Machine Learning Engineer Nanodegree

Capstone Project

Sampath Kumar

January 6th, 2017

Definition

Background

Across Africa, cholera, typhoid, dysentery and other diseases kill thousands each year. To help the people of Tanzania(2007), The Tanzanian government, with support from UN Development Programme(UNDP), responded to the water problems by the installation of Drinking Water Taps and Decentralized the maintenance for a quick response. Today this water infrastructure is facing repair and maintenance issues causing a disconnection for drinking water needs.

The Taarifa Platform is an open source web API, designed to close citizen feedback loops. Using Taarifa people can report their social issues(like water, electricity, food and other) from different forms of communications like SMS, Web Forums, Emails or Twitter. Later these reports are placed into a workflow where they can be followed up and acted upon while engaging citizens and community. A message then will to local central governing body notifying the issue & the location.

Problem Statement

Using the data gathered from Taarifa and the Tanzanian Ministry of Water, can we predict which pumps are functional, which need some repairs, and which don't work at all? Predicting one of these three classes that might have caused the failure of a water pump, can improve the maintenance operations to well prepare and ensure that clean, potable water is available to communities across Tanzania.

This project is inspired by DataDriven!

Metrics

Metric we are going to use is Accuracy Score.

As the evaluation metric of the competition use Accuracy Score /Classification Rate, we can use this metric.

The classification rate, which calculates the percentage of rows where the predicted class in the submission matches the actual class in the test set. The maximum is 1 and the minimum is 0. The goal is to maximize the classification rate.

Classification Rate = (1/N)* ∑ I(Prediction == Actual)

Example from Python Scikit Accuracy Score

```
>>> import numpy as np
>>> from sklearn.metrics import accuracy_score
>>> y_pred = [0, 2, 1, 3]
>>> y_true = [0, 1, 2, 3]
>>> accuracy_score(y_true, y_pred)
0.5
>>> accuracy_score(y_true, y_pred, normalize=False)
2
```

Analysis

Source/data files are available at DataDriven

File	Description
Training set values	The independent variables for the training set
Training set labels	The dependent variable (status_group) for each of the rows in Training set values
Test set values	The independent variables that need predictions
Submission format	The format for submitting your predictions

Data Exploration

Test & Train data sets consists of 39 columns each with 59400 rows and 14850 rows respectively, to predict 1 multi-labeled column.

Here is some simple analysis of columns along with some sample d. As the input data mostly consists of categorical data, for each we have also taken unique groups counts (or value counts) and plotted in horizontal bar charts for easy read.

Description of the Features

Index	Column Name	Unique Values	Sample Data
0	amount_tsh	98	['6000', '0', '25']
1	date_recorded	356	['2011-03-14', '2013-03-06', '2013-02-25']
2	funder	1897	['Roman', 'Grumeti', 'Lottery']
3	gps_height	2428	['1390', '1399', '686']
4	installer	2145	['Roman', 'GRUMETI', 'World']
5	longitude	57516	['34.9381', '34.6988', '37.4607']
6	latitude	57517	['-9.85632', '-2.14747', '-3.82133']
7	wpt_name	37400	['none', 'Zahanati', 'Kwa']

8	num_private	65	['0', '0', '0']
9	basin	9	['Lake', 'Nyasa', 'Lake']
10	subvillage	19287	['Mnyusi', 'B', 'Nyamara']
11	region	21	['Iringa', 'Mara', 'Manyara']
12	region_code	27	['11', '20', '21']
13	district_code	20	['5', '2', '4']
14	lga	125	['Ludewa', 'Serengeti', 'Simanjiro']
15	ward	2092	['Mundindi', 'Natta', 'Ngorika']
16	population	1049	['109', '280', '250']
17	public_meeting	2	['True', 'NaN', 'True']
18	recorded_by	1	['GeoData', 'Consultants', 'Ltd']
19	scheme_management	12	['VWC', 'Other', 'VWC']
20	scheme_name	2696	['Roman', 'NaN', 'Nyumba']
21	permit	2	['False', 'True', 'True']
22	construction_year	55	['1999', '2010', '2009']
23	extraction_type	18	['gravity', 'gravity', 'gravity']
24	extraction_type_group	13	['gravity', 'gravity', 'gravity']
25	extraction_type_class	7	['gravity', 'gravity', 'gravity']
26	management	12	['vwc', 'wug', 'vwc']
27	management_group	5	['user-group', 'user-group', 'user-group']
28	payment	7	['pay', 'annually', 'never']
29	payment_type	7	['annually', 'never', 'pay']
30	water_quality	8	['soft', 'soft', 'soft']
31	quality_group	6	['good', 'good', 'good']
32	quantity	5	['enough', 'insufficient', 'enough']
33	quantity_group	5	['enough', 'insufficient', 'enough']
34	source	10	['spring', 'rainwater', 'harvesting']
35	source_type	7	['spring', 'rainwater', 'harvesting']
36	source_class	3	['groundwater', 'surface', 'surface']
37	waterpoint_type	7	['communal', 'standpipe', 'communal']
38	waterpoint_type_group	6	['communal', 'standpipe', 'communal']

