Contents

1 Business Problem:

1.1 Background

In Tanzania Half of the population does not have acce
water pumps through coordination of various charitab
others require repair.

1.2 Problem Statement

We need to develop a ternary classification model to
repairs, and which don't work at all. An understanding
potable water is available to Tanzanian communities.

When examining what causes a waterpoint to fail we

- What kind of pump is operating
- When it was installed
- Where it was installed
- Who installed it
- How it is managed

2 Import Libraries

Contents

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.ticker as mtick
import seaborn as sns
import numpy as np
import scipy.stats as stats
import statsmodels.api as sm
# import xgboost as xgb
import catboost
import time
import warnings
warnings.filterwarnings('ignore')

from sklearn.utils import class_weight
from sklearn.metrics import accuracy_scc
from catboost import Pool, sum_models
from catboost import CatBoostClassifier
from statsmodels.formula.api import ols
from sklearn.feature_selection import RF
from sklearn.linear_model import LinearF
from sklearn.metrics import mean_squarec
from sklearn.model_selection import KFol
from sklearn.preprocessing import LabelE
from sklearn.tree import DecisionTreeReg
from sklearn.ensemble import RandomFores
from sklearn.ensemble import RandomFores
from sklearn.model_selection import trai
from sklearn.preprocessing import Standa

from sklearn.preprocessing import MinMax
from sklearn.model_selection import trai
from sklearn.metrics import classificati
from sklearn.ensemble import GradientBoc
from sklearn.linear_model import Logisti
from sklearn.model_selection import Gric
from sklearn import metrics
from sklearn.model_selection import Ranc
from scipy.stats import uniform, truncnc
from sklearn.preprocessing import OneHot
from sklearn.preprocessing import LabelE
```

3 Data Exploration

In [2]:
```
df_test = pd.read_csv('data/Test-set-val
```

In [3]:
```
df_train_set = pd.read_csv('data/Trainir
df_train_labels = pd.read_csv('data/Trai
```

3.1 Data Fields

Predict one of the three classes based on a number c

Contents

The following set of information about waterpoints is

- amount_tsh — Total static head (zero for open ta
- 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 c
- num_private — No information
- 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 wate
- scheme_name — Who operates the waterpoint
- permit — If the waterpoint is permitted
- construction_year — Year the waterpoint was co
- extraction_type — The kind of extraction the wat
- extraction_type_group — The kind of extraction t
- extraction_type_class — The kind of extraction tl
- management — How the waterpoint is managed
- management_group — How the waterpoint is ma
- 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 (duplicat
- 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

Contents

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

Out[4]:

	id	amount_tsh	date_recorded	funder	gps_l
0	69572	6000.0	2011-03-14	Roman	
1	8776	0.0	2013-03-06	Grumeti	
2	34310	25.0	2013-02-25	Lottery Club	
3	67743	0.0	2013-01-28	Unicef	
4	19728	0.0	2011-07-13	Action In A	

5 rows × 40 columns

In [5]:
```
df_train_set.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 59400 entries, 0 to 59399
Data columns (total 40 columns):
 #   Column                 Non-Null Cou
---  ------                 ------------
 0   id                     59400 non-nu
 1   amount_tsh             59400 non-nu
 2   date_recorded          59400 non-nu
 3   funder                 55765 non-nu
 4   gps_height             59400 non-nu
 5   installer              55745 non-nu
 6   longitude              59400 non-nu
 7   latitude               59400 non-nu
 8   wpt_name               59400 non-nu
 9   num_private            59400 non-nu
 10  basin                  59400 non-nu
 11  subvillage             59029 non-nu
 12  region                 59400 non-nu
 13  region_code            59400 non-nu
```

In [6]:
```
df_train_labels['status_group'].value_co
```

Out[6]:
```
functional                 0.543081
non functional             0.384242
functional needs repair    0.072677
Name: status_group, dtype: float64
```

Contents

In [7]:
```
df_train_labels.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 59400 entries, 0 to 59399
Data columns (total 2 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   id            59400 non-null  int64
 1   status_group  59400 non-null  objec
dtypes: int64(1), object(1)
memory usage: 928.2+ KB
```

In [8]:
```
df_train_labels['status_group'].value_cc
```

Out[8]:
```
functional                 0.543081
non functional             0.384242
functional needs repair    0.072677
Name: status_group, dtype: float64
```

- Most wells are either functional or non functional

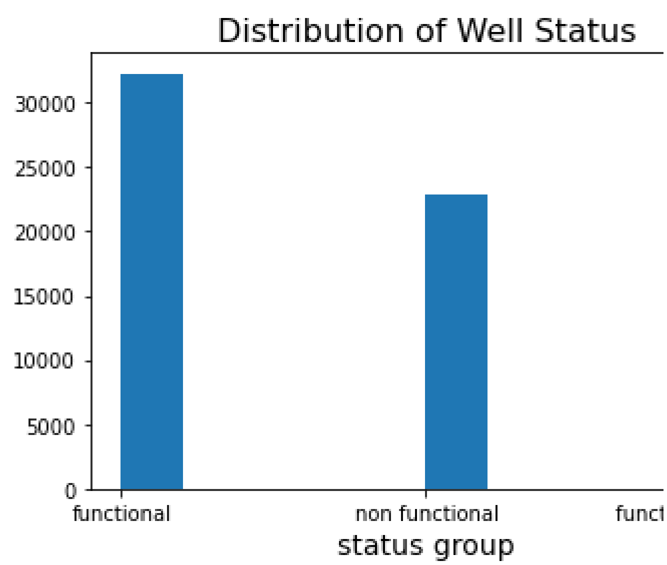
In [9]:
```
# print top 5 most frequent values in ea
for col in df_train_set.columns:
    print(col, '\n', df_train_set[col].v
```

```
id
 2047     0.000017
72310    0.000017
49805    0.000017
51852    0.000017
62091    0.000017
Name: id, dtype: float64


amount_tsh
 0.0       0.700993
500.0     0.052222
50.0      0.041616
1000.0    0.025051
20.0      0.024630
Name: amount_tsh, dtype: float64


date_recorded
```

- amount_tsh: majority are zeros (70%)
- gps_height: top value is zero (34%)
- longitude: top value is zero (3%) these are not va
- region and region_code: remove region
- wpt_name: top value is none. (5%)
- num_private: top value is zero (98%)

- population: top value is zero (36%)
- recorded_by: only one value can remove
- construction_year: top value is zero (34%)
- extraction_type and extraction_type_field are the
- quality_group looks to be a replacement for wate
- quantity and quantity_group look the same
- source vs source_type vs source_class: source_c
- waterpoint_type vs waterpoint_type_group: wate

In [10]:
```
# Plot of the target status group

plt.hist(df_train_labels['status_group']
plt.xlabel('status group', fontsize=14)
plt.title("Distribution of Well Status",
plt.show()
```



Here we see a strong class imbalance that will need t

4 Data Preparation

4.1 Missing Values

4.1.1 permit

In [11]:
```
df_train_set['permit'].value_counts(norm
```

Out[11]:
```
True      0.654074
False     0.294478
NaN       0.051448
Name: permit, dtype: float64
```

Contents

In [12]:
```
# df_train_set.subvillage.fillna(0, inpl
df_train_set.permit.fillna(0, inplace=Tr
# df_train_set.public_meeting.fillna(0,
# df_train_set.scheme_management.fillna(
```

In [13]:
```
df_train_set['permit'].value_counts(norm
```

Out[13]:
```
True     0.654074
False    0.345926
Name: permit, dtype: float64
```

In [14]:
```
#Make Permit boolean
df_train_set['permit'] = df_train_set['p
df_train_set['permit'].value_counts(norm
```

Out[14]:
```
1    0.654074
0    0.345926
Name: permit, dtype: float64
```

4.1.2 construction_year

Contents

In [15]:
```
# Close to 35% are zero lets replace thi
df_train_set['construction_year'].value_
```

Out[15]:
```
0       0.348636
2010    0.044529
2008    0.043990
2009    0.042643
2000    0.035202
2007    0.026717
2006    0.024764
2003    0.021650
2011    0.021145
2004    0.018906
2012    0.018249
2002    0.018098
1978    0.017458
1995    0.017071
2005    0.017020
1999    0.016481
1998    0.016263
1990    0.016061
1985    0.015909
1980    0.013653
1996    0.013653
1984    0.013114
1982    0.012525
1994    0.012424
1972    0.011919
1974    0.011380
1997    0.010842
1992    0.010774
1993    0.010236
2001    0.009091
1988    0.008771
1983    0.008215
1975    0.007357
1986    0.007306
1976    0.006970
1970    0.006919
1991    0.005455
1989    0.005320
1987    0.005084
1981    0.004007
1977    0.003401
1979    0.003232
1973    0.003098
2013    0.002963
1971    0.002441
1960    0.001717
1967    0.001481
1963    0.001431
1968    0.001296
1969    0.000993
1964    0.000673
1962    0.000505
1961    0.000354
1965    0.000320
```

Contents

```
1966    0.000286
Name: construction_year, dtype: float64
```

In [17]:
```
nonzero_median_year = df_train_set[ df_t
```

In [18]:
```
nonzero_median_year['construction_year']
nonzero_median_year['construction_year']
```

Out[18]:
```
2000.0
```

Contents

Contents

In [20]:
```
df_train_set['construction_year'].value_
```

Out[20]:
```
0       0.348636
2010    0.044529
2008    0.043990
2009    0.042643
2000    0.035202
2007    0.026717
2006    0.024764
2003    0.021650
2011    0.021145
2004    0.018906
2012    0.018249
2002    0.018098
1978    0.017458
1995    0.017071
2005    0.017020
1999    0.016481
1998    0.016263
1990    0.016061
1985    0.015909
1980    0.013653
1996    0.013653
1984    0.013114
1982    0.012525
1994    0.012424
1972    0.011919
1974    0.011380
1997    0.010842
1992    0.010774
1993    0.010236
2001    0.009091
1988    0.008771
1983    0.008215
1975    0.007357
1986    0.007306
1976    0.006970
1970    0.006919
1991    0.005455
1989    0.005320
1987    0.005084
1981    0.004007
1977    0.003401
1979    0.003232
1973    0.003098
2013    0.002963
1971    0.002441
1960    0.001717
1967    0.001481
1963    0.001431
1968    0.001296
1969    0.000993
1964    0.000673
1962    0.000505
1961    0.000354
1965    0.000320
```

```
1966    0.000286
Name: construction_year, dtype: float64
```

4.2 Replace mispellings and gro

4.2.1 Reformat Installer col

In [21]:
```
df_train_set['installer'] = df_train_set
df_train_set['installer'].replace(to_rep
```

In [22]:
```
df_train_set['installer'].value_counts(r
```

Out[22]:
```
dwe                 0.293013
other               0.074663
government          0.031835
hesawa              0.023485
rwe                 0.020303
                      ...
kkkt ndrumangeni    0.000017
safari roya         0.000017
tsrc                0.000017
magul               0.000017
hemed abdallah      0.000017
Name: installer, Length: 1934, dtype: fl
```

In [23]:
```
df_clean=pd.read_csv('data/lookups.csv')
df_train_set = pd.merge(df_train_set, df
df_train_set.head()
```

Out[23]:

	id	amount_tsh	date_recorded	funder	gps_h
0	69572	6000.0	2011-03-14	Roman	
1	8776	0.0	2013-03-06	Grumeti	
2	34310	25.0	2013-02-25	Lottery Club	
3	67743	0.0	2013-01-28	Unicef	
4	19728	0.0	2011-07-13	Action In A	

5 rows × 42 columns

In [24]:
```
df_train_set['installer_new'].isna().sum
```

Out[24]:
```
44958
```

Contents

In [25]:
```
df_train_set['installer'].value_counts(r
```

Out[25]:
```
dwe                   0.293013
other                 0.074663
government            0.031835
hesawa                0.023485
rwe                   0.020303
                        ...
kkkt ndrumangeni      0.000017
safari roya           0.000017
tsrc                  0.000017
magul                 0.000017
hemed abdallah        0.000017
Name: installer, Length: 1934, dtype: fl
```

In [26]:
```
df_train_set['installer_new'].fillna(df_

