Capstone 2 - Predicting Water Pump Condition EDA

September 8, 2019

1 Capstone 2 - Predicting Water Pump Condition in Tanzania EDA

Kenneth Liao

1.1 Background

The UN publishes and reviews a list of least developed countries (LDC) every 3 years. LDCs are "low-income countries confronting severe structural impediments to sustainable development. They are highly vulnerable to economic and environmental shocks and have low levels of human assets."[1]. Tanzania has been classified as an LDC since the UN published the first list of LDCs in 1971[2]. A common challenge of LDCs is a lack of infrastructure to support the development of the nation, including access to education and healthcare, waste management, and access to potable water.

According to UNICEF, as of 2017, more than 24 million Tanzanians lacked access to basic drinking water[3]. This corresponds to only 56.7% of the country's population having access to basic drinking water. Outside of developed urban areas, much of the potable water is accessed via water pumps.

Taarifa is an open-source platform for crowd-sourced reporting and triaging of infrastructure related issues. Together with the Tanzanian Ministry of Water, data has been collected for thousands of water pumps throughout Tanzania. The goal of this project is to be able to predict the condition of these water pumps to improve maintenance, reduce pump downtime, and ensure basic water access for millions of Tanzanians.

References

1. https://www.un.org/development/desa/dpad/least-developed-country-category.html
2. https://www.un.org/development/desa/dpad/wp-content/uploads/sites/45/publication/ldc_list.pdf
3. https://washwatch.org/en/countries/tanzania/summary/statistics/

1.1.1 Problem Description

Predict the operating condition of water pumps in Tanzania given various metadata on each water pump.

1.1.2 Strategy

The strategy will be to implement an XGBoost model as well as a neural network model for predictions and compare their performance.

1.1.3 Data

The dataset is provided by Taarifa, together with the Tanzanian Ministry of Water and is hosted by DrivenData.org:

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

1.2 Exploratory Data Analysis

Start by importing the necessary libraries and datasets.

```
In [1]: import pandas as pd
        import plotly.graph_objs as go
        from plotly.offline import iplot, plot, init_notebook_mode
        import plotly.express as px
        from config import credentials

        init_notebook_mode(connected=True)
```

```
In [2]: # load the data
        train = pd.read_csv('../data/train.csv')
        train_labels = pd.read_csv('../data/train-labels.csv')
```

1.2.1 Prediction Labels

```
In [3]: train_labels.shape

Out[3]: (59400, 2)
```

```
In [4]: train_labels.head()

Out[4]:       id    status_group
        0  69572      functional
        1   8776      functional
        2  34310      functional
        3  67743  non functional
        4  19728      functional
```

```
In [5]: train_labels.id.nunique()

Out[5]: 59400
```

The train_labels file contains the labels we want to predict, `status_group`. This is the condition of a given water pump.

```
In [6]: counts = train_labels.groupby('status_group').count()
        counts
```

```
Out[6]:                             id
        status_group
        functional                 32259
        functional needs repair     4317
        non functional             22824
```

```
In [7]: trace0 = go.Bar(name='functional', x=['functional'], y=[counts.loc['functional','id']]
                        marker=dict(color='lightgreen'), showlegend=False)

        trace1 = go.Bar(name='functional needs repair', x=['functional needs repair'], y=[coun
                        marker=dict(color='orange'), showlegend=False)

        trace2 = go.Bar(name='non functional', x=['non functional'], y=[counts.loc['non functi
                        marker=dict(color='tomato'), showlegend=False)

        layout = go.Layout(title='Pump Condition Distribution',
                           yaxis=dict(title='Count'))

        fig = go.Figure([trace0, trace1, trace2], layout=layout)

        iplot(fig, filename='pump-conditions.html')
```

```
In [8]: counts/counts.id.sum()
```

```
Out[8]:                                  id
        status_group
        functional                 0.543081
        functional needs repair    0.072677
        non functional             0.384242
```

54.3% of pumps are functional, while 7.3% are functional but require repair and 38.4% are non functional.