(For 39 column's unique values counts)(98, 356, 1897, 2428, 2145, 57516, 57517, 37400, 65, 9, 19287, 21, 27, 20, 125, 2092, 1049, 2, 1, 12, 2696, 2, 55, 18, 13, 7, 12, 5, 7, 7, 8, 6, 5, 5, 10, 7, 3, 7, 6)
(54298963910136592779415206217899249183840417288083079168000000000000

(69 digits) is product of these 39 unique values, which is exponentially greater than 59K records we have) is the ideal amount of sufficient data to cover each and every category.

Input labels data has 39(27 object columns and 16 non-object columns) Features with 59,400 rows. Although we seem to have a good data set, looking at the unique values counts from below 39 columns we can say that we could potentially encounter Curse of Dimensionality. But, as we can see some of columns pairs (extraction_type, extraction_type_group), (quantity & quantity_group), (source, source_class) seems have closer relation and column by 'recorded_by' has only one unique value. So, we might have a chance to escape Curse of Dimensionality.

Description of the Labels

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

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

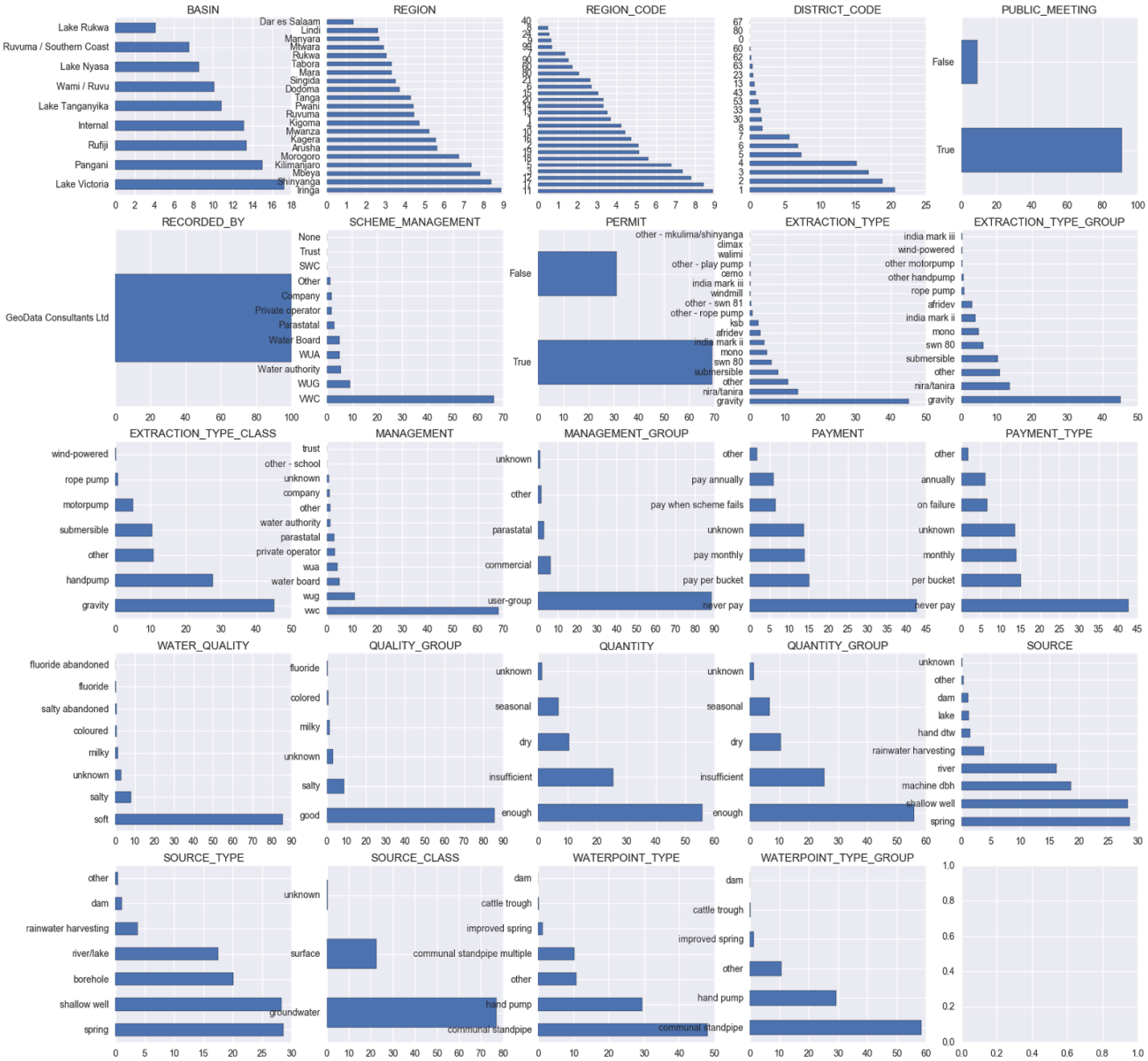
Prediction Labels	Counts	Percentage
functional	32259	54.30
non functional	22824	38.42
functional needs repair	4317	07.26

