simple yet very efficient approach to discriminative learning of linear classifiers under convex loss functions such as (linear) [Support Vector Machines](https://en.wikipedia.org/wiki/Support_vector_machine) and [Logistic Regression](https://en.wikipedia.org/wiki/Logistic_regression). It is essentially an **online learning** method. [SGDClassifier](http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDClassifier.html#sklearn.linear_model.SGDClassifier) supports multi-class classification by combining multiple binary classifiers in a “one versus all” (OVA) scheme. For each of the classes, a binary classifier is learned that discriminates between that and all other classes. At testing time, we compute the confidence score (i.e. the signed distances to the hyperplane) for each classifier and choose the class with the highest confidence.

SGD uses only one training sample per iteration to update the parameters and hence it is considered an online training algorithm.

Parametrs used :

**loss** : The loss function to be used

Default-‘hinge’

‘hinge’, which gives a linear SVM.

log’ loss gives logistic regression, a probabilistic classifier.

‘modified\_huber’ is another smooth loss that brings tolerance to outliers as well as probability estimates.

‘squared\_hinge’ is like hinge but is quadratically penalized.

‘perceptron’ is the linear loss used by the perceptron algorithm

**penalty** : regularization term) to be used

Defaults to ‘l2’

str, ‘none’, ‘l2’, ‘l1’, or ‘elasticnet’

‘l2’ ,the standard regularizer for linear SVM models.

‘l1’ and ‘elasticnet’ might bring sparsity to the model (feature selection) not achievable with ‘l2’.

3.alpha : Constant that multiplies the regularization term

Default- 0.0001

**n\_iter** :The number of passes over the training data

Defaults to 5

The number of iterations is set to 1 if using partial\_fit..

**n\_jobs** : integer, optional

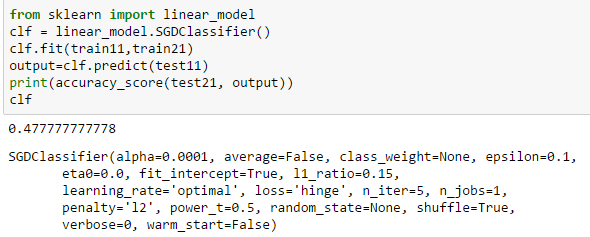
The number of CPUs to use to do the OVA (One Versus All, for multi-class problems) computation. -1 means ‘all CPUs’. Defaults to 1.

**class\_weight** : Weights associated with classes. Default – 1 for all classes

Preset for the class\_weight fit parameter.

The “balanced” mode uses the values of y to automatically adjust weights inversely proportional to class frequencies in the input data as n\_samples / (n\_classes \* np.bincount(y))

**1.Default SGD classifier**



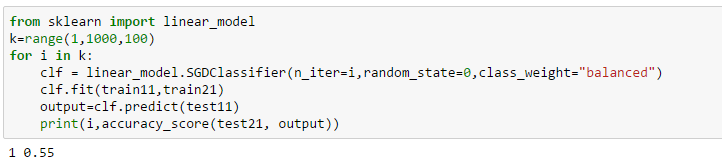
**2.SGD with iterations in increments of 100**

The number of iterations that should be done over the training data is specified and tested for different values (increments of 100 ).The accuracy varies drastically for the iterations indicating that n\_iterations play an important role in determining the accuracy. The SGD algorithm shuffles the data for each iteration and SGD fluctuates trying to reach a better local minimum each time and hence the variance in the resulting accuracy.It is hard to find the exact minimum for SGD and this explains the fluctuations in the score below

.