df_train_set
```

Out[26]:

	id	amount_tsh	date_recorded	funder
0	69572	6000.0	2011-03-14	Roman
1	8776	0.0	2013-03-06	Grumeti
2	34310	25.0	2013-02-25	Lottery Club
3	67743	0.0	2013-01-28	Unicef
4	19728	0.0	2011-07-13	Action In A
...	...	...	...	...
59395	60739	10.0	2013-05-03	Germany Republi
59396	27263	4700.0	2011-05-07	Cefa-njombe
59397	37057	0.0	2011-04-11	NaN
59398	31282	0.0	2011-03-08	Malec
59399	26348	0.0	2011-03-23	World Bank

59400 rows × 42 columns

In [27]:
```
df_train_set['installer_new'].isna().sun
```

Out[27]: 0

Contents

Contents

In [28]:
```
df_train_set['installer_new'].value_cour
```

Out[28]:
```
dwe                              0.304983
other                            0.074663
tanzanian government             0.062323
community                        0.033266
danida                           0.028300
                                   ...
b.a.p                            0.000017
lindi contractor                 0.000017
ji                               0.000017
friedkin conservation fund       0.000017
hemed abdallah                   0.000017
Name: installer_new, Length: 1652, dtype
```

In [29]:
```
del df_train_set['installer']
del df_train_set['installer_old']
```

In [30]:
```
df_installer_cnt = df_train_set['installl
df_installer_cnt
```

Out[30]:
```
dwe                              0.304983
other                            0.074663
tanzanian government             0.062323
community                        0.033266
danida                           0.028300
                                   ...
b.a.p                            0.000017
lindi contractor                 0.000017
ji                               0.000017
friedkin conservation fund       0.000017
hemed abdallah                   0.000017
Name: installer_new, Length: 1652, dtype
```

In [31]:
```
df_installer_cnt.head(20)
```

Out[31]:
```
dwe                     0.304983
other                   0.074663
tanzanian government    0.062323
community               0.033266
danida                  0.028300
hesawa                  0.023620
council                 0.022929
rwe                     0.020303
church                  0.016330
kkkt                    0.015623
finw                    0.013199
tcrs                    0.012458
world vision            0.012037
ces                     0.010269
amref                   0.007458
lga                     0.006953
tasaf                   0.006919
wedeco                  0.006700
dmdd                    0.006330
```

Contents

In [32]:
```
other_list = df_installer_cnt[df_install
other_list
```

```
 'unicef',
 'twesa',
 'da',
 'wu',
 'acra',
 'sema',
 'jaica',
 'oxfam',
 'shipo',
 'local',
 'idara ya maji',
 'villagers',
 'sengerema water department',
 'kiliwater',
 'dh',
 'kuwait',
 'distri',
 'lawatefuka water sup',
 'magadini-makiwaru wa',
```

In [33]:
```
df_train_set['installer_new'].replace(tc
df_train_set['installer_new'].value_cour
```

Out[33]:
```
other                    0.424360
dwe                      0.304983
tanzanian government     0.062323
community                0.033266
danida                   0.028300
hesawa                   0.023620
council                  0.022929
rwe                      0.020303
church                   0.016330
kkkt                     0.015623
finw                     0.013199
tcrs                     0.012458
world vision             0.012037
ces                      0.010269
Name: installer_new, dtype: float64
```

In [34]:
```
df_train_set.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 59400 entries, 0 to 59399
Data columns (total 40 columns):
 #   Column                 Non-Null Cou
---  ------                 ------------
 0   id                     59400 non-nu
 1   amount_tsh             59400 non-nu
 2   date_recorded          59400 non-nu
 3   funder                 55765 non-nu
 4   gps_height             59400 non-nu
 5   longitude              59400 non-nu
 6   latitude               59400 non-nu
 7   wpt_name               59400 non-nu
 8   num_private            59400 non-nu
 9   basin                  59400 non-nu
 10  subvillage             59029 non-nu
 11  region                 59400 non-nu
 12  region_code            59400 non-nu
 13  district_code          59400 non-nu
 14  lga                    59400 non-nu
 15  ward                   59400 non-nu
 16  population             59400 non-nu
 17  public_meeting         56066 non-nu
 18  recorded_by            59400 non-nu
 19  scheme_management      55523 non-nu
 20  scheme_name            31234 non-nu
 21  permit                 59400 non-nu
 22  construction_year      59400 non-nu
 23  extraction_type        59400 non-nu
 24  extraction_type_group  59400 non-nu
 25  extraction_type_class  59400 non-nu
 26  management             59400 non-nu
 27  management_group       59400 non-nu
 28  payment                59400 non-nu
 29  payment_type           59400 non-nu
 30  water_quality          59400 non-nu
 31  quality_group          59400 non-nu
 32  quantity               59400 non-nu
 33  quantity_group         59400 non-nu
 34  source                 59400 non-nu
 35  source_type            59400 non-nu
 36  source_class           59400 non-nu
 37  waterpoint_type        59400 non-nu
 38  waterpoint_type_group  59400 non-nu
 39  installer_new          59400 non-nu
dtypes: float64(3), int64(8), object(29)
memory usage: 18.6+ MB
```

4.2.2 Reformat Funder col

Contents

Contents

In [35]:
```
df_train_set['funder'] = df_train_set['f
df_train_set['funder'].replace(to_replac
```

In [36]:
```
df_train_set['funder'].value_counts(norm
```

Out[36]:
```
government of tanzania     0.152929
other                      0.074276
danida                     0.052424
hesawa                     0.037071
rwssp                      0.023131
                             ...
tlc/emmanuel kasoga        0.000017
villlage contributi        0.000017
hasnan murig (mbunge)      0.000017
samweli mshosha            0.000017
kajima                     0.000017
Name: funder, Length: 1897, dtype: float
```

In [37]:
```
df_clean_funder=pd.read_csv('data/lookup
df_train_set = pd.merge(df_train_set, df
df_train_set.head()
```

Out[37]:

	id	amount_tsh	date_recorded	funder	gps_h
0	69572	6000.0	2011-03-14	roman	
1	8776	0.0	2013-03-06	grumeti	
2	34310	25.0	2013-02-25	lottery club	
3	67743	0.0	2013-01-28	unicef	
4	19728	0.0	2011-07-13	action in a	

5 rows × 42 columns

In [38]:
```
df_train_set['funder_new'].isna().sum()
```

Out[38]: 39498

Contents

In [39]:
```
df_train_set['funder'].value_counts(norm
```

Out[39]:
```
government of tanzania    0.152929
other                     0.074276
danida                    0.052424
hesawa                    0.037071
rwssp                     0.023131
                            ...
tlc/emmanuel kasoga       0.000017
villlage contributi       0.000017
hasnan murig (mbunge)     0.000017
samweli mshosha           0.000017
kajima                    0.000017
Name: funder, Length: 1897, dtype: float
```

In [40]:
```
df_train_set['funder_new'].fillna(df_tra

df_train_set
```

Out[40]:

	id	amount_tsh	date_recorded	funder
0	69572	6000.0	2011-03-14	roman
1	8776	0.0	2013-03-06	grumeti
2	34310	25.0	2013-02-25	lottery club
3	67743	0.0	2013-01-28	unicef
4	19728	0.0	2011-07-13	action in a
...	...	...	...	...
59395	60739	10.0	2013-05-03	germany republi
59396	27263	4700.0	2011-05-07	cefa-njombe
59397	37057	0.0	2011-04-11	other
59398	31282	0.0	2011-03-08	malec
59399	26348	0.0	2011-03-23	world bank

59400 rows × 42 columns

Contents

In [41]:
```
df_train_set['funder_new'].isna().sum()
```
Out[41]:
```
0
```

In [42]:
```
df_train_set['funder_new'].value_counts(
```
Out[42]:
```
government       0.156869
other            0.074276
danida           0.052559
hesawa           0.037323
kkkt             0.026027
                   ...
chongolo         0.000017
lgdbg            0.000017
petro patrice    0.000017
mwakalinga       0.000017
quicklw          0.000017
Name: funder_new, Length: 1683, dtype: f
```

In [43]:
```
del df_train_set['funder']
del df_train_set['funder_old']
```

In [44]:
```
df_installer_cnt_funder = df_train_set['
df_installer_cnt_funder
```
Out[44]:
```
government       0.156869
other            0.074276
danida           0.052559
hesawa           0.037323
kkkt             0.026027
                   ...
chongolo         0.000017
lgdbg            0.000017
petro patrice    0.000017
mwakalinga       0.000017
quicklw          0.000017
Name: funder_new, Length: 1683, dtype: f
```

In [45]:
```
df_installer_cnt_funder.head(20)
```

Out[45]:
```
government            0.156869
other                 0.074276
danida                0.052559
hesawa                0.037323
kkkt                  0.026027
district council      0.025051
church                0.024529
rwssp                 0.023131
world bank            0.022710
world vision          0.021498
unicef                0.019798
tasaf                 0.014764
dhv                   0.013956
private individual    0.013906
dwsp                  0.013653
norad                 0.012879
finw                  0.012559
german group          0.012492
tcrs                  0.010455
ministry of water     0.009933
Name: funder_new, dtype: float64
```

In [46]:
```
other_list_funder = df_installer_cnt_fur
other_list_funder
```

```
 'halmashaur',
 'br',
 'professor ben ohio university',
 'member of parliament',
 'i.e.c',
 'lawate fuka water suppl',
 'abd',
 'sauwasa',
 'mws',
 'school',
 'parastatal',
 'menon',
 'summit for water',
 'rwsssp',
 'local',
 'unicef/ csp',
 'africare',
 'hapa',
 'el',
```

Contents

In [47]:
```
df_train_set['funder_new'].replace(to_re
df_train_set['funder_new'].value_counts(
```

Out[47]:
```
other                  0.485842
government             0.156869
danida                 0.052559
hesawa                 0.037323
kkkt                   0.026027
district council       0.025051
church                 0.024529
rwssp                  0.023131
world bank             0.022710
world vision           0.021498
unicef                 0.019798
tasaf                  0.014764
dhv                    0.013956
private individual     0.013906
dwsp                   0.013653
norad                  0.012879
finw                   0.012559
german group           0.012492
tcrs                   0.010455
Name: funder_new, dtype: float64
```

Contents

4.2.3 Group Other Remaining Columns

In [48]:
```
def group_other(col,perc):
    df_train_set[col] = df_train_set[col
    df_cnt = df_train_set[col].value_cou
    other_list = df_cnt[df_cnt<perc].inc