As numbers show we have data for unequal proportions. So the in normal circumstances if we train a model to learn the there might be changes where model tried to predict only first two groups which would only include ~92% data for learning.

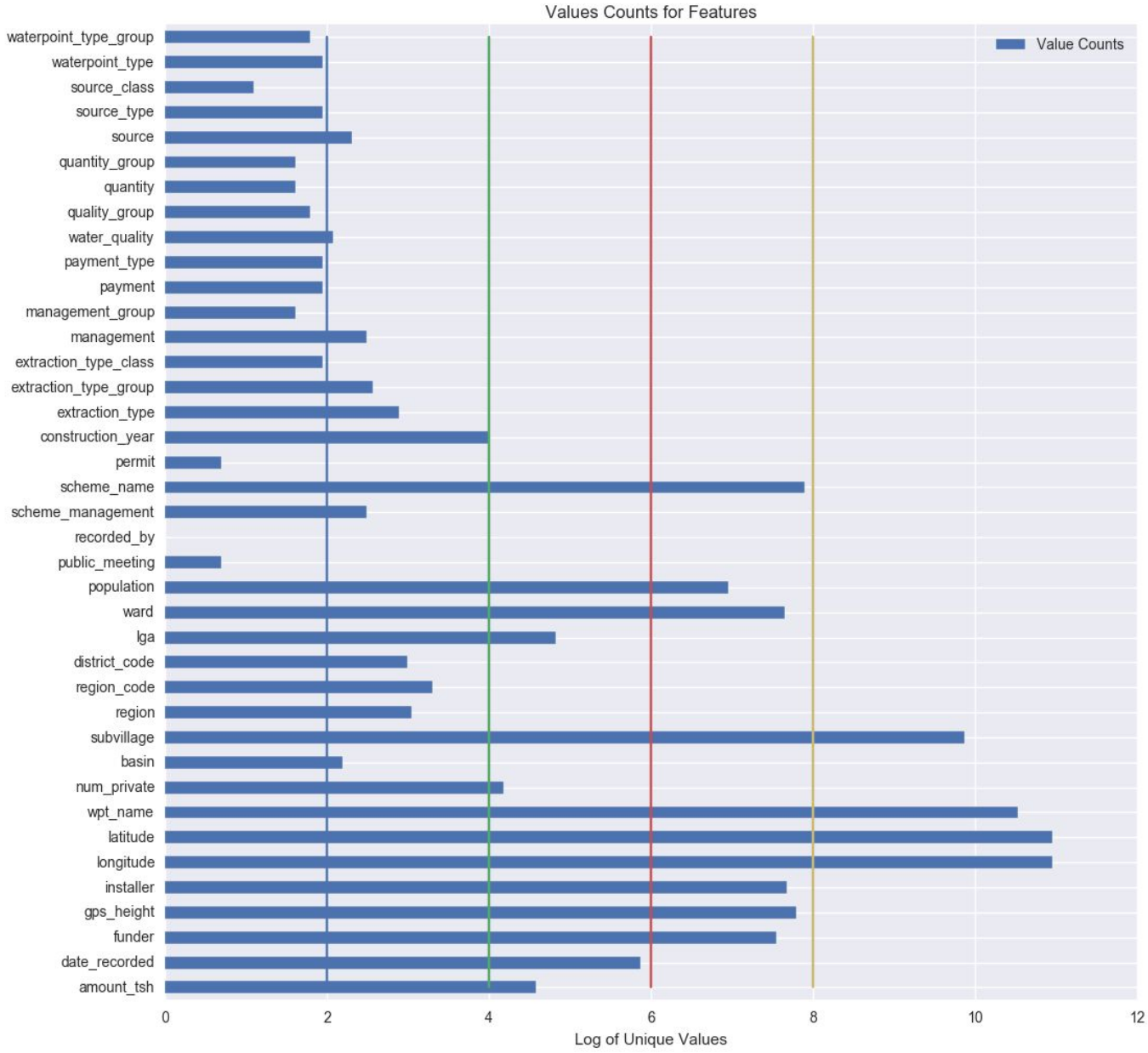
To create a generic model which could work in all scenario, we will use stratification selection for splitting test-train data.