1.2.2 Features

```
In [9]: train = train.set_index('id')
        train.head()
```

```
Out[9]:        amount_tsh date_recorded        funder  gps_height     installer  \
        id
        69572      6000.0    2011-03-14         Roman        1390         Roman
        8776          0.0    2013-03-06       Grumeti        1399       GRUMETI
        34310        25.0    2013-02-25  Lottery Club         686  World vision
        67743         0.0    2013-01-28        Unicef         263        UNICEF
        19728         0.0    2011-07-13   Action In A           0       Artisan

               longitude   latitude              wpt_name  num_private  \
        id
        69572  34.938093  -9.856322                  none            0
```

```
8776   34.698766  -2.147466             Zahanati            0
34310  37.460664  -3.821329          Kwa Mahundi            0
67743  38.486161 -11.155298  Zahanati Ya Nanyumbu            0
19728  31.130847  -1.825359              Shuleni            0

                          basin          ...          payment_type  \
id                                       ...
69572                Lake Nyasa          ...              annually
8776              Lake Victoria          ...             never pay
34310                   Pangani          ...            per bucket
67743  Ruvuma / Southern Coast          ...             never pay
19728             Lake Victoria          ...             never pay

      water_quality  quality_group      quantity quantity_group  \
id
69572          soft           good        enough         enough
8776           soft           good  insufficient   insufficient
34310          soft           good        enough         enough
67743          soft           good           dry            dry
19728          soft           good      seasonal       seasonal

                     source          source_type source_class  \
id
69572                spring               spring  groundwater
8776   rainwater harvesting  rainwater harvesting      surface
34310                   dam                  dam      surface
67743           machine dbh             borehole  groundwater
19728  rainwater harvesting  rainwater harvesting      surface

                   waterpoint_type waterpoint_type_group
id
69572           communal standpipe    communal standpipe
8776            communal standpipe    communal standpipe
34310  communal standpipe multiple    communal standpipe
67743  communal standpipe multiple    communal standpipe
19728           communal standpipe    communal standpipe

[5 rows x 39 columns]
```

In [10]:
```
train.shape
```

Out[10]:
```
(59400, 39)
```

The shape of the full feature data is (59400,39). Having more than 2 orders of magnitude worth of samples compared to the number of features will help avoid the curse of dimensionality. The concern would be with the label with the smallest sample size. The data for the condition "functional needs repair" is (4317, 39). In thise case, the number of samples is still 2 orders of magnitude larger than the number of features.

The preview above shows that the training data contains 39 features with mixed datatypes. The descriptions of each feature are described below, from the data source.

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 latitude - GPS coordinate wpt_name - Name of the waterpoint 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 waterpoint scheme_name - Who operates the waterpoint permit - If the waterpoint is permitted construction_year - Year the waterpoint was constructed extraction_type - The kind of extraction the waterpoint uses extraction_type_group - The kind of extraction the waterpoint uses extraction_type_class - The kind of extraction the waterpoint uses management - How the waterpoint is managed management_group - How the waterpoint 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 waterpoint waterpoint_type_group - The kind of waterpoint

Some of the feature descriptions look very similar to each other. I will be checking if there are duplicate features, or if some of the features are derived from others, this will help reduce the number of features to just the essential ones.

```
In [89]: def data_check(data, col):
             print('There are %s ' % data[col].isnull().sum() + ' null values.')
             print('There are %s ' % data[col].nunique() + ' unique values.')
```

```
In [90]: def plot_hist(data, col, ylog=False, xlog=False):

             if ylog:
                 ymode='log'
             else:
                 ymode=None
             if xlog:
                 xmode='log'
             else:
                 xmode=None

             trace = go.Histogram(x=data[col], name='col')

             layout = go.Layout(title=f'{col} Distribution',
                           yaxis=dict(title='Count', type=ymode),
                                xaxis=dict(type=xmode))

             fig = go.Figure([trace], layout=layout)

             iplot(fig, filename=f'{col}-dist.html')
```

amount_tsh `amount_tsh` is the "total static head" of the pump, described as the total amount of water available to the waterpoint. Total static head is the distance from the surface of the water source to the surface where the water is collected from the pump. It is related to the pressure of the water source and therefore the amount of water present. The distribution of `amount_tsh` is shown below on a semi-log plot. Most of the pumps are near 0 total static head.

```
In [12]: plot_hist(train, col='amount_tsh', ylog=True)
```

```
In [13]: train.amount_tsh.value_counts().head()
```

```
Out[13]: 0.0       41639
         500.0      3102
         50.0       2472
         1000.0     1488
         20.0       1463
         Name: amount_tsh, dtype: int64
```