    return df_train_set[col].replace(to_
    print(df_train_set[col].value_counts
```

4.2.3.1 lga

In [49]:
```
df_train_set['lga'].value_counts(normali
```

Out[49]:
```
Njombe          0.042138
Arusha Rural    0.021077
Moshi Rural     0.021061
Bariadi         0.019815
Rungwe          0.018620
                  ...
Moshi Urban     0.001330
Kigoma Urban    0.001195
Arusha Urban    0.001061
Lindi Urban     0.000354
Nyamagana       0.000017
Name: lga, Length: 125, dtype: float64
```

Contents

In [50]:
```
group_other('lga',.01)
```

In [51]:
```
df_train_set['lga'].value_counts(normali
```

Out[51]:
```
other            0.475438
njombe           0.042138
arusha rural     0.021077
moshi rural      0.021061
bariadi          0.019815
rungwe           0.018620
kilosa           0.018418
kasulu           0.017626
mbozi            0.017407
meru             0.016987
bagamoyo         0.016785
singida rural    0.016751
kilombero        0.016145
same             0.014764
kibondo          0.014714
kyela            0.014461
kahama           0.014074
magu             0.013872
kigoma rural     0.013872
maswa            0.013620
karagwe          0.012980
mbinga           0.012626
iringa rural     0.012256
serengeti        0.012054
lushoto          0.011684
namtumbo         0.011684
songea rural     0.011667
mpanda           0.011431
mvomero          0.011296
ngara            0.011263
ulanga           0.011195
makete           0.010606
kwimba           0.010556
mbarali          0.010539
hai              0.010522
rombo            0.010000
Name: lga, dtype: float64
```

4.3 Columns to drop

4.3.1 Mostly Empty

Contents

In [52]:
```
df_train_set['num_private'].value_counts
```

Out[52]:
```
0      0.987256
6      0.001364
1      0.001229
5      0.000774
8      0.000774
         ...
180    0.000017
213    0.000017
23     0.000017
55     0.000017
94     0.000017
Name: num_private, Length: 65, dtype: fl
```

4.3.2 Many Individual Values

In [53]:
```
df_train_set['wpt_name'].value_counts(nc
```

Out[53]:
```
none                             0.059983
Shuleni                          0.029428
Zahanati                         0.013973
Msikitini                        0.009007
Kanisani                         0.005438
                                   ...
Kwa Mzee Zenubius                0.000017
Msanya                           0.000017
Kwa Mbumbuli                     0.000017
Irkisale Secondary School        0.000017
Kwa Kiambae                      0.000017
Name: wpt_name, Length: 37400, dtype: fl
```

In [54]:
```
df_train_set['ward'].value_counts(normal
```

Out[54]:
```
Igosi              0.005168
Imalinyi           0.004242
Siha Kati          0.003906
Mdandu             0.003889
Nduruma            0.003653
                     ...
Uwanja wa Ndege    0.000017
Nsemulwa           0.000017
Kitete             0.000017
Linda              0.000017
Chinugulu          0.000017
Name: ward, Length: 2092, dtype: float64
```

4.3.3 Not Significant

4.3.3.1 The features scheme_management, quant date_recorded, and recorded_by will be deleted fo

Contents

In [55]:
```
def drop(df):
# Columns to drop
    df.drop([
        'scheme_management', 'quantity_q
        'extraction_type', 'extraction_t
        'subvillage', 'public_meeting',
        axis=1, inplace=True)
    return df
```

To do: look at feat importance of different geographic

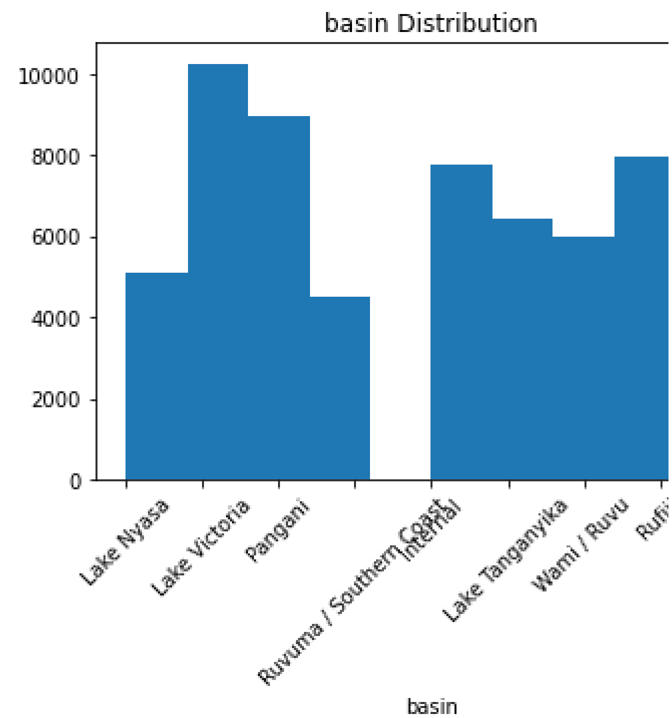
- basin — Geographic water basin
- subvillage — Geographic location
- region_code — Geographic location (coded)
- district_code — Geographic location (coded)
- lga — Geographic location
- ward — Geographic location

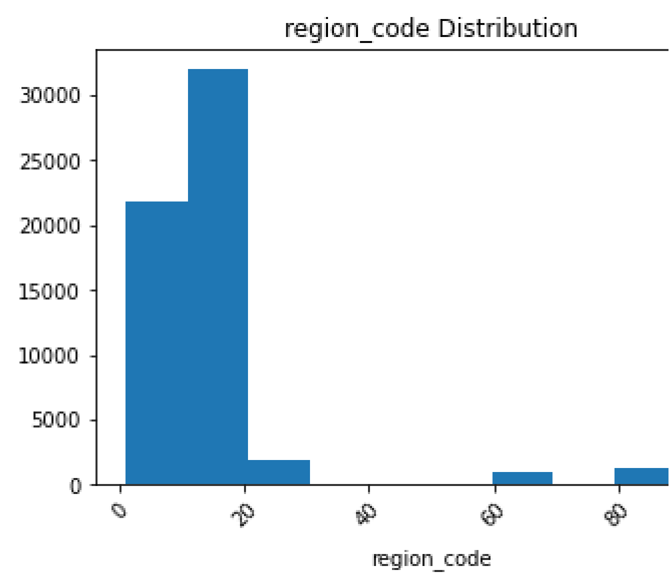
4.4 Categorical and Numerical

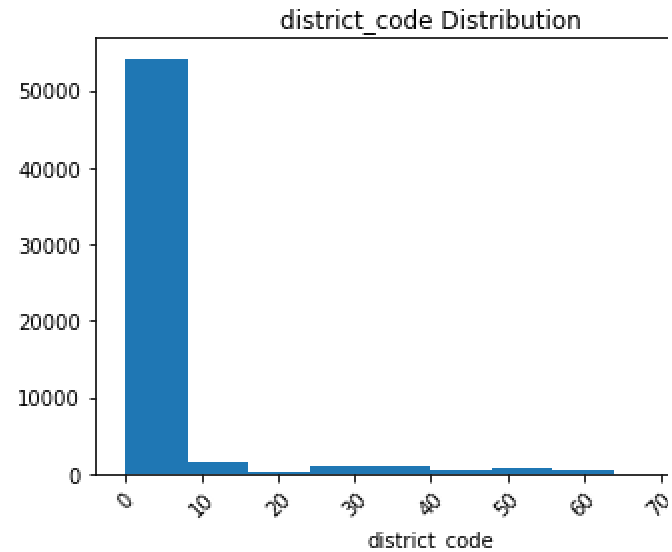
In [56]:
```
categorical = ['basin','region_code', 'c
               'extraction_type_group','
               'quality_group', 'quantit
               'installer_new', 'funder_
```

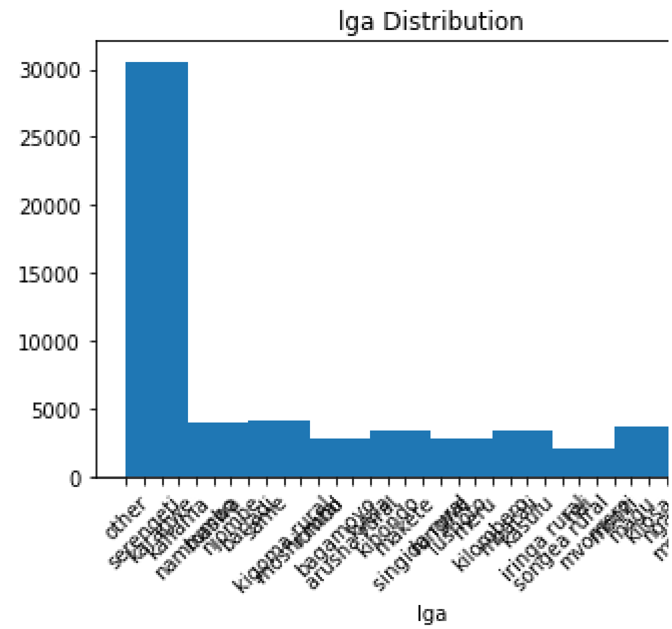
Contents

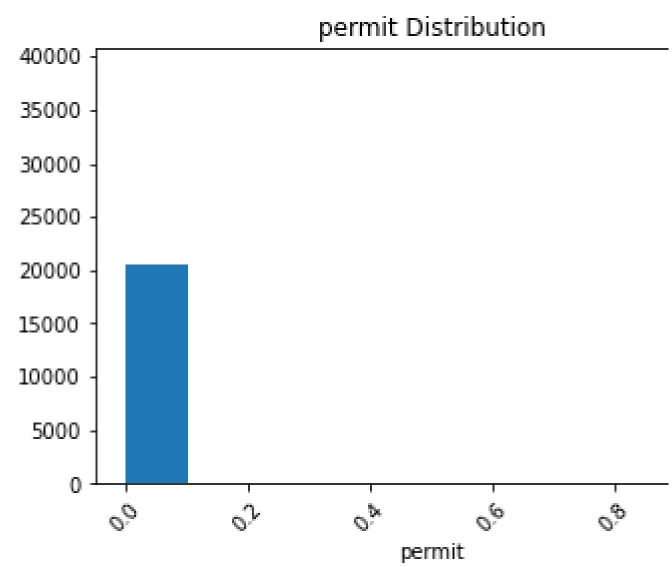
In [57]:
```
for column in categorical:
    plt.hist(df_train_set[column])
    plt.xlabel(column)
    plt.xticks(rotation=45)
    plt.title("{} Distribution".format(c
    plt.show()
```



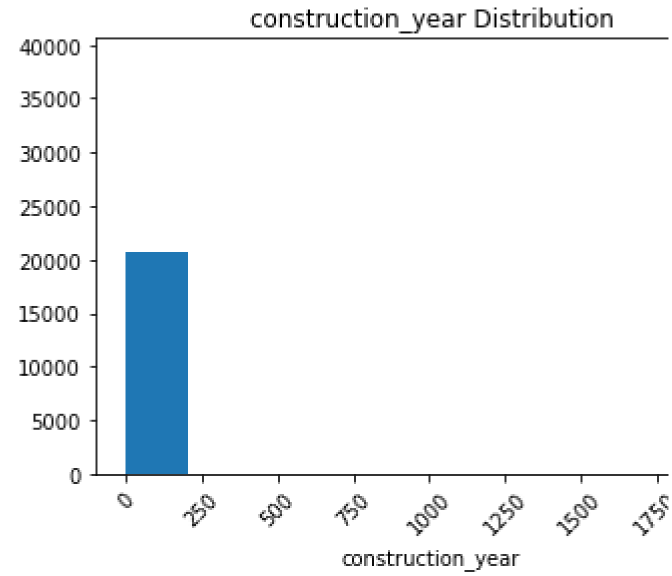


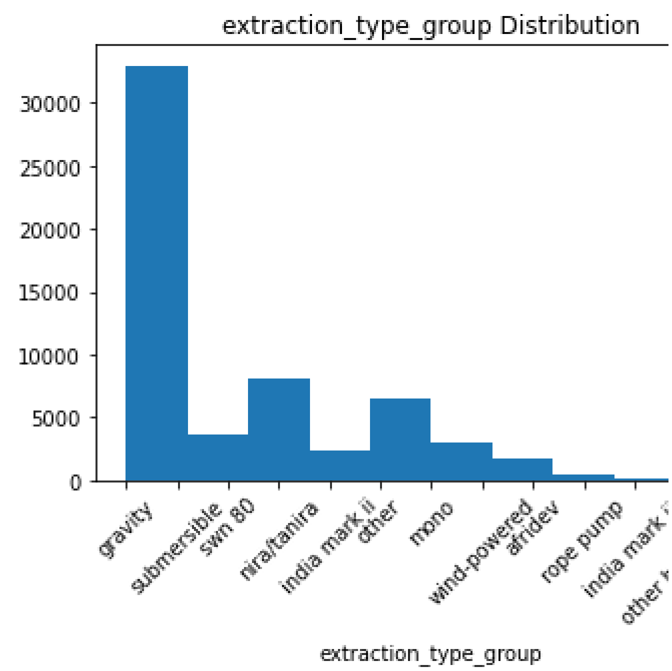

district_code Distribution
50000
40000
30000
20000
10000
0
0
10
20
30
40
50
60
70
district_code


lga Distribution
30000
25000
20000
15000
10000
5000
0
other
lga

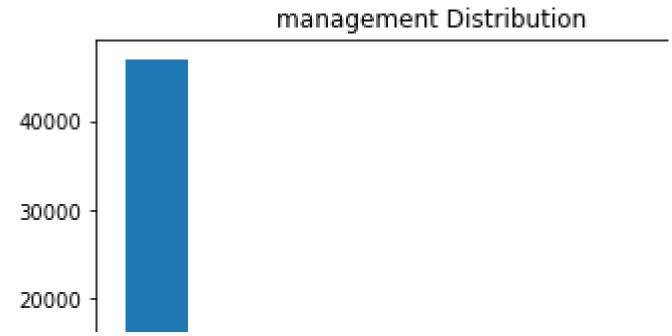

permit Distribution
40000
35000
30000
25000
20000
15000
10000
5000
0
0.0
0.2
0.4
0.6
0.8
permit

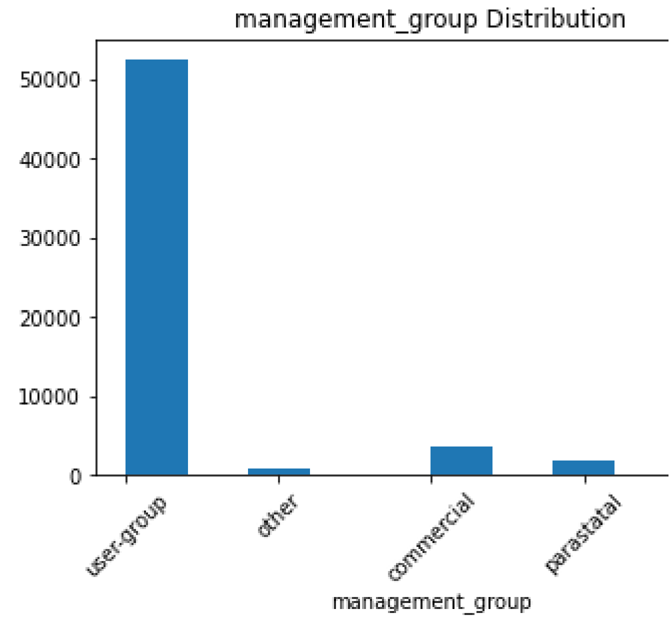
Contents

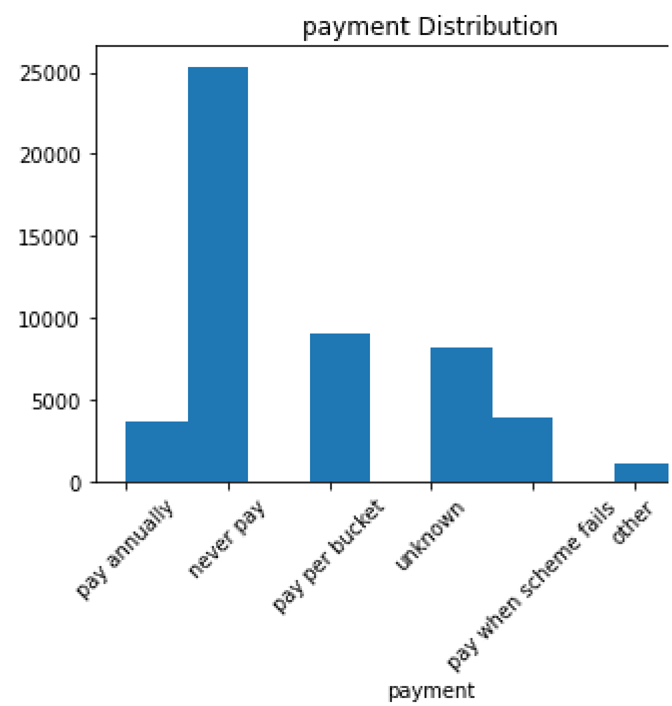

construction_year Distribution
40000
35000
30000
25000
20000
15000
10000
5000
0
0
250
500
750
1000
1250
1500
construction_year


extraction_type_group Distribution
30000
25000
20000
15000
10000
5000
0
gravity
submersible
swn 80
nira/tanira
india mark ii
other
mono
wind-powered
afridev
rope pump
extraction_type_group

Contents

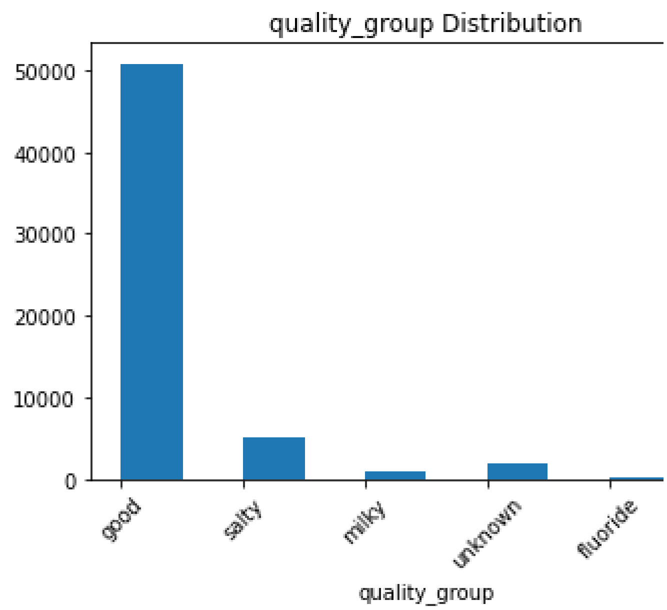

management Distribution
40000
30000
20000

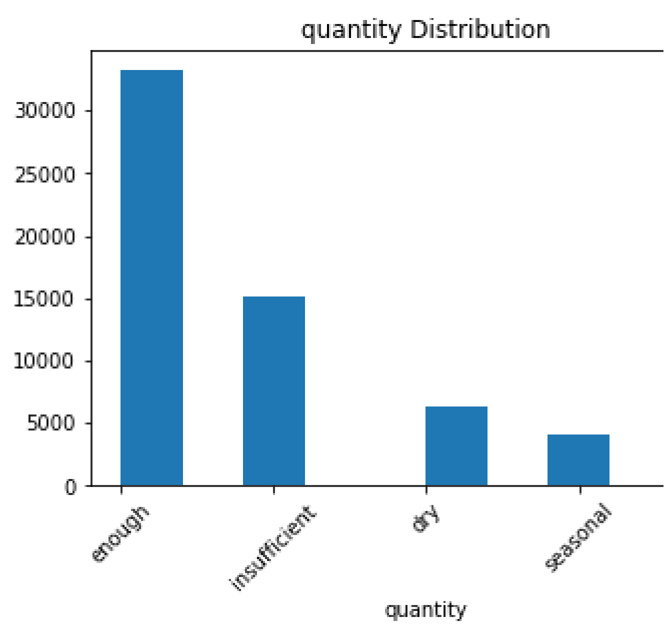

management_group Distribution
50000
40000
30000
20000
10000
0
user-group
other
commercial
parastatal
management_group

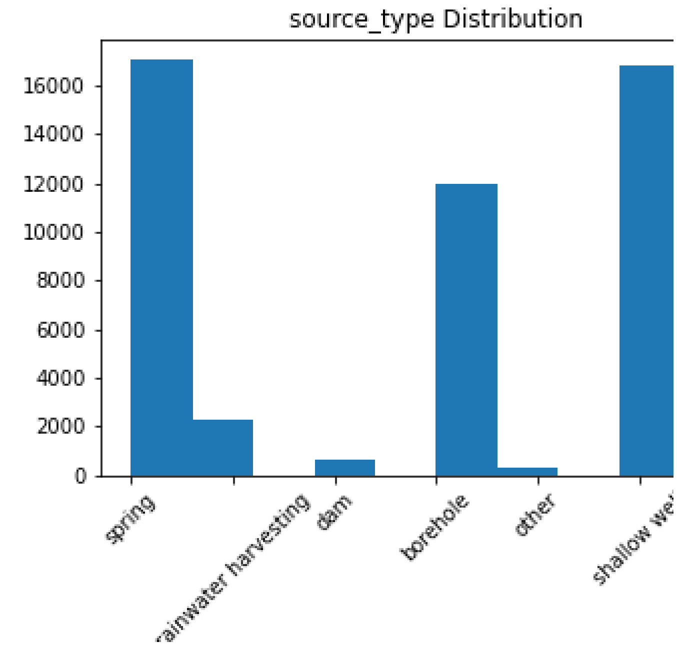

payment Distribution
25000
20000
15000
10000
5000
0
pay annually
never pay
pay per bucket
unknown
pay when scheme fails
other
payment

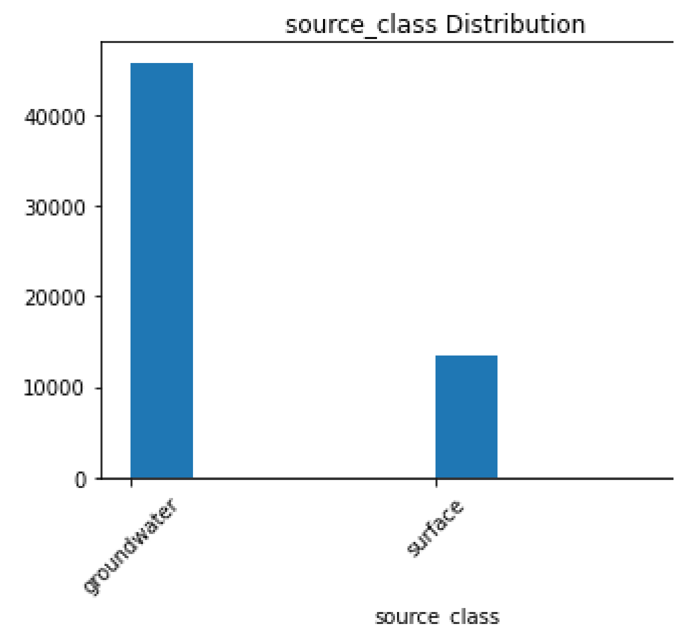
Contents

Contents


quality_group Distribution
50000
40000
30000
20000
10000
0
good
salty
milky
unknown
fluoride
quality_group

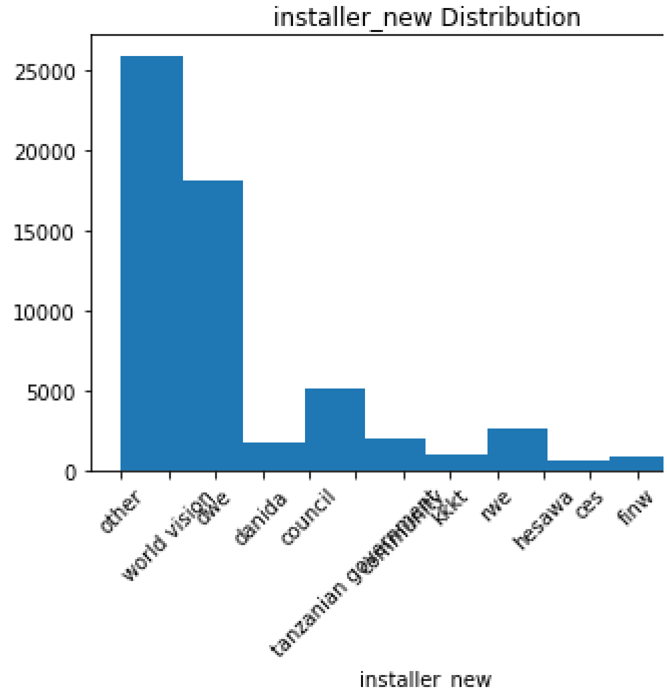

quantity Distribution
30000
25000
20000
15000
10000
5000
0
enough
insufficient
dry
seasonal
quantity

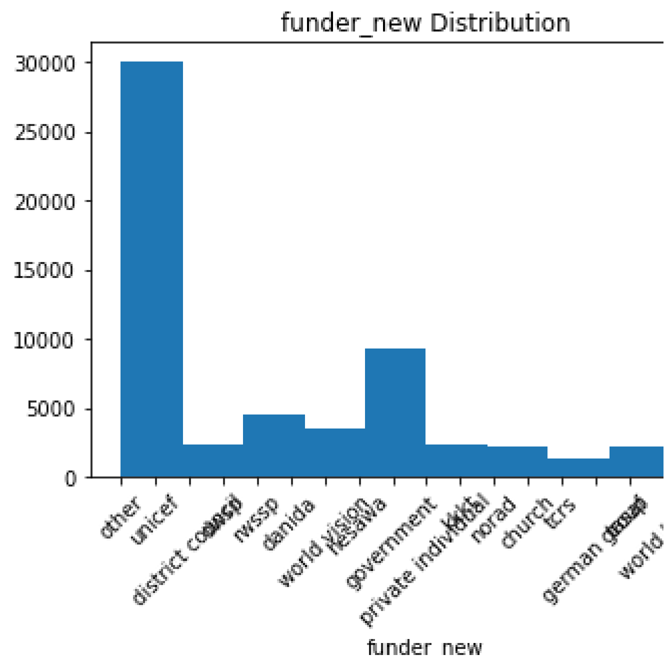

source_type Distribution
16000
14000
12000
10000
8000
6000
4000
2000
0
spring
rainwater harvesting
dam
borehole
other


source_class Distribution
40000
30000
20000
10000
0
groundwater
surface
source_class

Contents

Contents





4.4.1 Join Target: df_train_set to df_tra

Contents

In [58]:
```
df_train = df_train_set.join(df_train_la
df_train = drop(df_train)