**3. SGD with iterations + classweight**

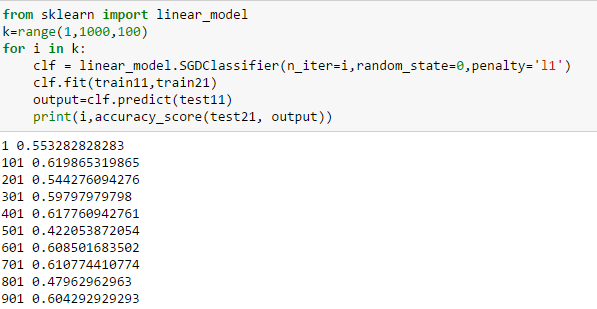
Having a balanced class weight for classes causes decrease in the accuracy as the weights are assigned inversely proportional to that of the class frequency and as the functional but needs repair class has low frequency it gets assigned more weight and as most of the datapoints in this class is mispredicted the accuracy drops.



**4. SGD with iterations+ l1 regularization parameter**

Regularization parameter is used to avoid overfitting , the default used is l2 which uses the squares difference and l1 uses the absolute difference.Using L2 will produce a model with many small coefficients, where L1 will choose a model with a large number of 0 coefficients and a few large coefficients and Elastic net is a combination of the two.

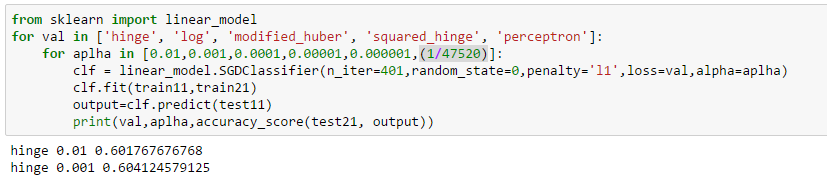
From the below result we can understand that many of the coefficients are 0 and hence our accuracy is high when using l1 norm compared to the above scores.





The comparison of the above charts show that l1 regualrization when used makes the model better than using l2.Balanced weight decreases the accuracy and when there are around 600 iterations the accuracy dips for all the above mentioned parameters. The best is to have less than 200 iterations which is confirmed by the below grid search.

**5. SGD with different alpha and loss values**

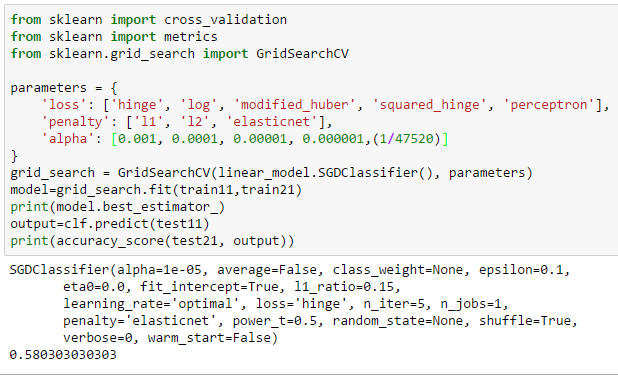


|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Hinge** | **Log** | **Modified\_huber** | **Squared\_hinge** | **Perceptron** |
| **0.01** | 0.601768 | 0.594444 | 0.59983165 | 0.601178451 | 0.600673401 |
| **0.001** | 0.604125 | 0.607492 | 0.587794613 | 0.60479798 | 0.609090909 |
| **0.0001** | 0.617761 | 0.597391 | 0.613804714 | 0.617508418 | 0.608417508 |
| **1.00E-05** | 0.583333 | 0.59596 | 0.614057239 | 0.585521886 | 0.588215488 |
| **1.00E-06** | 0.61229 | 0.612542 | 0.614814815 | 0.605723906 | 0.603872054 |
| **2.10E-05** | 0.610859 | 0.606818 | 0.591414141 | 0.615993266 | 0.58030303 |

The table shows that alpha with 0.0001 for squared\_hinge loss gives the best output with l1 penalty.

