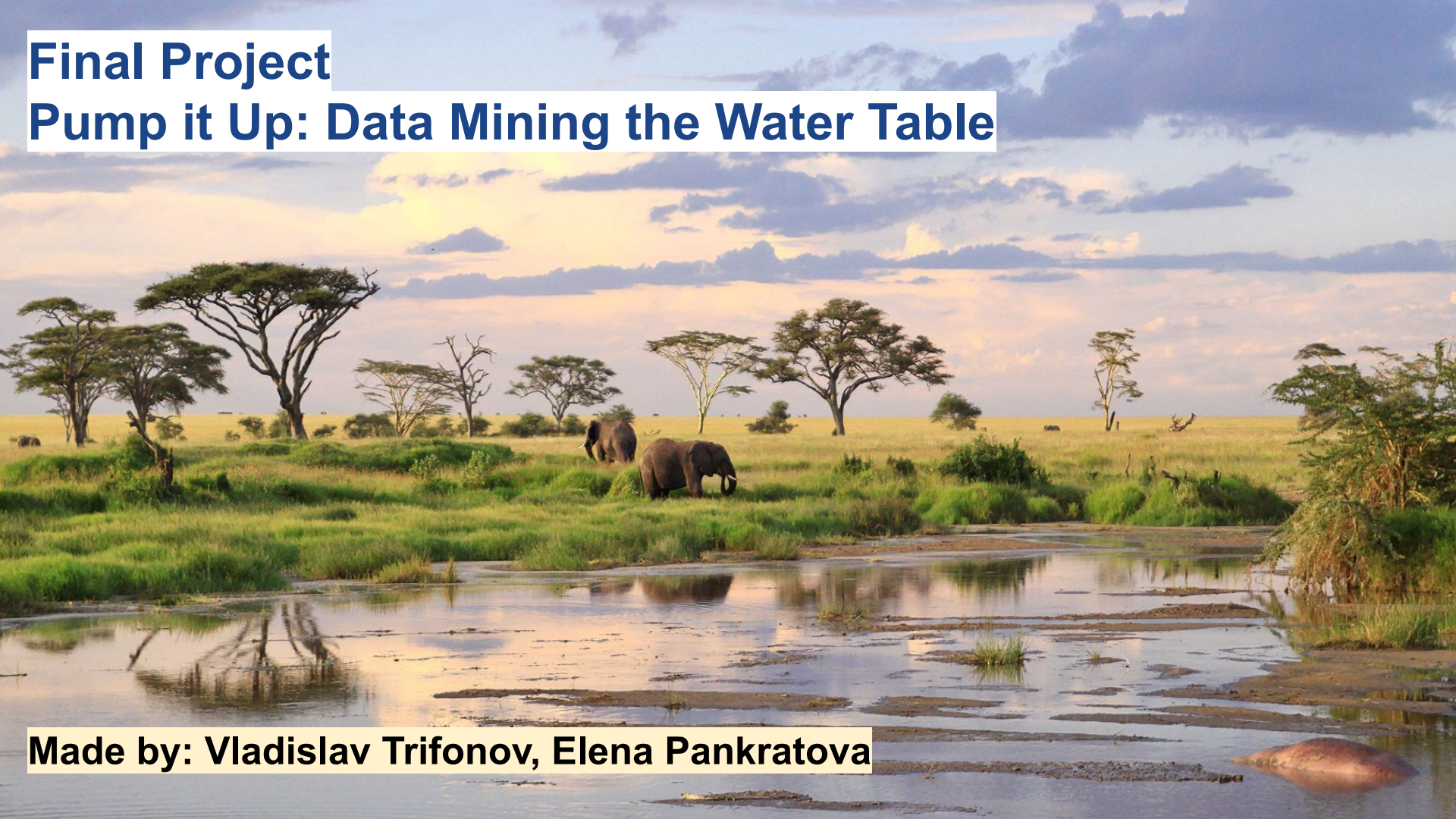
Final Project
Pump it Up: Data Mining the Water Table