df_test = drop(df_test)
```

In [59]:
```
df_train.info()
```
```
 8   lga                    59400 non-nu
 9   population             59400 non-nu
 10  permit                 59400 non-nu
 11  construction_year      59400 non-nu
 12  extraction_type_group  59400 non-nu
 13  management             59400 non-nu
 14  management_group       59400 non-nu
 15  payment                59400 non-nu
 16  quality_group          59400 non-nu
 17  quantity               59400 non-nu
 18  source_type            59400 non-nu
 19  source_class           59400 non-nu
 20  waterpoint_type        59400 non-nu
 21  installer_new          59400 non-nu
 22  funder_new             59400 non-nu
 23  id_right               47492 non-nu
 24  status_group           47492 non-nu
dtypes: float64(4), int64(7), object(14)
memory usage: 11.8+ MB
```

In [60]:
```
df_train.columns
```
Out[60]:
```
Index(['id_left', 'amount_tsh', 'gps_hei
       'region_code', 'district_code', '
       'construction_year', 'extraction_
       'management_group', 'payment', 'q
       'source_type', 'source_class', 'w
       'funder_new', 'id_right', 'status
      dtype='object')
```

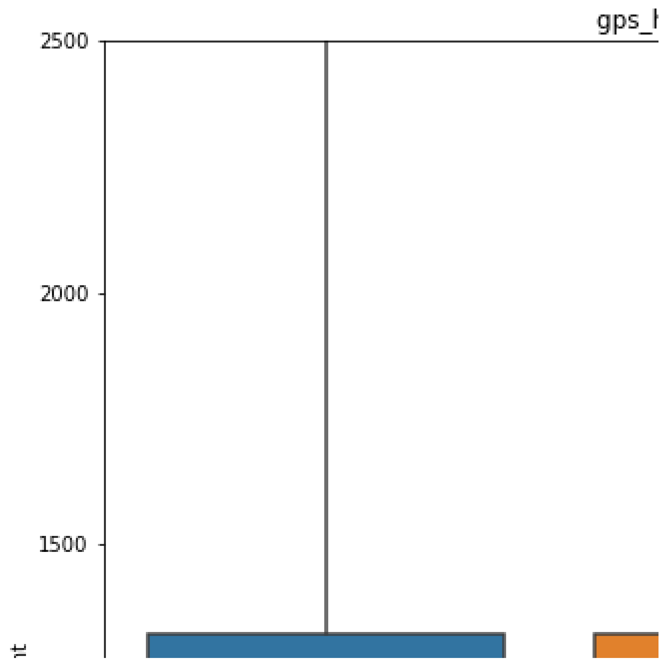
In [61]:
```
df_train['status_group'].value_counts(no
```
Out[61]:
```
functional                 0.436229
non functional             0.305522
NaN                        0.200471
functional needs repair    0.057778
Name: status_group, dtype: float64
```

In [62]:
```
numerical = ['amount_tsh','gps_height','
```

In [63]:
```
numerical_large = ['gps_height','populat
```

In [64]:
```
numerical_small = ['amount_tsh']
```

In [65]:
```
for variable in numerical_large:
    ax, figure = plt.subplots(1,1,figsiz
    plt.ylim(-100,2500)
    sns.boxplot(x='status_group', y=vari
    plt.title("{} vs. Well Condition".fc
```



In [66]:
```
del df_train['gps_height']
del df_train['population']
```

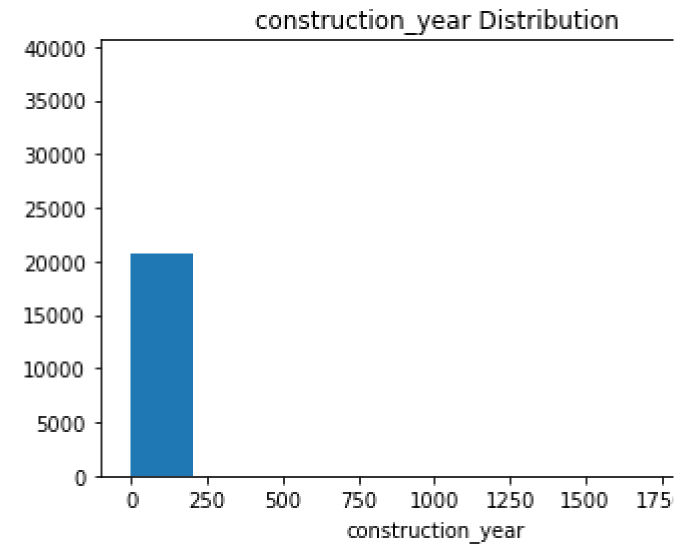
In [67]:
```
for variable in numerical_small:
    ax, figure = plt.subplots(1,1,figsiz
    plt.ylim(-1,200)
    sns.boxplot(x='status_group', y=vari
    plt.title("{} vs. Well Condition".fc
```

Contents

4.4.2 Column Binning

In [68]:
```
year_plot = ['construction_year']
```

In [69]:
```
for column in year_plot:
    plt.hist(df_train[column])
    plt.xlabel(column)
    plt.title("{} Distribution".format(c
    plt.show()
```



In [70]:
```
bin_features = ['amount_tsh','constructi
```

In [71]:
```
df_train['construction_year'].value_cour
```

Out[71]:
```
0       0.348636
2010    0.044529
2008    0.043990
2009    0.042643
2000    0.035202
2007    0.026717
2006    0.024764
2003    0.021650
2011    0.021145
2004    0.018906
2012    0.018249
2002    0.018098
1978    0.017458
1995    0.017071
2005    0.017020
1999    0.016481
1998    0.016263
1990    0.016061
1985    0.015909
```

Contents

In [72]:
```
df_train['amount_tsh'].value_counts(norm
```

Out[72]:
```
0.0         0.700993
500.0       0.052222
50.0        0.041616
1000.0      0.025051
20.0        0.024630
              ...
8500.0      0.000017
6300.0      0.000017
220.0       0.000017
138000.0    0.000017
12.0        0.000017
Name: amount_tsh, Length: 98, dtype: flo
```

In [73]:
```
df_train.amount_tsh.fillna(0, inplace=Tr
```

In [74]:
```
# and we will also take a look at their
for feat in bin_features:
    print(df_train[feat].describe())
```