Exploratory Visualization

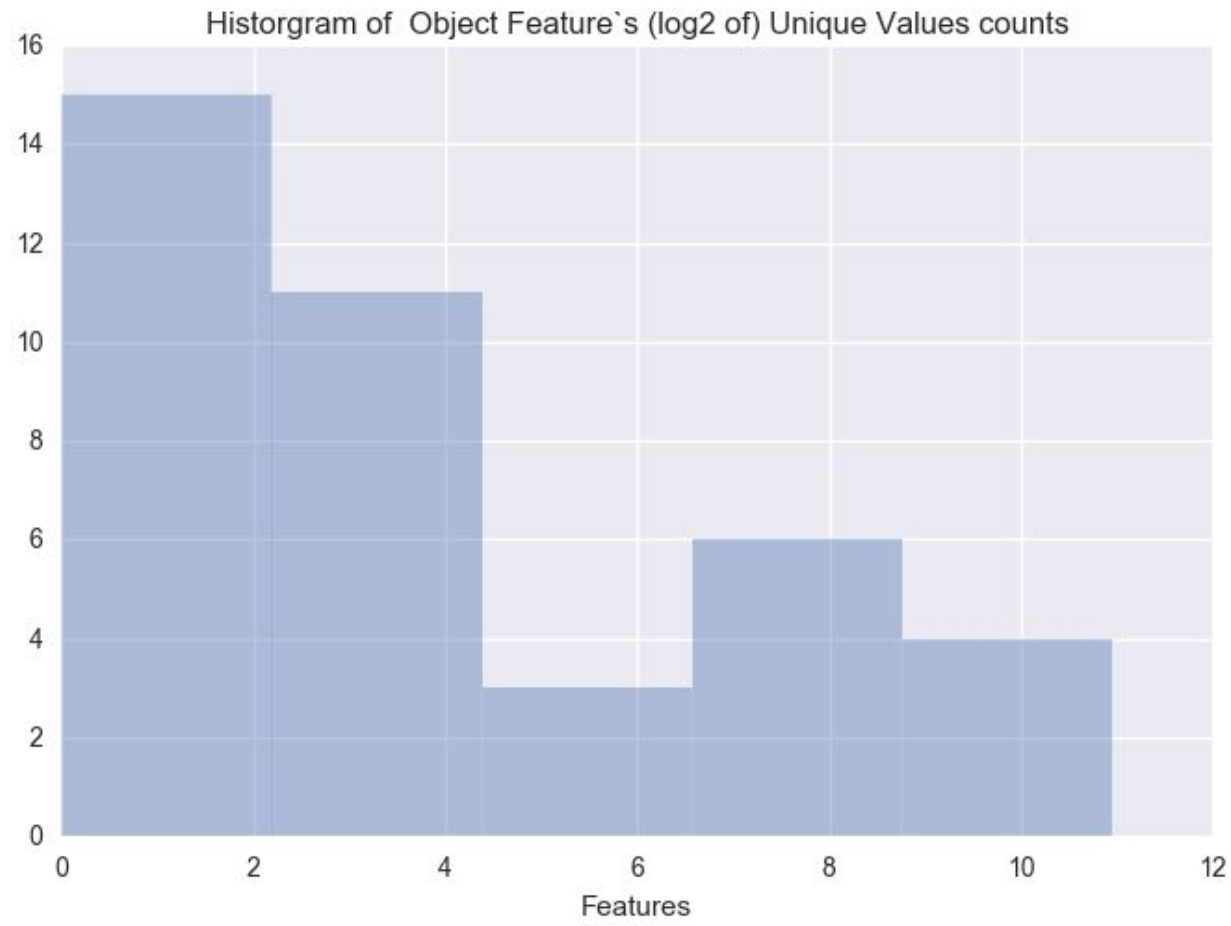
Visualization of Object Columns Value Counts.