Made by: Vladislav Trifonov, Elena Pankratova

Problem: people in Tanzania have lack clean water

Aim: build a model which predicts functionality of pumps

Initial data:

40 features, 59400 pumps

General infromation about pump

- `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

Can assume that this is locations of a basins

- `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. It's coded region
- `district_code` - Geographic location. It's coded district

Labels

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

Scheme and scheme_managment look as identical but is has to be investigated

- `lga` - Geographic location of the district of the village with people
- `ward` - Geographic location of village which people use well
- `population` - Population around the well
- `public_meeting` - True/False (for water?)
- `recorded_by` - Consulting corporation that provieds the data
- `scheme_management` - Who operates the waterpoint
- `scheme_name` - Who operates the waterpoint. Complicated names of specific types of the operating.
- `permit` - If the waterpoint is permitted
- `construction_year` - Year the waterpoint was constructed
- `extraction_type` - The kind of extraction the waterpoint uses. Fisrt scale
- `extraction_type_group` - The kind of extraction the waterpoint uses. Second scale
- `extraction_type_class` - The kind of extraction the waterpoint uses. Third scale
- `management` - How the waterpoint is managed
- `management_group` - How the waterpoint is managed
- `payment` - What the water costs
- `payment_type` - What the water costs. Duplicates 'payment'

As well we have almost the same values in both this features (only 356 are differ) we will drop 'quality_group'.

- `water_quality` - The quality of the water
- `quality_group` - The quality of the water

Source and source_type almost match. Source has some not important values. May be can combine them into one 'other'

- `quantity` - The quantity of water
- `quantity_group` - The quantity of water. Duplicates 'quantity'
- `source` - The source of the water
- `source_type` - The source of the water
- `source_class` - The source of the water

Water_type_group almost matches water_type. Will drop it.

- `waterpoint_type` - The kind of waterpoint
- `waterpoint_type_group` - The kind of waterpoint