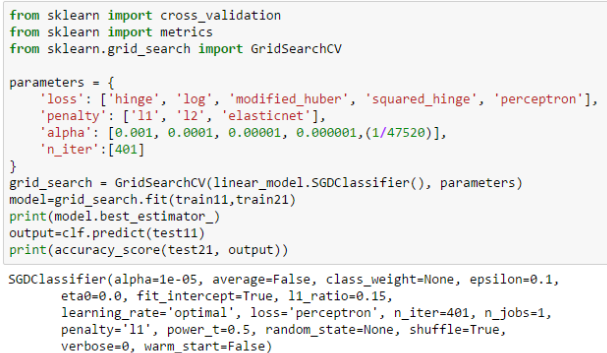
**6. Grid search results**

Random tests on the algorithm to find the best model can be exhaustive and depends on the value of the chosen by the user and is limited by it .Hence Grid search is used it helped in identifying several aspects like choosing penalty as elastic-net which would have not been considered if Grid search had not been used.Elastic net regularization overcomes the limitations of L1 and L2 norm and hence it performs better when using grid search in the given setting.



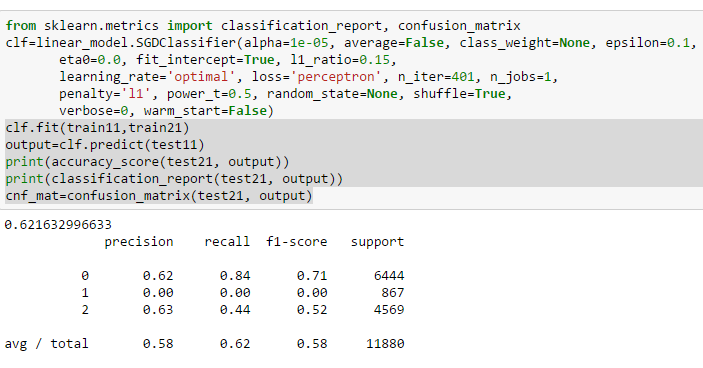
Though the above table shows that alpha with 0.0001 for squared\_hinge loss gives the best output, grid search gives the output that alpha with 0.00001 is better with hinge loss which seemed to perform low in the other graph with l1 penalty but with elastic net regularization the model works well.

After trying to verify by changing the number of iteration to 401 where the model performed better through user test it can be found that the L1 penalty is more suited for this iteration as in accordance with our assumption and the loss is perceptron, though perceptron in this setting had accuracy of 58% the scores in grid search improved due to the fact that it uses cross validation and gives the average score across all folds

.

**Best Model metrics:**

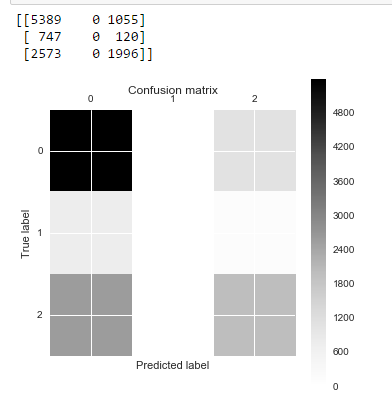
The model from Grid search yields with iteration=401 gives better results and it gives an accuracy of 62.1%. The classification report is also shown in the figure below:



Though the recall is high for class -functional , it is 0 for functional but needs repair label and less than 50% for non-functional label.

**Confusion Matrix:**

It can be seen from the below confusion Matrix that , the SGD Classifier fails to classify functional but needs repair and it classifies most of the non functional classes as functional. Thus this being a linear model doesn’t help in classifying the data correctly and it can’t be a best model for our data.



**Model 10: One vs Rest**

The strategy consists in fitting one classifier per class. For each classifier, the class is fitted against all the other classes. In addition to its computational efficiency (only n\_classes classifiers are needed)since each class is represented by one and only one classifier, it is possible to gain knowledge about the class by inspecting its corresponding classifier.

OVR was run on default models and the models that performed the best for KNN, Random Forests and SGD.Generally **OVR on the best model should perform better than the default one and this assumption is validated by the accuracy** obtained below.