The top 5 value counts for `amount_tsh` are summarized above. 41,639 of the 59,400 pumps have a total static head of 0. This implies the water source is at the surface.

```
In [14]: train.amount_tsh.isnull().any()
```

```
Out[14]: False
```

There are no null values in this feature column!

date_recorded

```
In [15]: train.date_recorded.isnull().any()
```

```
Out[15]: False
```

The `date_recorded` column also has no missing values.

```
In [16]: plot_hist(train, col='date_recorded')
```

The first date we have a record for is 2002-10-14. The data is very sparse prior to 2011. If we zoom in to Jan 2011 and beyond, there appear to be 4 periods of heavy data collection. After April 2013, there also appears to be some more regular data collection each month, on about 200-300 pumps at a time.

funder

```
In [91]: data_check(train, 'funder')
```

```
There are 3635  null values.
There are 1897  unique values.
```

3,635 of the 59,400 samples are missing their funding source. I can't imagine how the funding source would be related to a pump failing or not so I think it's safe to drop this column.

gps_height

```
In [19]: plot_hist(train, col='gps_height')
```

```
In [20]: train.gps_height.isnull().any()
```

```
Out[20]: False
```

The `gps_height` is the altitude of the water pump. Luckily, there are no missing data in this column.

installer

```
In [93]: data_check(train, 'installer')
```

```
There are 3655  null values.
There are 2145  unique values.
```

There are 2,145 unique installers. Unfortunately, there are also 3,655 missing values for `installer`. This feature can affect water pump condition if for example, the quality of the installation varies between installers.

latitude, longitude

```
In [94]: data_check(train, 'latitude')
```

```
There are 0  null values.
There are 57517  unique values.
```

```
In [95]: data_check(train, 'longitude')
```

```
There are 0  null values.
There are 57516  unique values.
```

No missing values for latitude or longitude!

```
In [23]: px.set_mapbox_access_token(credentials['mapbox_token'])

         fig = px.scatter_mapbox(train, lat="latitude", lon="longitude",
                                 color='gps_height',
                                 size_max=15, zoom=4.5)
         fig.show()
```

The map above shows the locations of the pumps with a color gradient showing the altitude of each pump (`gps_height`). The pumps are mostly in concentrated areas but there are some small groups of dispersed pump sites.

wpt_name

```
In [96]: data_check(train, 'wpt_name')

There are 0  null values.
There are 37400  unique values.
```

The `wpt_name` is simply the name of the water point. Interestingly, every waterpoint has a name, there are no null values. There are 37,400 unique names while we have 59,400 water points, so waterpoints can have duplicate names. Below are 20 sample names.

```
In [48]: train.wpt_name.unique()[:20]

Out[48]: array(['none', 'Zahanati', 'Kwa Mahundi', 'Zahanati Ya Nanyumbu',
                'Shuleni', 'Tajiri', 'Kwa Ngomho', 'Tushirikiane',
                'Kwa Ramadhan Musa', 'Kwapeto', 'Mzee Hokororo',
                'Kwa Alid Nchimbi', 'Pamba', 'Kwa John Izack Mmari', 'Mwabasabi',
                "Kwa Juvenal Ching'Ombe", 'Kwa John Mtenzi', 'Kwa Rose Chaula',
                'Ngomee', 'Muungano'], dtype=object)
```

num_private

```
In [97]: data_check(train, 'num_private')

There are 0  null values.
There are 65  unique values.
```

```
In [54]: train.num_private.value_counts()

Out[54]: 0      58643
         6         81
         1         73
         5         46
         8         46
         32        40
         45        36
         15        35
         39        30
         93        28
         3         27
         7         26
         2         23
         65        22
         47        21
         102       20
         4         20
         17        17
```

```
80          15
20          14
25          12
11          11
41          10
34          10
16           8
120          7
150          6
22           6
12           5
24           5
          ...
14           3
61           3
27           2
26           2
160          1
30           1
698          1
60           1
1402         1
450          1
668          1
131          1
35           1
672          1
42           1
136          1
87           1
300          1
280          1
141          1
62           1
111          1
240          1
1776         1
755          1
180          1
213          1
23           1
55           1
94           1
Name: num_private, Length: 65, dtype: int64
```