Bar plot of all Object Column's Value counts. __ NOTES:__ Values shown in image are log transformed to show differences visually.



Histogram of all Object Column's Value counts.



__ Observations and Suggestions __

- Most of the data seems categorical: As this would increase the number of dimensions the results vary, we can take a deep look of how data is distributed across the groups and focus of the groups which contribute more information overall.
- Need to check Date columns
    - we shall convert date -> day, month, year, weekday, total_no_of_day_from_reference_point. These splits for two reasons.
        - Reason 1: It might be possible that in some location all specific set of complaints are registered on a start/mid/at end of the month. It might also be possible that they are registered on every Monday or so.
        - Reason 2: Taking as much information as possible.
- Need to check Float and Bool columns

    - Longitude & latitude(features) seem to hold (0,0) instead of NULL which is acting as outliers for now.
    - Longitude and latitude(features) are too much precise(in input date) that would make it too difficult to generalize. As generally water pumps are not installed next to each other to maintain a precision of high accuracy, we can reduce it.
    - Few boolean columns are having Nan(NAN or Null) values which does not completely make them boolean series. If we look into this scenario of observing not having data as another indication (new information), to preserve this knowledge we can convert them into labels.
- Following pairs looks closely related
    - quantity & quantity_group
    - quality_group & water_quality
    - extraction_type, extraction_type_class & extraction_type_group
- Other
    - categorical columns like installer, funder, scheme_name seems to hold data issue like text capitalization & trailing spaces.
    - recorded_by, seems to hold only a single value
    - population & amount_tsh, values are for some given as zero