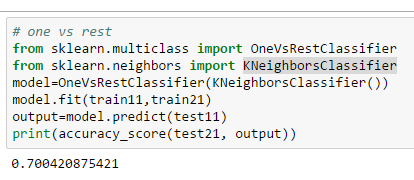
For KNN,when comparing the best model with OVR on best model the results a**re the same** with very minute variations in accuracy.Hence we can use the best KNN without OVR.

For Random forests the **accuracy was 81.27% on the best model and OVR on the best model gave a slight improvement of 81.31%**. If the minue improvement matters then OVR on the best model can be used else using the best Random forest model would suffice.

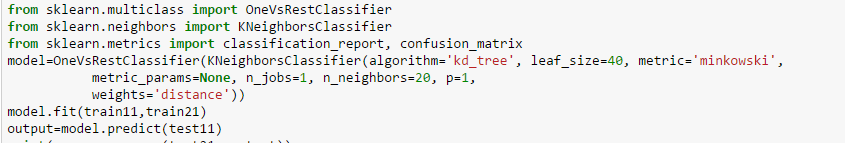
The interesting results can be seen when using OVR on the best SGD classifier , though the accuracy is low when compared to the bestSGD , it can be seen that the **recall which was 0% for functional but needs repair label has increased to 20%** , but the recall for non-functional label dropped to 9%. Hence we can conclude that with OVR on best SGD the **functional but needs repair classes can be predicted with much higher accuracy** though the overall accuracy is low.

**K-NN**

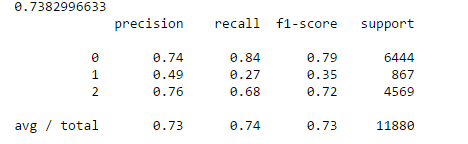
**OVR on Default KNN**



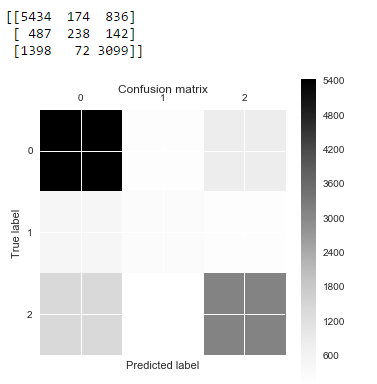
**OVR on Best KNN**



Accuracy and Classification Report:

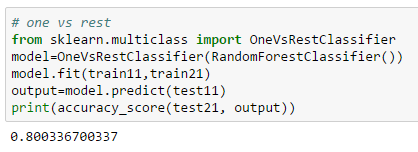


Confusion Matrix:

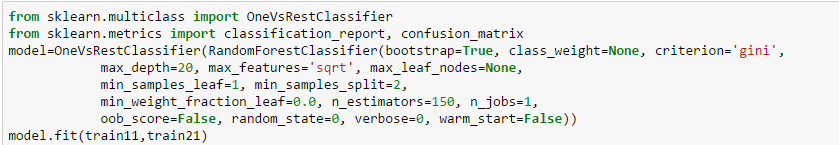


**Random Forests**

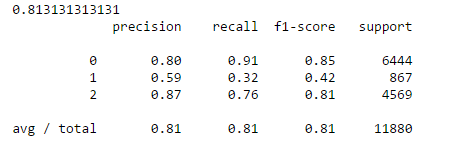
OVR ondefault Random forest classifier



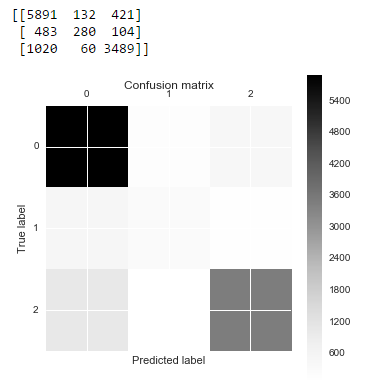
OVR on Best Random Forest classifier



Accuracy and Classification Report:

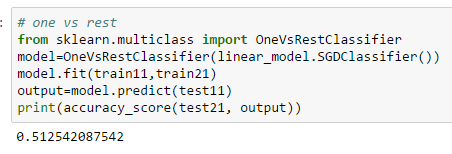


Confusion Matrix:



**SGD classifier:**

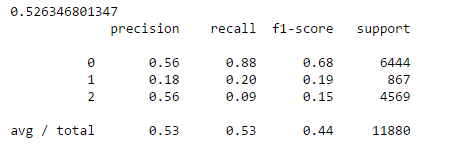
OVR on default SGD classifier



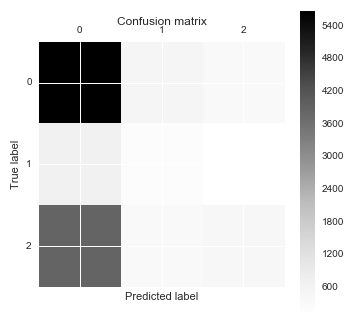
OVR on best SGD classifier



Accuracy and Classification report:



Confusion Matrix:



**Summary for Member 3:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **K-NN** | **Random Forest** | **SGD classifier** | **OVR** |
| **Comments** | Accuracy-73.83%  Recall 0-84%  Recall 1-27%  Recall 2-68% | Accuracy-81.27%  Recall 0-92%  Recall 1-30%  Recall 2-76% | Accuracy-62.16%  Recall 0-84%  Recall 1-0%  Recall 2-44% | Better to use the best of the 3 models than using OVR on them |

Here Recall 0,Recall 1,Recall 2- corresponds to recall for classes functional, functional but needs repair and non functional.

The best model after tuning is RandomForest which yields high recall for most of the classes and accuracy among all the models that were considered.

**Overall Results Summary :**

Results after tuning the model hyperparameters :

|  |  |  |
| --- | --- | --- |
| **Model** | **Accuracy** | **Recall** |
| **Random**  **Forest** | 81.27% | Class 0-92%  Class1-30%  Class 2-76% |
| **K-NN** | 73.83% | Class 0-84%  Class 1-27%  Class 2-68% |
| **SGD**  **classifier** | 62.16% | Class 0-84%  Class 1-0%  Class 2-44% |
| **Gradient Boosted Classifier** | 79.2% | Class 0-86%  Class 1-38%  Class 2-78% |
| **Adaboost** | 74.23% | CLass 0-88%  Class 1-12%  Class 2-66% |
| **Linear SVC with OvO** | 59.30% | Class 0-68%  Class 1-24%  CLass 2-64% |
| **Naive Bayes** | 57.20% | Class 0-57%  Class1-37%  Class 2-0.08% |
| **Decision Tree** | 79.5% | Class 0-53%  Class1-0.8%  Class 2-40% |
| **Logistic Regression** | 68% | Class 0-69%  Class1-30%  Class 2-0% |

# 

# Lessons Learned

* Data preprocessing plays a major role in improving the score/accuracy of the model
* If the model is sensitive to noise then it is important to address the issues like outlier, missing values, redundant columns in order to improve the model performance.
* Too many distinct values may be a hindrance and might overfit the data which can be reduced by preprocessing the data, hence this again asserts the importance of data preprocessing
* PreProcessing must be done on features that have major effect , if we perform transformations on features that aren’t important the model accuracy will not improve greatly
* Non linear model best fits our dataset and hence models like Random Forest performed better
* Random forest can handle noisy data, on the other hand it can easily overfit the data as well , hence it is important to see that the model does not overfit the data .
* A right trade off between the depth of the tree and features and number of nodes required to make a split are important in Random Forests.
* Accuracy may not be the only evaluation metric for a model , in case of a multi class classification like in our problem it makes sense to include the recall as well