Unfortunately, there is no description of `num_private`. There are 65 unique values for this feature with no m issing data. The vast majority of samples however have a value of 0 for this feature, 58,643 out of 59,400 have a value of 0.

basin

```
In [98]: data_check(train, 'basin')
```

```
There are 0  null values.
There are 9  unique values.
```

```
In [55]: plot_hist(train, col='basin')
```

basin refers to the water basin for the water source. The water basin with the most water pumps is Lake Victoria and the one with the least is Lake Rukwa. Below we can visualize the distribution of waterpumps according to which basin they draw water from. Lake Victoria is at the very North end of Tanzania, and we can see all of the neighboring waterpoints that access it.

```
In [63]: px.set_mapbox_access_token(credentials['mapbox_token'])

         fig = px.scatter_mapbox(train, lat="latitude", lon="longitude",
                                 color='basin',
                                 size_max=15, zoom=4.5)
         fig.show()
```

subvillage

```
In [99]: data_check(train, 'subvillage')
```

```
There are 371  null values.
There are 19287  unique values.
```

There are 19,287 unique subvillages with 371 missing values.

region

```
In [100]: data_check(train, 'region')
```

```
There are 0  null values.
There are 21  unique values.
```

```
In [62]: plot_hist(train, col='region')
```

There are 21 unique regions with 0 missing values. We can see how there are multiple regions that draw from the same basins.

```
In [64]: px.set_mapbox_access_token(credentials['mapbox_token'])

         fig = px.scatter_mapbox(train, lat="latitude", lon="longitude",
                                 color='region',
                                 size_max=15, zoom=4.5)
         fig.show()
```

region_code

```
In [101]: data_check(train, 'region_code')
```

```
There are 0  null values.
There are 27  unique values.
```

There are actually 27 unique `region_codes` which is unexpected. Let's see how they are mapped to region names.

```
In [70]: train[['date_recorded', 'region', 'region_code']].groupby(['region', 'region_code']).c
```

```
Out[70]:                               date_recorded
         region        region_code
         Arusha        2                       3024
                       24                       326
         Dar es Salaam 7                        805
         Dodoma        1                       2201
         Iringa        11                      5294
         Kagera        18                      3316
         Kigoma        16                      2816
         Kilimanjaro   3                       4379
         Lindi         8                        300
                       18                         8
                       80                      1238
         Manyara       21                      1583
         Mara          20                      1969
         Mbeya         12                      4639
         Morogoro      5                       4006
         Mtwara        9                        390
                       90                       917
                       99                       423
         Mwanza        17                        55
                       19                      3047
         Pwani         6                       1609
                       40                         1
                       60                      1025
         Rukwa         15                      1808
         Ruvuma        10                      2640
         Shinyanga     11                         6
                       14                        20
                       17                      4956
         Singida       13                      2093
         Tabora        14                      1959
         Tanga         4                       2513
                       5                         34
```

So a given region can have multiple region codes. region_codes are not completely unique across regions. Notice that region 5 appears both for the Tanga region and Morogoro.

district code

```
In [102]: data_check(train, 'district_code')
```

```
There are 0  null values.
There are 20  unique values.
```

There are 20 unique district codes with no missing values.

```
In [73]: train[['date_recorded', 'region_code', 'district_code']].groupby(['region_code', 'dis
```

```
Out[73]:                              date_recorded
         region_code district_code
         1           0                           23
                     1                          888
                     3                          361
                     4                          347
                     5                          358
                     6                          224
         2           1                          189
                     2                         1206
                     3                          109
                     5                          201
                     6                          310
                     7                         1009
         3           1                          595
                     2                          519
                     3                          877
                     4                         1225
                     5                          620
                     6                          109
                     7                          434
         4           1                          698
                     2                          408
                     3                          323
                     4                          110
                     5                          293
                     6                          266
                     7                          127
                     8                          288
         5           1                         1128
                     2                          521
                     3                          997
         ...                                    ...
         19          5                          332
                     6                          488
                     7                          347
                     8                          132
```

```
20          1                        171
            2                        716
            3                        396
            4                        438
            6                        248
21          1                        550
            2                        274
            3                        297
            4                        276
            5                        186
24          30                       326
40          43                         1
60          33                       115
            43                       350
            53                       454
            60                        63
            63                        37
            67                         6
80          13                       391
            23                       293
            43                       154
            53                       291
            62                       109
90          33                       759
            63                       158
99          1                        423

[130 rows x 1 columns]
```