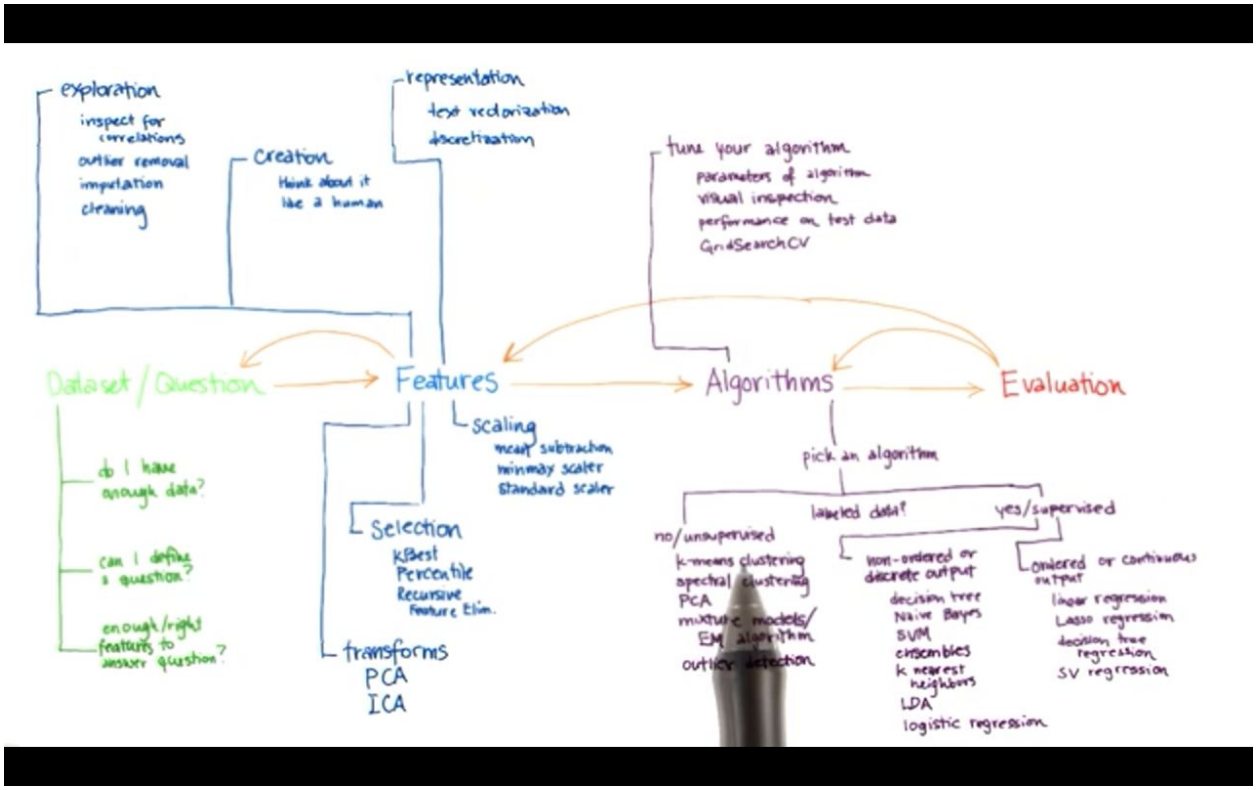
Benchmark Model

With simplistic data labeling and with the help of Random Forest Classifiers, we have created a benchmark submission of 0.7970 for which source code is here.

Methodology

Initial Project Design

As shown in below image, we are going to do a step by step development progress on here.



With Random Forest Classifier, we were able to generate a benchmark of 0.7970. So, first we will start with going to check understanding of Random Forest worked and what features contributed it to generate this score in training.

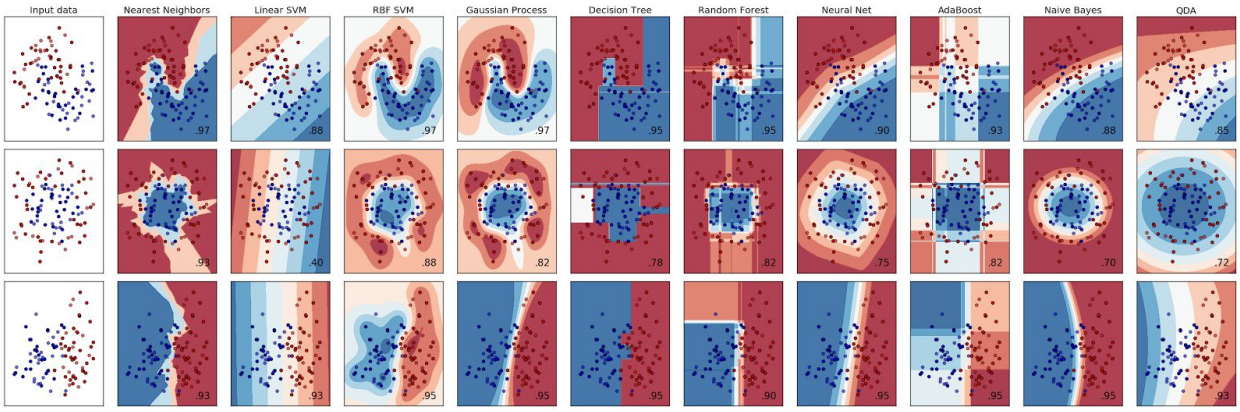
Project Design

1. Questions on data
2. Feature Exploration
    - PCA Transformation Checking
    - Select K Best Checking
    - Exploration - outliers check
3. Algorithm Selection
    - Unsupervised Learning Exploration(Gaussian Process, KMeans)
    - Supervised Learning(GBT Trees, Nearest Neighbors, RF, One-vs-One)
    - Parameter Tuning
4. Evaluation. Back to 1 with.
5. Re-Evaluation with threshold improvisation check.
6. Submission

Algorithms and Techniques

As described in the introduction, a smart understanding of which water points will fail can improve maintenance operations and ensure that clean, potable water is available to communities across Tanzania.

A Classifier comparison from Sklearn's documentation



For initial understanding, as we can see from above analysis on different kinds of datasets, I find that *Nearest Neighbor* performs better when Random Forest is performing low. Also to have different learning process from that of Random Forest, we will also consider GBT Trees, which seem seems performed better than Random Forest.

We will be using Gaussian Process, KMeans clustering for unsupervised Learning exploration. We have taken two models different kinds of models for exploration, so that we can compare one among them.

So, We will use familiar (inherently) multi-class Supervised Classifiers like Tree Algorithms(RF, GBT). As these models are easy to train, self-learning & evaluation nature make them a general good technique.

From dataset features, we have coordinates like longitude, latitude and pump models and other, it might even possible that similar issues are observed in certain neighbor and have been reported already so Nearest Neighbor models could also perform well.

During model selection, we will also explore One-vs-Rest Sklearn's MultiClassification Technique. As the data is unbalanced, we believe that a One-vs-Rest might perform well.