```
count     59400.000000
mean        317.650385
std        2997.574558
min           0.000000
25%           0.000000
50%           0.000000
75%          20.000000
max      350000.000000
Name: amount_tsh, dtype: float64
count    59400.000000
mean      1300.652475
std        951.620547
min          0.000000
25%          0.000000
50%       1986.000000
75%       2004.000000
max       2013.000000
Name: construction_year, dtype: float64
```

In [75]:
```
# based on some intuition and the quant
tsh_bins = [-1,0,20,350000]
construction_year_bins = [0,1,1985,2005,
```

In [76]:
```
# now lets convert these columns to bin
# note that when binning these, the def

df_train['amount_tsh'] = pd.cut(df_trai
df_train['construction_year'] = pd.cut(
```

Contents

Contents

In [77]:
```
# and we will also take a look at the di
for feat in bin_features:
    print(df_train[feat].value_counts(nc
```

```
(-1, 0]          0.700993
(20, 350000]     0.249495
(0, 20]          0.049512
Name: amount_tsh, dtype: float64
NaN                   0.348636
(1985.0, 2005.0]      0.275707
(2005.0, 2015.0]      0.225000
(1.0, 1985.0]         0.150657
(0.0, 1.0]            0.000000
Name: construction_year, dtype: float64
```

In [78]:
```
df_train['amount_tsh'].value_counts(norm
```

Out[78]:
```
(-1, 0]          0.700993
(20, 350000]     0.249495
(0, 20]          0.049512
Name: amount_tsh, dtype: float64
```

In [79]:
```
df_train['amount_tsh']
```

Out[79]:
```
0         (20, 350000]
1              (-1, 0]
2         (20, 350000]
3              (-1, 0]
4              (-1, 0]
                ...
59395          (0, 20]
59396     (20, 350000]
59397          (-1, 0]
59398          (-1, 0]
59399          (-1, 0]
Name: amount_tsh, Length: 59400, dtype:
Categories (3, interval[int64]): [(-1, (
```

In [80]:
```
df_train['amount_tsh'].value_counts(norm
```

Out[80]:
```
(-1, 0]          0.700993
(20, 350000]     0.249495
(0, 20]          0.049512
Name: amount_tsh, dtype: float64
```

In [81]:
```
df_train.columns
```

Out[81]:
```
Index(['id_left', 'amount_tsh', 'longitu
       'region_code', 'district_code', '
       'extraction_type_group', 'managem
       'quality_group', 'quantity', 'sou
       'waterpoint_type', 'installer_nev
       'status_group'],
      dtype='object')
```

Contents

4.4.3 Clean Target

In [82]:
```
df_train['status_group'].value_counts(nc
```

Out[82]:
```
functional                 0.436229
non functional             0.305522
NaN                        0.200471
functional needs repair    0.057778
Name: status_group, dtype: float64
```

In [83]:
```
df_train.status_group.fillna(0, inplace=
```

In [84]:
```
df_train['status_group'].value_counts(nc
```

Out[84]:
```
functional                 0.436229
non functional             0.305522
0                          0.200471
functional needs repair    0.057778
Name: status_group, dtype: float64
```

In [85]:
```
# delte rows where prediction values dic
df_train = df_train[df_train.status_grou
```

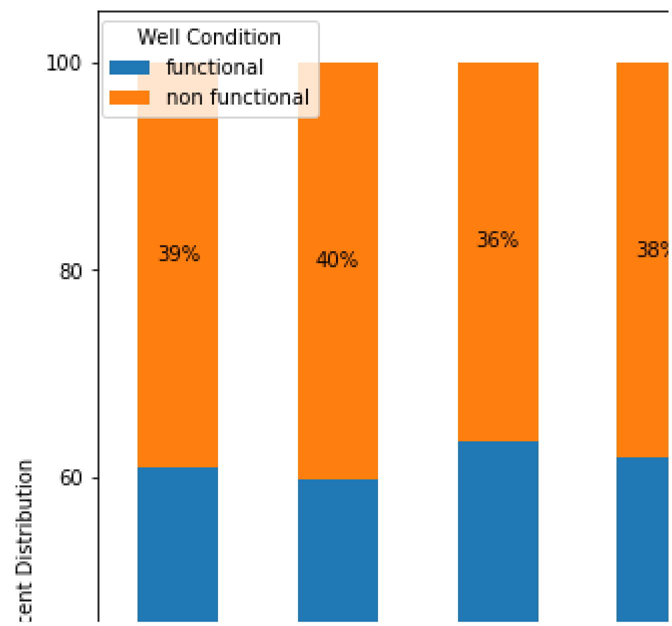
4.4.4 Visualizations

In [184]:
```
df_train['status_group_2'] = df_train['s
di = {'functional': 'functional', 'non f
df_train['status_group_2'].replace(di, i
```

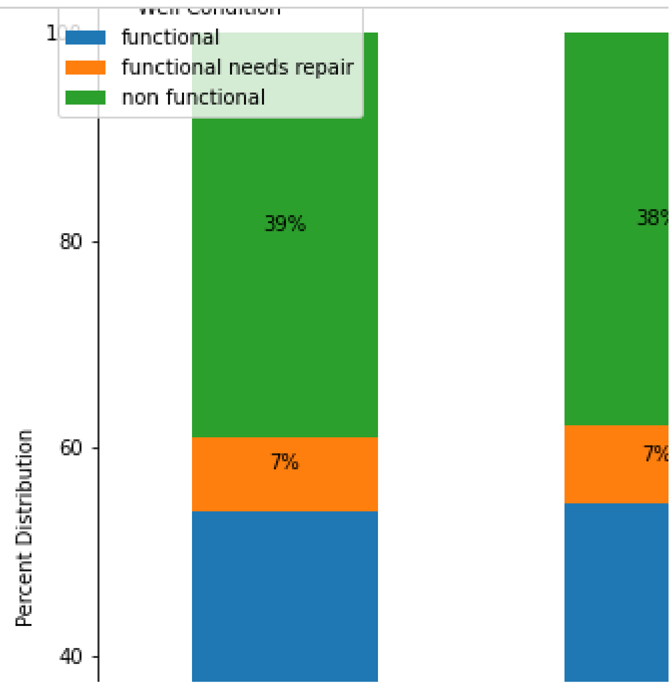
In [191]:
```
# creating a list
visuals = df_train.columns