The groupby table above shows that for each region_code, there can be multiple districts. The district codes are not unique between regions.

lga

```
In [103]: data_check(train, 'lga')
```

```
There are 0  null values.
There are 125  unique values.
```

lga has 125 unique values with no missing data.

```
In [75]: train[['date_recorded', 'region_code', 'district_code', 'lga']].groupby(['region_code
```

```
Out[75]:                                       date_recorded
        region_code lga          district_code
        1           Bahi         6                       224
                    Chamwino     4                       347
                    Dodoma Urban 5                       358
```

	Kondoa	1	523
	Kongwa	3	361
	Mpwapwa	0	23
		1	365
2	Arusha Rural	2	1206
		3	46
	Arusha Urban	3	63
	Longido	6	310
	Meru	7	1009
	Monduli	1	189
	Ngorongoro	5	201
3	Hai	5	617
		6	8
	Moshi Rural	1	7
		4	1219
		5	3
		6	22
	Moshi Urban	6	79
	Mwanga	2	519
	Rombo	1	588
		4	6
	Same	3	877
	Siha	7	434
4	Handeni	6	254
	Kilindi	7	127
	Korogwe	1	4
		2	408
...			...
20	Musoma Rural	3	396
	Rorya	6	210
	Serengeti	2	716
	Tarime	1	171
		6	38
21	Babati	1	511
	Hanang	2	274
	Kiteto	4	7
		5	186
	Mbulu	3	297
	Simanjiro	1	39
		4	269
24	Karatu	30	326
40	Mkuranga	43	1
60	Kisarawe	33	115
	Mafia	60	63
		63	37
		67	6
	Mkuranga	43	350
	Rufiji	53	454

```
80          Kilwa        13                  391
            Lindi Rural  23                  293
                         62                   88
            Lindi Urban  62                   21
            Liwale       43                  154
            Ruangwa      53                  291
90          Masasi       33                  528
            Nanyumbu     63                  158
            Newala       33                  231
99          Mtwara Rural 1                   423

[176 rows x 1 columns]
```

```
In [42]: plot_hist(train, col='lga')
```

ward

```
In [104]: data_check(train, 'ward')
```

```
There are 0  null values.
There are 2092  unique values.
```

`ward` has 2092 unique values with no missing data.

population

```
In [105]: data_check(train, 'population')
```

```
There are 0  null values.
There are 1049  unique values.
```

```
In [83]: plot_hist(train, col='population', ylog=True)
```

There are 1049 unique values of `population` with 0 missing data. This number is different than the unique number of water pumps, suggesting that multiple pumps can serve the same population, or there are multiple populations with the same number of people.

public meeting

```
In [109]: data_check(train, 'public_meeting')
```

```
There are 3334  null values.
There are 2  unique values.
```

```
In [107]: train.public_meeting.value_counts()
```

```
Out[107]: True     51011
          False     5055
          Name: public_meeting, dtype: int64
```

`public_meeting` is a binary feature. 3,334 values are missing for this feature.
No description is provided for this feature.

recorded_by

```
In [110]: data_check(train, 'recorded_by')
```

```
There are 0  null values.
There are 1  unique values.
```

```
In [111]: train.recorded_by.unique()
```

```
Out[111]: array(['GeoData Consultants Ltd'], dtype=object)
```

All of the data was recorded by GeoData Consultants. This value is the same for the entire dataset and thus can be dropped.

scheme_management

```
In [112]: data_check(train, 'scheme_management')
```

```
There are 3877  null values.
There are 12  unique values.
```

```
In [113]: plot_hist(train, col='scheme_management')
```

scheme_name

```
In [115]: data_check(train, 'scheme_name')
```

```
There are 28166  null values.
There are 2696  unique values.
```

permit

```
In [116]: data_check(train, 'permit')
```

```
There are 3056  null values.
There are 2  unique values.
```

```
In [117]: train.permit.value_counts()
```

```
Out[117]: True     38852
          False    17492
          Name: permit, dtype: int64
```