We have tried following Algorithms to check which kind of family model fits well data with simple parameters. During

- GBT Trees
- Nearest Neighbors
- Random Forest
- Multi Class
    - One vs Rest with Random Forest
    - One vs One with Random Forest
- Parameter tuning
- XGBOOST

Data Preprocessing

As mentioned earlier majority of the data is object columns,

- Date Columns: We have one columns *date_recorded*, which supposed to show on which data record was added.
- Bool Columns: Instead of deleting null or replacing null with True or False, we have converted all these into numbers and thus not losing any information.

```
True -> 1, False ->2, Nan or None ->3
```

- Int Columns: For some integer columns like Geo location co ordinates, the precision is so good that we can point the location to centimeter level. But this seemed to too precise(too much information) and so in hit and trial, when we reduced the precision up to 3 digit we found that our benchmark model was performing well.

For other numerical model, labeling will work as MinMaxScalar.

- Object Columns: During the sub group plotting we have noticed that minor text capitalization issue and spaces. So we have applied a transformer to convert all the object data to lower case ASCII strings.

Top 5 columns with huge varieties

- funder, 1898
- installer, 2146
- wpt_name, 37400
- subvillage, 19288
- scheme_nam, 2697

For these columns as we look into details we have observed that most of the data has lots of cardinality issues and here are some stats collected for these columns.

- funder:
    - 100.0 percentage of DATA coverage mean, 1881 (in number) groups
    - 97.0 percentage of DATA coverage mean, 592 (in number) groups ##
    - 90.5 percentage of DATA coverage mean, 237 (in number) groups
- installer:
    - 100.0 percentage of DATA coverage mean, 1867 (in number) groups
    - 97.0 percentage of DATA coverage mean, 599 (in number) groups ##
- wpt_name:
    - 80.0 percentage of DATA coverage mean, 24838 (in number) groups ##
- subvillage:
    - 80.5 percentage of DATA coverage mean, 8715 (in number) groups ##
    - 83.0 percentage of DATA coverage mean, 9458 (in number) groups
- ward:
    - 80.0 percentage of DATA coverage mean, 998 (in number) groups ##
    - 91.5 percentage of DATA coverage mean, 1397 (in number) groups
    - 100.0 percentage of DATA coverage mean, 2093 (in number) groups
- scheme_name:

- 100.0 percentage of DATA coverage mean, 2486 (in number) groups
- 91.5 percentage of DATA coverage mean, 870 (in number) groups
- 80.5 percentage of DATA coverage mean, 363 (in number) groups
- 85.0 percentage of DATA coverage mean, 524 (in number) groups ##

NOTE: Marked with double hashes are the selected values for coverage.

In the Ipython Notebook, we have created a generic helper script to do this and Ipython Widget for experimentations.

For further references say, lets call this new formatted/transformed data as (Pre-)Processed Data. At this stage we are having 43 features.

Feature Selection

After preprocessing, we have tried 3 methods of dimensionality reductions.

Variance Threshold:

VarianceThreshold is a simple baseline approach to feature selection. It removes all features whose variance doesn't meet some threshold. By default, it removes all zero-variance features, i.e. features that have the same value in all samples.

We have taken a variance threshold limit of 80%, implies columns with less than 80 are to be dropped. We found one columns *recorded_by* which has a variance threshold less than 80%.

KBest select

KBest is one of the Univariate feature selection methods that works by selecting the best features based on univariate statistical tests.

Well known statistical tests for classification are chi2, f_classif, mutual_info_classif. Using Random Forest we are checking which statistical method is generating better results for K=30(selected columns).

Statistical Test	Train Score	Test Score
chi2	0.98428731762065091	0.79966329966329963

_classif	).97432098765432096	).79286195286195282
nutual_info_classif	).98410774410774415	).79447811447811445

KBest statistical method: Chi2 wins.

Like we have tried all these three methods to be sure, we shall also check the number of columns to find the best number of minimum required columns/features to better score.

AMOUNT_TSH, DATE_RECORDED, FUNDER, GPS_HEIGHT, INSTALLER, LONGITUDE, LATITUDE, NUM_PRIVATE, BASIN, SUBVILLAGE, REGION, REGION_CODE, DISTRICT_CODE, LGA, WARD, POPULATION, PUBLIC_MEETING, SCHEME_MANAGEMENT, SCHEME_NAME, PERMIT, CONSTRUCTION_YEAR, EXTRACTION_TYPE, EXTRACTION_TYPE_GROUP, EXTRACTION_TYPE_CLASS, MANAGEMENT, MANAGEMENT_GROUP, PAYMENT, PAYMENT_TYPE