# items to be removed
unwanted_ele = ['id_left','longitude','l

visuals = [ele for ele in visuals if ele
```

Contents

In [198]:
```
for column in visuals:
    ax= pd.crosstab(df_train[column], df
    ax_1 = ax.plot.bar(figsize=(10,10),s
    display(ax)
    plt.legend(loc='upper center', bbox_
    plt.xlabel(column)
    plt.xticks(rotation=65)
    plt.ylabel('Percent Distribution')

    for rec in ax_1.patches:
        height = rec.get_height()
        ax_1.text(rec.get_x() + rec.get_
              rec.get_y() + height / 2,
              "{:.0f}%".format(height),
              ha='center',
              va='bottom')
    plt.show()
```



In [199]:
```
for column in visuals:
    ax= pd.crosstab(df_train[column], df
    ax_1 = ax.plot.bar(figsize=(10,10),s
    display(ax)
    plt.legend(loc='upper center', bbox_
    plt.xlabel(column)
    plt.xticks(rotation=65)
    plt.ylabel('Percent Distribution')

    for rec in ax_1.patches:
        height = rec.get_height()
        ax_1.text(rec.get_x() + rec.get_
                rec.get_y() + height / 2,
                "{:.0f}%".format(height),
                ha='center',
                va='bottom')
plt.show()
```



In [203]:
```
df_train['latitude'].value_counts(normal
```

Out[203]:
```
-2.000000e-08    0.030574
-6.980122e+00    0.000042
-2.494546e+00    0.000042
-6.956746e+00    0.000042
-6.964258e+00    0.000042
                   ...
-6.899767e+00    0.000021
-4.422470e+00    0.000021
-8.446126e+00    0.000021
-3.220441e+00    0.000021
-7.714786e+00    0.000021
Name: latitude, Length: 45996, dtype: fl
```

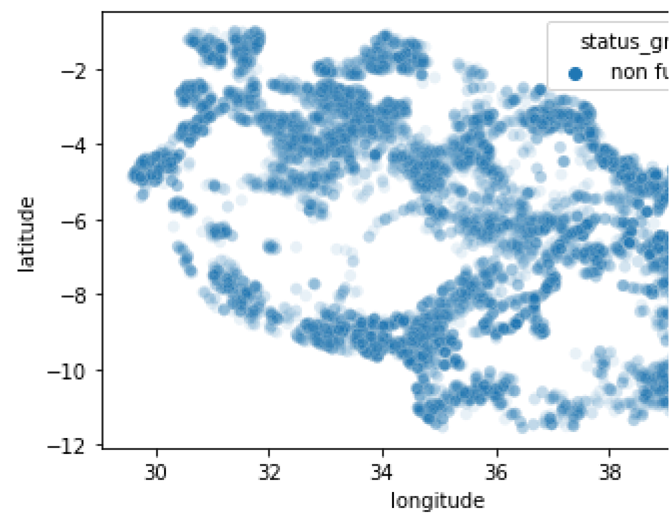
Contents

In [204]:
```
df_train['longitude'].value_counts(norma
```

Out[204]:
```
0.000000     0.030574
37.314250    0.000042
32.984790    0.000042
39.105307    0.000042
39.095087    0.000042
               ...
29.810473    0.000021
37.383938    0.000021
38.991929    0.000021
38.455444    0.000021
36.781854    0.000021
Name: longitude, Length: 45994, dtype: i
```

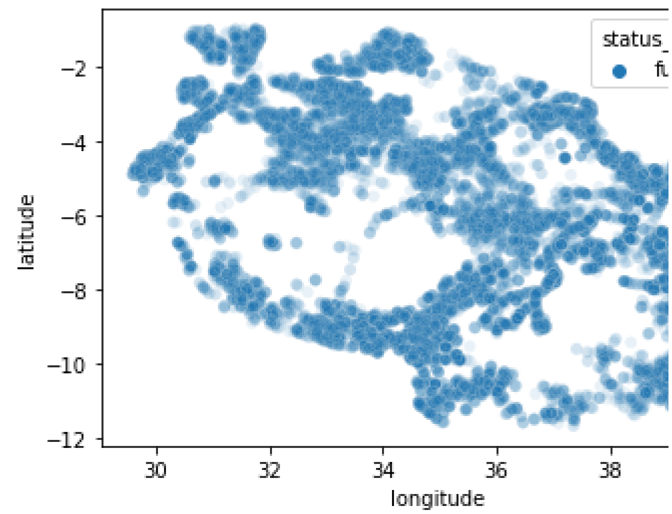
In [205]:
```
df_visual = df_train[df_train.longitude
```

In [223]:
```
ıs.scatterplot('longitude','latitude', da