construction_year

```
In [118]: data_check(train, 'construction_year')
```

```
There are 0  null values.
There are 55  unique values.
```

```
In [121]: train.construction_year.value_counts()
```

```
Out[121]: 0       20709
          2010     2645
          2008     2613
          2009     2533
          2000     2091
          2007     1587
          2006     1471
          2003     1286
          2011     1256
          2004     1123
          2012     1084
          2002     1075
          1978     1037
          1995     1014
          2005     1011
          1999      979
          1998      966
          1990      954
          1985      945
          1980      811
          1996      811
          1984      779
          1982      744
          1994      738
          1972      708
          1974      676
          1997      644
          1992      640
          1993      608
          2001      540
          1988      521
          1983      488
          1975      437
          1986      434
          1976      414
          1970      411
          1991      324
          1989      316
          1987      302
```

```
1981     238
1977     202
1979     192
1973     184
2013     176
1971     145
1960     102
1967      88
1963      85
1968      77
1969      59
1964      40
1962      30
1961      21
1965      19
1966      17
Name: construction_year, dtype: int64
```

There are no null values for `construction_year`, however 20,709 of the waterpoints have a construction year of 0. This likely means that the year of construction year is not known for the water pump.

extraction_type

```
In [122]: data_check(train, 'extraction_type')
```

```
There are 0  null values.
There are 18  unique values.
```

```
In [123]: plot_hist(train, col='extraction_type')
```

There are 18 unique extraction types with no missing data. The most common is the gravity extraction type with 26,780 water pumps of this type.

extraction_type_group

```
In [124]: data_check(train, 'extraction_type_group')
```

```
There are 0  null values.
There are 13  unique values.
```

```
In [125]: plot_hist(train, col='extraction_type_group')
```

extraction_type_class

```
In [126]: data_check(train, 'extraction_type_class')
```

```
There are 0  null values.
There are 7  unique values.
```

```
In [127]: plot_hist(train, col='extraction_type_class')
```

```
In [131]: train[['extraction_type_class', 'extraction_type_group', 'extraction_type', 'date_re
```

```
Out[131]:                                                                             date_recorded
          extraction_type_class extraction_type_group extraction_type
          gravity               gravity               gravity                             26780
          handpump              afridev               afridev                              1770
                                india mark ii         india mark ii                        2400
                                india mark iii        india mark iii                         98
                                nira/tanira           nira/tanira                          8154
                                other handpump        other - mkulima/shinyanga               2
                                                      other - play pump                      85
                                                      other - swn 81                        229
                                                      walimi                                 48
                                swn 80                swn 80                               3670
          motorpump             mono                  mono                                 2865
                                other motorpump       cemo                                   90
                                                      climax                                 32
          other                 other                 other                                6430
          rope pump             rope pump             other - rope pump                     451
          submersible           submersible           ksb                                  1415
                                                      submersible                          4764
          wind-powered          wind-powered          windmill                              117
```

The groupby above shows the breakdown from extraction class to group to type. Since class and group add no new information that's not already covered by `extraction_type`, we can drop these two extra features.

management

```
In [129]: data_check(train, 'management')
```

```
There are 0  null values.
There are 12  unique values.
```

```
In [128]: plot_hist(train, col='management')
```

management_group

```
In [132]: data_check(train, 'management_group')
```

```
There are 0  null values.
There are 5  unique values.
```

```
In [133]: plot_hist(train, col='management_group')
```

```
In [170]: train[['management_group', 'management', 'date_recorded']].groupby(['management_grou
```

```
Out[170]:                                   date_recorded
          management_group management
          commercial       company                    685
                           private operator          1971
                           trust                       78
                           water authority            904
          other            other                      844
                           other - school              99
          parastatal       parastatal                1768
          unknown          unknown                    561
          user-group       vwc                      40507
                           water board               2933
                           wua                       2535
                           wug                       6515
```

payment

```
In [137]: data_check(train, 'payment')
```

```
There are 0  null values.
There are 7  unique values.
```

```
In [138]: plot_hist(train, col='payment')
```

payment_type

```
In [139]: data_check(train, 'payment_type')
```

```
There are 0  null values.
There are 7  unique values.
```

```
In [140]: plot_hist(train, col='payment_type')
```

```
In [142]: train[['payment_type', 'payment', 'date_recorded']].groupby(['payment_type', 'paymen
```

```
Out[142]:                                   date_recorded
          payment_type payment
          annually     pay annually                  3642
          monthly      pay monthly                   8300
          never pay    never pay                    25348
          on failure   pay when scheme fails         3914
          other        other                         1054
          per bucket   pay per bucket                8985
          unknown      unknown                       8157
```

payment_type and payment are duplicate columns. We can drop one of them.