Results of previous runs Trail 1


```
[{'cols': 1, 'test': 0.52659932659932662, 'train': 0.57483726150392822},
 {'cols': 5, 'test': 0.68962962962962959, 'train': 0.94240179573512906},
 {'cols': 9, 'test': 0.7211447811447812, 'train': 0.97638608305274976},
 {'cols': 13, 'test': 0.75380471380471381, 'train': 0.97955106621773291},
 {'cols': 17, 'test': 0.76134680134680133, 'train': 0.98071829405162736},
 {'cols': 21, 'test': 0.76511784511784509, 'train': 0.98076318742985413},
 {'cols': 25, 'test': 0.80033670033670035, 'train': 0.98316498316498313},
 {'cols': 29, 'test': 0.80053872053872055, 'train': 0.98379349046015707},
 {'cols': 33, 'test': 0.80040404040404045, 'train': 0.98390572390572395},
 {'cols': 37, 'test': 0.79993265993265994, 'train': 0.98341189674523011}]
```


Trail 2


```
[{'cols': 23, 'test': 0.7976430976430976, 'train': 0.9836812570145903},
 {'cols': 25, 'test': 0.80033670033670035, 'train': 0.98316498316498313},
 {'cols': 27, 'test': 0.80101010101010106, 'train': 0.9829405162738496},
 {'cols': 29, 'test': 0.80053872053872055, 'train': 0.98379349046015707},
 {'cols': 31, 'test': 0.80000000000000004, 'train': 0.98381593714927051}]
```


Trail 3

[{'cols': 26, 'test': 0.80309764309764309, 'train': 0.98359147025813698},
{'cols': 27, 'test': 0.80101010101010106, 'train': 0.9829405162738496},
{'cols': 28, 'test': 0.80222222222222217, 'train': 0.98334455667789}]

As per Okham Razor's rules, we are going to select the simplest and well performing. Luckily, we have got kbest_selected_cols at *26* which is comparatively top performer among other K-selections and also lower than actually number of columns

*Conclusion: Using __Chi2_* with KBest, we found 26 best selected columns for generating results._

PCA

PCA, Linear dimensionality reduction using Singular Value Decomposition of the data to project it to a lower dimensional space.

Here is the cumulative

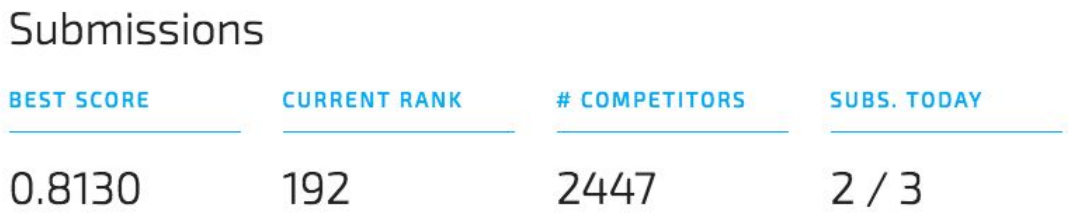
Like KBest, in a similar fashion we have tried PCA model but we have encounter some decrease in score. As we can understand from the results, that transformed will have lower dimensions but it might be always to learn from it.

Results

Initial benchmark using Random Forest was 0.7970.

First revision

After initial benchmark, as planned in project design we tried basic data transformations, labelisations, different algorithms, hyper parameter tuning and even multi class classifiers. At this stage we did not go much deep into understanding of data labels and transformation and we kept it as second revision.



Second revision

In the second revision, we have build a special custom Labeliser to get in depth understanding of labels and analyzed that labels like funder, wpt_name, subvillage and other. Here we found the interesting details like only 50% or 30% of total groups are covering 80% to 90% of data while only minimal amount of data is too much scattered. So, we have optimized these groups accordingly in funder, wpt_name, ward and other columns. (Details of these are already explained in the Data PreProcessing Stage.)

Final score we achieved is 0.8201.

Future Improvements Projects

- Like correlation for numerical columns, objects Columns can be looked up for Associations Techniques.
- Unsupervised learning explorations were failed and could not generate much results till now, so there is scope for further research in Unsupervised learning.

Sources & References

- DataDriven
- Choosing a ML Classifier
- Submission Code
- Wikipedia: Water Supply & Sanitation in Tanzania
- UN Report
- UN 2007 Water Taps Installation
- GBT Video Lecture, GBT
- Multi-class Classification
- Multi-class and multi label algorithms
- Multi-class Metric
- Standford UnSupervised Learning