.t.show()
```



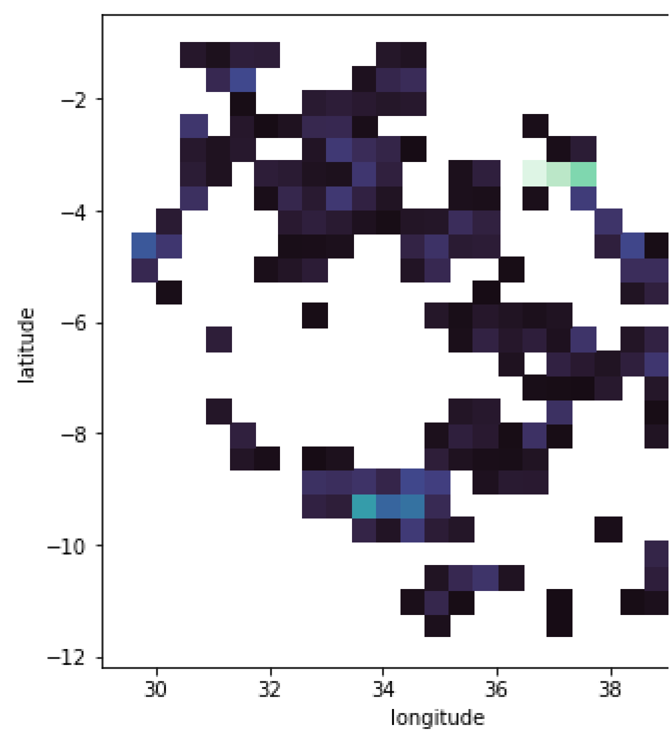
Contents

In [224]:
```
:atterplot('longitude','latitude', data=d
```
```
ıow()
```



In [218]:
```
# Draw a combo histogram and scatterplot
f, ax = plt.subplots(figsize=(6, 6))
#sns.scatterplot(x='longitude', y='latit
sns.histplot(x='longitude', y='latitude'
             bins=25, pthresh=.1, cmap='
             data=df_visual[df_visual['s
#sns.kdeplot(x='longitude', y='latitude
```

Out[218]: <AxesSubplot:xlabel='longitude', ylabel=

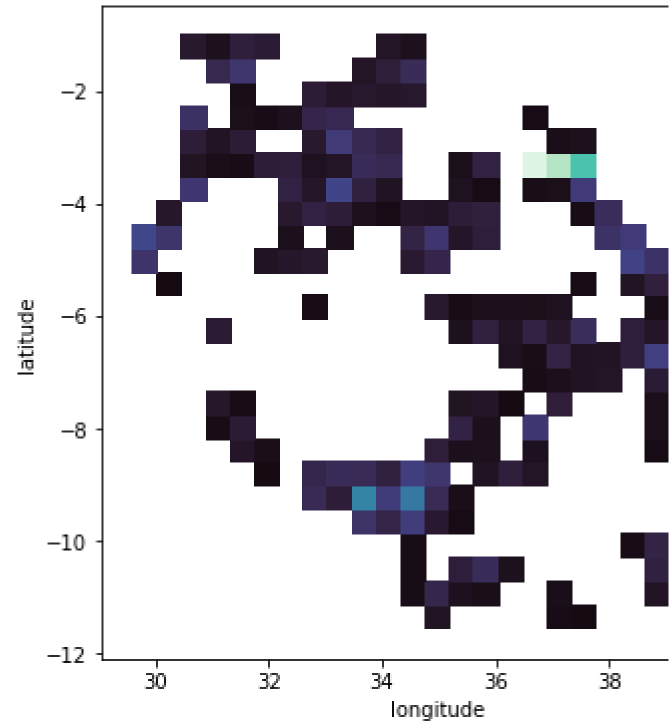


Contents

Contents

In [219]:
```
# Draw a combo histogram and scatterplot
f, ax = plt.subplots(figsize=(6, 6))
#sns.scatterplot(x='longitude', y='latit
sns.histplot(x='longitude', y='latitude'
             bins=25, pthresh=.1, cmap='
             data=df_visual[df_visual['s
#sns.kdeplot(x='longitude', y='latitude
```

Out[219]:
```
<AxesSubplot:xlabel='longitude', ylabel=
```



5 Train Test Split

In [86]:
```
X=df_train.drop(columns = ['status_group
y=df_train['status_group'] #Target
```

In [87]:
```
X_train, X_test, y_train, y_test = trair
```

In [88]:
```
X_train
```

le	lga	permit	construction_year	...	management_group	payme
53	other	1	(1985.0, 2005.0]	...	user-group	neve pa
5	hai	1	(1985.0, 2005.0]	...	user-group	oth
2	other	0	(1985.0, 2005.0]	...	user-group	neve pa
3	same	1	(1985.0, 2005.0]	...	user-group	pa annual
3	other	0	(1.0, 1985.0]	...	user-group	neve pa
...	...	...	...	...	...	
7	mbarali	1	NaN	...	user-group	neve pa

In [89]:
```
X_train.shape
```
Out[89]:
```
(33244, 22)
```

In [90]:
```
X_test.shape
```
Out[90]:
```
(14248, 22)
```

6 Encode Features

In [91]:
```
df_train.columns
```
Out[91]:
```
Index(['id_left', 'amount_tsh', 'longitu
       'region_code', 'district_code', '
       'extraction_type_group', 'managem
       'quality_group', 'quantity', 'sou
       'waterpoint_type', 'installer_new
       'status_group'],
      dtype='object')
```

In [92]:
```
col_names = list(df_train.columns)
```

In [93]:
```
col_names_ohe = col_names
```

In [94]:
```
remove_ohe = ['id_left','id_right','perm
```

Contents

Contents

In [95]:
```
for ele in remove_ohe:
    if ele in col_names_ohe:
        col_names_ohe.remove(ele)
col_names_ohe
```

Out[95]:
```
['amount_tsh',
 'basin',
 'region_code',
 'district_code',
 'lga',
 'construction_year',
 'extraction_type_group',
 'management',
 'management_group',
 'payment',
 'quality_group',
 'quantity',
 'source_type',
 'source_class',
 'waterpoint_type',
 'installer_new',
 'funder_new']
```

In [96]:
```
X_train[col_names] = X_train[col_names_c
X_test[col_names] = X_test[col_names_ohe
```

In [97]:
```
X_train_ohe = X_train[col_names_ohe]
X_test_ohe = X_test[col_names_ohe]
```

6.1 X_train Encode

In [98]:
```
X_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 33244 entries, 13863 to 309
Data columns (total 22 columns):
 #   Column                 Non-Null Cou
---  ------                 ------------
 0   id_left                33244 non-nu
 1   amount_tsh             33244 non-nu
 2   longitude              33244 non-nu
 3   latitude               33244 non-nu
 4   basin                  33244 non-nu
 5   region_code            33244 non-nu
 6   district_code          33244 non-nu
 7   lga                    33244 non-nu
 8   permit                 33244 non-nu
 9   construction_year      21674 non-nu
 10  extraction_type_group  33244 non-nu
 11  management             33244 non-nu
 12  management_group       33244 non-nu
 13  payment                33244 non-nu
 14  quality_group          33244 non-nu
 15  quantity               33244 non-nu
 16  source_type            33244 non-nu
 17  source_class           33244 non-nu
 18  waterpoint_type        33244 non-nu
 19  installer_new          33244 non-nu
 20  funder_new             33244 non-nu
 21  id_right               33244 non-nu
dtypes: category(17), float64(3), int64
memory usage: 2.1 MB
```

Change Sci kit learn OHE set sparse to false set hand

In [99]:
```
enc = OneHotEncoder(handle_unknown='igno
```

In [100]:
```
enc.fit(X_train_ohe)
```

Out[100]:
```
OneHotEncoder(handle_unknown='ignore', s
```

In [101]:
```
enc.categories_
```

Out[101]:
```
[array([Interval(-1, 0, closed='right'),
        Interval(20, 350000, closed='rig
 array(['Internal', 'Lake Nyasa', 'Lake
        'Lake Victoria', 'Pangani', 'Rui
        'Wami / Ruvu'], dtype=object),
 array([ 1,  2,  3,  4,  5,  6,  7,  8,
        18, 19, 20, 21, 24, 40, 60, 80,
 array([ 0,  1,  2,  3,  4,  5,  6,  7,
        63, 67, 80]),
 array(['arusha rural', 'bagamoyo', 'bar
        'kahama', 'karagwe', 'kasulu', '
        'kilombero', 'kilosa', 'kwimba',
        'makete', 'maswa', 'mbarali', 'n
        'moshi rural', 'mpanda', 'mvomer
        'other', 'rombo', 'rungwe', 'san
        'songea rural', 'ulanga'], dtype
 array([Interval(1.0, 1985.0, closed='ri
        Interval(1985.0, 2005.0, closed=
        Interval(2005.0, 2015.0, closed=
```

In [102]:
```
X_train2 = enc.transform(X_train_ohe)
```

In [103]:
```
X_train2.shape
```

Out[103]:
```
(33244, 197)
```

In [104]:
```
enc.get_feature_names()
column_name = enc.get_feature_names()
one_hot_encoded_frame =  pd.DataFrame(X_
```

In [105]:
```
X_train = one_hot_encoded_frame.reset_in
```

Contents

Contents

In [106]:
```
X_train.drop(col_names_ohe, axis=1, inpl
X_train
```

Out[106]:

	x0_(-1, 0]	x0_(0, 20]	x0_(20, 350000]	x1_Internal	x1_Lake Nyasa
0	1.0	0.0	0.0	0.0	0.0
1	0.0	0.0	1.0	0.0	0.0
2	1.0	0.0	0.0	0.0	0.0
3	0.0	0.0	1.0	0.0	0.0
4	1.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...
33239	1.0	0.0	0.0	0.0	0.0
33240	0.0	1.0	0.0	0.0	0.0
33241	1.0	0.0	0.0	0.0	0.0
33242	0.0	0.0	1.0	0.0	0.0

In [107]:
```
del X_train['id_left']
del X_train['id_right']
```

In [108]:
```
X_train
```

Out[108]:

	x0_(-1, 0]	x0_(0, 20]	x0_(20, 350000]	x1_Internal	x1_Lake Nyasa
0	1.0	0.0	0.0	0.0	0.0
1	0.0	0.0	1.0	0.0	0.0
2	1.0	0.0	0.0	0.0	0.0
3	0.0	0.0	1.0	0.0	0.0
4	1.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...
33239	1.0	0.0	0.0	0.0	0.0
33240	0.0	1.0	0.0	0.0	0.0
33241	1.0	0.0	0.0	0.0	0.0
33242	0.0	0.0	1.0	0.0	0.0

6.2 X_test Encode

Contents

In [109]:
```
X_test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 14248 entries, 50633 to 3017
Data columns (total 22 columns):
 #   Column                 Non-Null Cou
---  ------                 ------------
 0   id_left                14248 non-nu
 1   amount_tsh             14248 non-nu
 2   longitude              14248 non-nu
 3   latitude               14248 non-nu
 4   basin                  14248 non-nu
 5   region_code            14248 non-nu
 6   district_code          14248 non-nu
 7   lga                    14248 non-nu
 8   permit                 14248 non-nu
 9   construction_year      9298 non-nul
 10  extraction_type_group  14248 non-nu
 11  management             14248 non-nu
 12  management_group       14248 non-nu
 13  payment                14248 non-nu
```

In [110]:
```
enc.categories_
```

Out[110]:
```
[array([Interval(-1, 0, closed='right'),
        Interval(20, 350000, closed='ri
 array(['Internal', 'Lake Nyasa', 'Lake
        'Lake Victoria', 'Pangani', 'Ru
        'Wami / Ruvu'], dtype=object),
 array([ 1,  2,  3,  4,  5,  6,  7,  8,
        18, 19, 20, 21, 24, 40, 60, 80,
 array([ 0,  1,  2,  3,  4,  5,  6,  7,
        63, 67, 80]),
 array(['arusha rural', 'bagamoyo', 'ba
        'kahama', 'karagwe', 'kasulu', '
        'kilombero', 'kilosa', 'kwimba',
        'makete', 'maswa', 'mbarali', 'r
        'moshi rural', 'mpanda', 'mvome
        'other', 'rombo', 'rungwe', 'san
        'songea rural', 'ulanga'], dtype
 array([Interval(1.0, 1985.0, closed='ri
        Interval(1985.0, 2005.0, closed=
        Interval(2005.0, 2015.0, closed=
```

In [111]:
```
X_test2 = enc.transform(X_test_ohe)
```

In [112]:
```
X_test2.shape
```

Out[112]:
```
(14248, 197)
```

In [113]:
```
enc.get_feature_names()
column_name = enc.get_feature_names()
one_hot_encoded_frame =  pd.DataFrame(X_
```

Contents

In [114]:
```
X_test = one_hot_encoded_frame.reset_inc
```

In [115]:
```
X_test.drop(col_names_ohe, axis=1, inpla
X_test
```

Out[115]:

	x0_(-1, 0]	x0_(0, 20]	x0_(20, 350000]	x1_Internal	x1_Lake Nyasa
0	1.0	0.0	0.0	0.0	0.0
1	1.0	0.0	0.0	0.0	1.0
2	1.0	0.0	0.0	0.0	0.0
3	0.0	0.0	1.0	0.0	0.0
4	0.0	0.0	1.0	0.0	0.0
...	...	...	...	...	...
14243	1.0	0.0	0.0	0.0	0.0
14244	1.0	0.0	0.0	1.0	0.0
14245	1.0	0.0	0.0	1.0	0.0
14246	0.0	0.0	1.0	0.0	0.0

In [116]:
```
del X_test['id_left']
del X_test['id_right']
```

In [117]:
```
X_test
```

Out[117]:

	x0_(-1, 0]	x0_(0, 20]	x0_(20, 350000]	x1_Internal	x1_Lake Nyasa
0	1.0	0.0	0.0	0.0	0.0
1	1.0	0.0	0.0	0.0	1.0
2	1.0	0.0	0.0	0.0	0.0
3	0.0	0.0	1.0	0.0	0.0
4	0.0	0.0	1.0	0.0	0.0
...	...	...	...	...	...
14243	1.0	0.0	0.0	0.0	0.0
14244	1.0	0.0	0.0	1.0	0.0
14245	1.0	0.0	0.0	1.0	0.0
14246	0.0	0.0	1.0	0.0	0.0

In [118]:
```
X_train.shape
```

Out[118]: (33244, 200)

In [119]:
```
X_test.shape
```
Out[119]:
```
(14248, 200)
```

Contents

6.3 Delete 'other' columns OHE

For Text data which was too low incidence to make it

6.3.1 Delete Other Train

In [120]:
```
X_train
```
Out[120]:

	x0_(-1, 0]	x0_(0, 20]	x0_(20, 350000]	x1_Internal	x1_Lake Nyasa
0	1.0	0.0	0.0	0.0	0.0
1	0.0	0.0	1.0	0.0	0.0
2	1.0	0.0	0.0	0.0	0.0
3	0.0	0.0	1.0	0.0	0.0
4	1.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...
33239	1.0	0.0	0.0	0.0	0.0
33240	0.0	1.0	0.0	0.0	0.0
33241	1.0	0.0	0.0	0.0	0.0
33242	0.0	0.0	1.0	0.0	0.0
33243	1.0	0.0	0.0	1.0	0.0

33244 rows × 200 columns

In [121]:
```
other_cols = [col for col in X_train.co]
other_cols
```
Out[121]:
```
['x4_other',
 'x6_other',
 'x6_other handpump',
 'x6_other motorpump',
 'x7_other',
 'x7_other - school',
 'x8_other',
 'x9_other',
 'x12_other',
 'x14_other',
 'x15_other',
 'x16_other']
```

In [122]:
```
X_train = X_train.drop(other_cols, axis=
X_train.columns
```

Out[122]:
```
Index(['x0_(-1, 0]', 'x0_(0, 20]', 'x0_(
       'x1_Lake Nyasa', 'x1_Lake Rukwa',
       'x1_Lake Victoria', 'x1_Pangani',
       ...
       'x16_private individual', 'x16_rw
       'x16_unicef', 'x16_world bank', '
       'latitude', 'permit'],
      dtype='object', length=188)
```

6.3.2 Delete Other Test

In [123]:
```
X_test = X_test.drop(other_cols, axis=1)
X_test.columns
```

Out[123]:
```
Index(['x0_(-1, 0]', 'x0_(0, 20]', 'x0_(
       'x1_Lake Nyasa', 'x1_Lake Rukwa',
       'x1_Lake Victoria', 'x1_Pangani',
       ...
       'x16_private individual', 'x16_rw
       'x16_unicef', 'x16_world bank', '
       'latitude', 'permit'],
      dtype='object', length=188)
```

In [124]:
```
other_cols_test = [col for col in X_test
other_cols_test
```

Out[124]:
```
[]
```

6.3.3 Label Encode Target

In [125]:
```
di = {0: 0, 'functional': 1, 'non functi
y_train.replace(di, inplace=True)
```

In [126]:
```
di = {0: 0, 'functional': 1, 'non functi
y_test.replace(di, inplace=True)
```

7 Model Development

7.1 Random Forest Classifier

Contents

In [127]:
```
# Instantiate and fit the RandomForestCl
rfc=RandomForestClassifier(n_estimators=
rfc.fit(X_train,y_train)
```

Out[127]:
```
RandomForestClassifier(class_weight='bal
```

In [128]:
```
training_start = time.perf_counter()
rfc.fit(X_train, y_train)
training_end = time.perf_counter()
prediction_start = time.perf_counter()
y_pred = rfc.predict(X_test)
y_pred_train = rfc.predict(X_train)
prediction_end = time.perf_counter()
acc_rfc = (y_pred == y_test).sum().astyp
rfc_train_time = training_end-training_s
rfc_prediction_time = prediction_end-pre
print("Scikit-Learn's Random Forest Clas
print("Time consumed for training: %4.3f
print("Time consumed for prediction: %6.
```

```
Scikit-Learn's Random Forest Classifier'
Time consumed for training: 1.057 second
Time consumed for prediction: 0.24293 se
```

In [129]:
```
print(metrics.classification_report(y_te
```

```
              precision    recall  f1-sc

           1       0.55      0.65       (
           2       0.39      0.32       (
           3       0.10      0.06       (

    accuracy                            (
   macro avg       0.35      0.34       (
weighted avg       0.46      0.48       (
```

7.1.1 Check for Overfit

In [130]:
```
# View confusion matrix for train data a
confusion_matrix(y_train, y_pred_train)
```

Out[130]:
```
array([[17717,   294,   113],
       [  865, 11780,    70],
       [  207,   116,  2082]])
```

In [131]:
```
# View confusion matrix for test data ar
confusion_matrix(y_test, y_pred)
```

Out[131]:
```
array([[5026, 2429,  333],
       [3449, 1760,  224],
       [ 638,  329,   60]])
```

Contents

Contents

In [132]:
```
rfc_training_preds = rfc.predict(X_trair
rfc_training_accuracy = accuracy_score(y

rfc_val_preds = rfc.predict(X_test) # y_
rfc_val_accuracy = accuracy_score(y_test
print(rfc_training_accuracy)
print(rfc_val_accuracy)
```

```
0.9499157742750571
0.48048848961257723
```

Because we are looking for the minority class accura

Lets take a look at other metrics in the classification r

In [133]:
```
print(metrics.classification_report(y_tr
```

```
              precision    recall  f1-sc

           1       0.94      0.98       (
           2       0.97      0.93       (
           3       0.92      0.87       (

    accuracy                            (
   macro avg       0.94      0.92       (
weighted avg       0.95      0.95       (
```

In [134]:
```
print(metrics.classification_report(y_te
```