water_quality

```
In [144]: data_check(train, 'water_quality')
```

```
There are 0  null values.
There are 8  unique values.
```

```
In [145]: plot_hist(train, col='water_quality')
```

quality_group

```
In [146]: data_check(train, 'quality_group')
```

```
There are 0  null values.
There are 6  unique values.
```

```
In [147]: plot_hist(train, col='quality_group')
```

```
In [149]: train[['quality_group', 'water_quality', 'date_recorded']].groupby(['quality_group',
```

```
Out[149]:                                   date_recorded
          quality_group water_quality
          colored       coloured                      490
          fluoride      fluoride                      200
                        fluoride abandoned             17
          good          soft                        50818
          milky         milky                         804
          salty         salty                        4856
                        salty abandoned               339
          unknown       unknown                      1876
```

Again, quality_group gives us no new information so it can be dropped.

quantity

```
In [151]: data_check(train, 'quantity')
```

```
There are 0  null values.
There are 5  unique values.
```

```
In [152]: plot_hist(train, col='quantity')
```

quantity_group

```
In [153]: data_check(train, 'quantity_group')
```

```
There are 0  null values.
There are 5  unique values.
```

```
In [155]: train[['quantity_group', 'quantity', 'date_recorded']].groupby(['quantity_group', 'q
```

```
Out[155]:                                date_recorded
          quantity_group quantity
          dry            dry                      6246
          enough         enough                  33186
          insufficient   insufficient            15129
          seasonal       seasonal                 4050
          unknown        unknown                   789
```

```
In [173]: (train.quantity_group==train.quantity).all()
```

```
Out[173]: True
```

`quantity_group` and `quantity` are also duplicate features, we can drop `quantity_group`.

source

```
In [156]: data_check(train, 'source')
```

```
There are 0  null values.
There are 10  unique values.
```

```
In [157]: plot_hist(train, col='source')
```

source_type

```
In [158]: data_check(train, 'source_type')
```

```
There are 0  null values.
There are 7  unique values.
```

```
In [159]: plot_hist(train, col='source_type')
```

source_class

```
In [161]: data_check(train, 'source_class')
```

```
There are 0  null values.
There are 3  unique values.
```

```
In [162]: plot_hist(train, col='source_class')
```

```
In [163]: train[['source_class', 'source_type', 'source', 'date_recorded']].groupby(['source_c
```

```
Out[163]:                                                       date_recorded
          source_class source_type          source
          groundwater  borehole             hand dtw                      874
                                            machine dbh                 11075
                       shallow well         shallow well                16824
                       spring               spring                      17021
          surface      dam                  dam                           656
                       rainwater harvesting rainwater harvesting         2295
                       river/lake           lake                          765
                                            river                        9612
          unknown      other                other                         212
                                            unknown                        66
```

waterpoint_type

```
In [165]: data_check(train, 'waterpoint_type')
```

```
There are 0  null values.
There are 7  unique values.
```

```
In [166]: plot_hist(train, col='waterpoint_type')
```

waterpoint_type_group

```
In [167]: data_check(train, 'waterpoint_type_group')
```

```
There are 0  null values.
There are 6  unique values.
```

```
In [168]: plot_hist(train, col='waterpoint_type_group')
```

```
In [169]: train[['waterpoint_type_group', 'waterpoint_type', 'date_recorded']].groupby(['water
```

Out[169]:

```
                                                 date_recorded
waterpoint_type_group waterpoint_type
cattle trough         cattle trough                        116
communal standpipe    communal standpipe                 28522
                      communal standpipe multiple         6103
dam                   dam                                    7
hand pump             hand pump                          17488
improved spring       improved spring                      784
other                 other                               6380
```

At this point we saw several features that are related to each other and are broken down into increasing granularity. We saw this for `waterpoint_type`, `source`, `water_quality`, `extraction_type`, and `management`. It's unclear to me whether the machine learning models would perform best if I was to keep only the most granular features and drop the higher categories. My plan is to create two datasets, one that keeps all of the levels of these features and one that keeps only the most granular and compare their performance.