```
              precision    recall  f1-sc

           1       0.55      0.65       (
           2       0.39      0.32       (
           3       0.10      0.06       (

    accuracy                            (
   macro avg       0.35      0.34       (
weighted avg       0.46      0.48       (
```

Add labels so it's clear which class is being describe

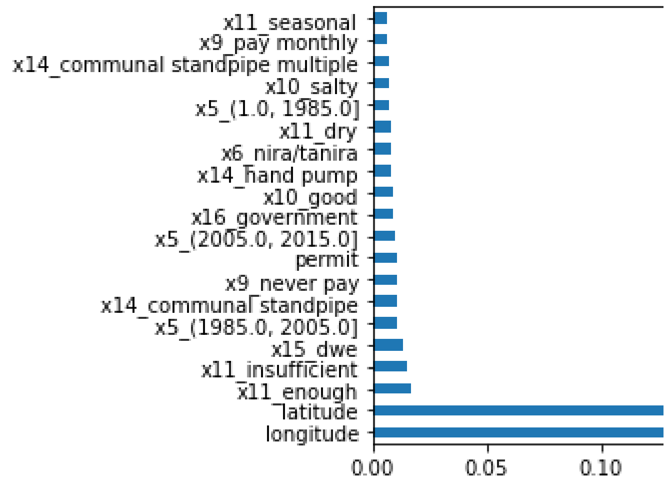
7.1.2 Feature Importance

In [135]:
```
# create list so that random forest mode
rfc_columns = list(X_train.columns)
```

Contents

In [136]:
```
feat_importances = pd.Series(rfc.feature
feat_importances.nlargest(20).plot(kind=
```

Out[136]: <AxesSubplot:>



7.1.3 Model reiteration - parameter tur

Don't set anything to random, choose the ones that y
percentage of features.

In [137]:
```
model_params = {
    # number of trees
    'n_estimators': [5,10,20,50,100,200]
    # number of max features
    'max_features': [10,15,20,50,100],
    # max number of levels in each decis
    'max_depth': [5,10,15],
    'min_samples_split' : [100,1000]
}
```

In [138]:
```
# Instantiate and fit the RandomForestCl
rfc=RandomForestClassifier(class_weight=
rfc = RandomizedSearchCV(rfc,model_paran
rfc.fit(X_train,y_train)
```

Out[138]:
```
RandomizedSearchCV(cv=3,
                   estimator=RandomFores
                   param_distributions={



                   random_state=1)
```

In [139]:
```
print(rfc.best_estimator_.get_params())
```

```
{'bootstrap': True, 'ccp_alpha': 0.0, 'c
s': 50, 'max_leaf_nodes': None, 'max_sam
mples_leaf': 1, 'min_samples_split': 10(
core': False, 'random_state': None, 'ver
```

In [140]:
```
print(metrics.classification_report(y_tr
```

```
              precision    recall  f1-sc

           1       0.94      0.98      (
           2       0.97      0.93      (
           3       0.92      0.87      (

    accuracy                           (
   macro avg       0.94      0.92      (
weighted avg       0.95      0.95      (
```

In [141]:
```
print(metrics.classification_report(y_te
```

```
              precision    recall  f1-sc

           1       0.55      0.65      (
           2       0.39      0.32      (
           3       0.10      0.06      (

    accuracy                           (
   macro avg       0.35      0.34      (
weighted avg       0.46      0.48      (
```

7.2 Gradient Boosting Classifie

In [142]:
```
gb_clf = GradientBoostingClassifier()
gb_clf.fit(X_train, y_train)
y_pred = gb_clf.predict(X_test)
y_pred_train = gb_clf.predict(X_train)

# print("Accuracy score (training): {0:.
# print("Accuracy score (validation): {(
```

Contents

Contents

In [143]:
```
print(metrics.classification_report(y_te
```

```
              precision    recall  f1-sc

           1       0.55      0.99       (
           2       0.34      0.01       (
           3       0.50      0.00       (

    accuracy                            (
   macro avg       0.46      0.33       (
weighted avg       0.46      0.54       (
```

This model did worse at finding the minority class

7.2.1 Check for Overfit

In [144]:
```
# View confusion matrix for train data a
confusion_matrix(y_train, y_pred_train)
```

Out[144]:
```
array([[18074,    50,     0],
       [12410,   305,     0],
       [ 2360,    32,    13]])
```

In [145]:
```
# View confusion matrix for test data an
confusion_matrix(y_test, y_pred)
```

Out[145]:
```
array([[7706,   82,    0],
       [5386,   46,    1],
       [1019,    7,    1]])
```

In [146]:
```
print(metrics.classification_report(y_tr
```

```
              precision    recall  f1-sc

           1       0.55      1.00       (
           2       0.79      0.02       (
           3       1.00      0.01       (

    accuracy                            (
   macro avg       0.78      0.34       (
weighted avg       0.67      0.55       (
```

Contents

In [147]:
```
print(metrics.classification_report(y_te
```

```
              precision    recall  f1-sc

           1       0.55      0.99      (
           2       0.34      0.01      (
           3       0.50      0.00      (

    accuracy                           (
   macro avg       0.46      0.33      (
weighted avg       0.46      0.54      (
```

7.2.2 Model reiteration - parameter tur

In [148]:
```
model_params = {
    # number of boosting stages
    'n_estimators': [5,10,20,50,100,200]
    # number of max features
    'max_features': [10,15,20,50,100],
    # Learning rate
    'learning_rate':[.25,.5,.75,1],
    #The minimum number of samples requi
    'min_samples_split' : [100,1000]
}
```

In [149]:
```
gb_clf = RandomizedSearchCV(gb_clf,model
gb_clf.fit(X_train,y_train)
```

Out[149]:
```
RandomizedSearchCV(cv=3, estimator=Gradi
                   param_distributions={



                   random_state=1)
```

In [150]:
```
print(metrics.classification_report(y_tr
```

```
              precision    recall  f1-sc

           1       0.55      1.00      (
           2       0.79      0.02      (
           3       1.00      0.01      (

    accuracy                           (
   macro avg       0.78      0.34      (
weighted avg       0.67      0.55      (
```

In [151]:
```
print(metrics.classification_report(y_te
```

```
              precision    recall  f1-sc

           1       0.55      0.99       (
           2       0.34      0.01       (
           3       0.50      0.00       (

    accuracy                            (
   macro avg       0.46      0.33       (
weighted avg       0.46      0.54       (
```

In [152]:
```
lr_list = [0.05, 0.075, 0.1, 0.25, 0.5,

for learning_rate in lr_list:
    gb_clf = GradientBoostingClassifier(
    gb_clf.fit(X_train, y_train)

    print("Learning rate: ", learning_ra
    print("Accuracy score (training): {(
    print("Accuracy score (validation):
```

```
Accuracy score (validation): 0.546
Learning rate:  0.075
Accuracy score (training): 0.546
Accuracy score (validation): 0.546
Learning rate:  0.1
Accuracy score (training): 0.547
Accuracy score (validation): 0.546
Learning rate:  0.25
Accuracy score (training): 0.550
Accuracy score (validation): 0.545
Learning rate:  0.5
Accuracy score (training): 0.558
Accuracy score (validation): 0.541
Learning rate:  0.75
Accuracy score (training): 0.558
Accuracy score (validation): 0.539
Learning rate:  1
Accuracy score (training): 0.561
Accuracy score (validation): 0.538
```

7.3 Logistic Regression

7.3.1 unbalanced

Contents

Contents

In [153]:
```
# Logistic model
log_clf = LogisticRegression(random_stat
log_model = log_clf.fit(X_train, y_trair
log_training_preds = log_clf.predict(X_t
log_training_accuracy = accuracy_score(y
log_val_preds = log_clf.predict(X_test)
log_val_accuracy = accuracy_score(y_test
```

In [154]:
```
#Confusion matrix for Logistic Regressio
log_matrix = confusion_matrix(y_test, lo
print('Confusion Matrix:\n', log_matrix)
```

```
Confusion Matrix:
 [[7677  111    0]
 [5361   72    0]
 [1009   18    0]]
```

In [155]:
```
print(log_training_accuracy)
print(log_val_accuracy)
```

```
0.5455119720851883
0.5438658057271196
```

In [156]:
```
y_pred = log_model.predict(X_test)
```

In [157]:
```
print(metrics.classification_report(y_te
```

```
              precision    recall  f1-sc

           1       0.55      0.99      (
           2       0.36      0.01      (
           3       0.00      0.00      (

    accuracy                           (
   macro avg       0.30      0.33      (
weighted avg       0.44      0.54      (
```

7.3.2 balanced

In [158]:
```
# Logistic model
log_clf = LogisticRegression(multi_class
log_model = log_clf.fit(X_train, y_trair
y_pred_train = log_model.predict(X_trair
y_pred = log_model.predict(X_test)
log_training_accuracy = accuracy_score(y
log_val_accuracy = accuracy_score(y_test
```

Contents

In [159]:
```
#Confusion matrix for Logistic Regressic
log_matrix = confusion_matrix(y_test, y_
print('Confusion Matrix:\n', log_matrix)
```

```
Confusion Matrix:
 [[2546 2399 2843]
 [1735 1657 2041]
 [ 347  297  383]]
```

In [160]:
```
print(log_training_accuracy)
print(log_val_accuracy)
```

```
0.349446516664613
0.32186973610331276
```

In [161]:
```
print(metrics.classification_report(y_te
```

```
              precision    recall  f1-sc

           1       0.55      0.33       (
           2       0.38      0.30       (
           3       0.07      0.37       (

    accuracy                            (
   macro avg       0.33      0.33       (
weighted avg       0.45      0.32       (
```

7.3.3 Check for Overfit

In [162]:
```
# View confusion matrix for train data a
confusion_matrix(y_train, y_pred_train)
```

Out[162]:
```
array([[6334, 5293, 6497],
       [3973, 4206, 4536],
       [ 686,  642, 1077]])
```

In [163]:
```
# View confusion matrix for test data ai
confusion_matrix(y_test, y_pred)
```

Out[163]:
```
array([[2546, 2399, 2843],
       [1735, 1657, 2041],
       [ 347,  297,  383]])
```

Contents

In [164]:
```
print(metrics.classification_report(y_tr
```

```
              precision    recall  f1-sc

           1       0.58      0.35       (
           2       0.41      0.33       (
           3       0.09      0.45       (

    accuracy                            (
   macro avg       0.36      0.38       (
weighted avg       0.48      0.35       (
```

In [165]:
```
print(metrics.classification_report(y_te
```

```
              precision    recall  f1-sc

           1       0.55      0.33       (
           2       0.38      0.30       (
           3       0.07      0.37       (

    accuracy                            (
   macro avg       0.33      0.33       (
weighted avg       0.45      0.32       (
```

7.3.4 Model reiteration - parameter tur

In [166]:
```
# Logistic model
log_clf = LogisticRegression(multi_class
log_model = log_clf.fit(X_train, y_trair
y_pred_train = log_model.predict(X_trair
y_pred = log_model.predict(X_test)
log_training_accuracy = accuracy_score(y
log_val_accuracy = accuracy_score(y_test
```

In [167]:
```
model_params = {
    #Algorithm to use in the optimizatic
    'solver': ['newton-cg', 'sag', 'saga
}
```

In [168]:
```
# Instantiate and fit the LogisticReg Mc
log_clf = LogisticRegression(multi_class
log_clf = RandomizedSearchCV(log_clf,moc
log_clf.fit(X_train,y_train)
```

Out[168]:
```
RandomizedSearchCV(cv=3,
                   estimator=LogisticReg

                   param_distributions={
```

In [169]:
```
print(log_clf.best_estimator_.get_params
```

```
{'C': 1.0, 'class_weight': 'balanced', '
'max_iter': 100, 'multi_class': 'multin
'tol': 0.0001, 'verbose': 0, 'warm_start
```

In [170]:
```
print(metrics.classification_report(y_tr
```

```
              precision    recall  f1-sc

           1       0.57      0.30      (
           2       0.41      0.35      (
           3       0.09      0.46      (

    accuracy                           (
   macro avg       0.36      0.37      (
weighted avg       0.47      0.33      (
```

In [171]:
```
print(metrics.classification_report(y_te
```

```
              precision    recall  f1-sc

           1       0.54      0.28      (
           2       0.38      0.33      (
           3       0.07      0.38      (

    accuracy                           (
   macro avg       0.33      0.33      (
weighted avg       0.44      0.30      (
```

8 Conclusions

Most Important Factors in water pump functionality:

- geographic location
- Water Quantity
- Construction Year
- Whether it was permited
- Who funded the pump
- Who installed the pump

- Installing pumps in low water quantity areas lead
- Permited pumps were more likely to be functiona
- Pumps older than 1985 have poor functionality

9 Business Recommendat

Contents

- Expand permiting system. Waterpoints which we
- Areas which already have low water quantities ne
- Add a data feature for best guess of when pump
  - This can be used to better predict lifetime of
- Review all pumps older than 1985 as these are m

Contents

10 Future Work

Things to work on if more time:

- Exploration of other numerical features such as g
  - when we examined a boxplot of these featur
  - However we may have removed these prema
- Look more closely at pump functionality by geog
  - The latitude and longitude of the waterpoint
  - Since this is by far our best feature further in