Load the data

Ввод [2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

sns.set()
%matplotlib inline
```

Ввод [3]:
```
X = pd.read_csv('train_features.csv')
y = pd.read_csv('train_labels.csv')
df = pd.concat([X, y.drop('id', axis=1)], axis=1)
```

Ввод [4]:
```
pd.set_option('display.max_columns', 80)
pd.set_option('display.max_rows', 80)
pd.set_option('display.max_seq_items', 80)
```

Ввод [8]:
```
df.head(3)
```

Out[8]:

	id	amount_tsh	date_recorded	funder	gps_height	installer	longitude	latitude	wpt_name	num_private	basin	subvillage	region	region_code
0	69572	6000.0	2011-03-14	Roman	1390	Roman	34.938093	-9.856322	none	0	Lake Nyasa	Mnyusi B	Iringa	11
1	8776	0.0	2013-03-06	Grumeti	1399	GRUMETI	34.698766	-2.147466	Zahanati	0	Lake Victoria	Nyamara	Mara	20
2	34310	25.0	2013-02-25	Lottery Club	686	World vision	37.460664	-3.821329	Kwa Mahundi	0	Pangani	Majengo	Manyara	21

Preprocessing. Data cleaning

Columns with the same values:

- management_group
  management
- water_quality
  waterpoint_type_group
  quality_group
- payment
  payment_type
- quantity_group
  quantity
- source
  source_class
  source_type
- waterpoint_type_group
  waterpoint_type
- extraction_type_group
  extraction_type_class
  extraction_type

Non-informative columns:

- recorded_by
- region
- num_private
- wpt_name

Preprocessing. Data cleaning

A lot of vairables are copies of others. Some of them extends others. Let's take a look.

Ввод [5]:
```
df.num_private.value_counts()[:5]
```

Out[5]:
```
0    58643
6       81
1       73
8       46
5       46
Name: num_private, dtype: int64
```

Ввод [5]:
```
df.groupby(['waterpoint_type_group', 'waterpoint_type']).count()
```

Out[5]:

		id	amount_tsh	date_recorded	funder	gps_height	installer	longitude	latitude	wpt_name	num_private	basin	su
waterpoint_type_group	**waterpoint_type**												
cattle trough	**cattle trough**	116	116	116	113	116	110	116	116	116	116	116	
communal standpipe	**communal standpipe**	28522	28522	28522	25778	28522	25790	28522	28522	28522	28522	28522	
	communal standpipe multiple	6103	6103	6103	6064	6103	6063	6103	6103	6103	6103	6103	
dam	**dam**	7	7	7	7	7	7	7	7	7	7	7	
hand pump	**hand pump**	17488	17488	17488	16997	17488	16977	17488	17488	17488	17488	17488	
improved spring	**improved spring**	784	784	784	743	784	748	784	784	784	784	784	
other	**other**	6380	6380	6380	6063	6380	6050	6380	6380	6380	6380	6380	

Preprocessing. Data cleaning

Ввод [6]:
```
df.groupby(['source_type', 'source']).count()
```

Out[6]:

source_type	source	id	amount_tsh	date_recorded	funder	gps_height	installer	longitude	latitude	wpt_name	num_private	basin	subvillage	regior
borehole	hand dtw	874	874	874	868	874	868	874	874	874	874	874	874	874
	machine dbh	11075	11075	11075	10252	11075	10246	11075	11075	11075	11075	11075	10849	11075
dam	dam	656	656	656	647	656	646	656	656	656	656	656	656	656
other	other	212	212	212	204	212	204	212	212	212	212	212	212	212
	unknown	66	66	66	45	66	46	66	66	66	66	66	66	66
rainwater harvesting	rainwater harvesting	2295	2295	2295	2099	2295	2096	2295	2295	2295	2295	2295	2293	2295
river/lake	lake	765	765	765	763	765	762	765	765	765	765	765	764	765
	river	9612	9612	9612	8715	9612	8721	9612	9612	9612	9612	9612	9612	9612
shallow well	shallow well	16824	16824	16824	16302	16824	16286	16824	16824	16824	16824	16824	16817	16824
spring	spring	17021	17021	17021	15870	17021	15870	17021	17021	17021	17021	17021	16886	17021

Preprocessing. Data cleaning

Ввод [7]:
```
df.groupby(['source_class', 'source_type']).count()
```

Out[7]:

source_class	source_type	id	amount_tsh	date_recorded	funder	gps_height	installer	longitude	latitude	wpt_name	num_private	basin	subvillage	reg
groundwater	borehole	11949	11949	11949	11120	11949	11114	11949	11949	11949	11949	11949	11723	11
	shallow well	16824	16824	16824	16302	16824	16286	16824	16824	16824	16824	16824	16817	16
	spring	17021	17021	17021	15870	17021	15870	17021	17021	17021	17021	17021	16886	17
surface	dam	656	656	656	647	656	646	656	656	656	656	656	656	
	rainwater harvesting	2295	2295	2295	2099	2295	2096	2295	2295	2295	2295	2295	2293	2
	river/lake	10377	10377	10377	9478	10377	9483	10377	10377	10377	10377	10377	10376	10
unknown	other	278	278	278	249	278	250	278	278	278	278	278	278	

Ввод [8]:
```
df.groupby(['quantity_group', 'quantity']).count()
```

Out[8]:

quantity_group	quantity	id	amount_tsh	date_recorded	funder	gps_height	installer	longitude	latitude	wpt_name	num_private	basin	subvillage	re
dry	dry	6246	6246	6246	5781	6246	5782	6246	6246	6246	6246	6246	6087	6
enough	enough	33186	33186	33186	31963	33186	31964	33186	33186	33186	33186	33186	33087	33
insufficient	insufficient	15129	15129	15129	13950	15129	13940	15129	15129	15129	15129	15129	15020	15
seasonal	seasonal	4050	4050	4050	3414	4050	3415	4050	4050	4050	4050	4050	4046	
unknown	unknown	789	789	789	657	789	644	789	789	789	789	789	789	

Preprocessing. Data cleaning

Ввод [9]:
```
df.groupby(['quality_group', 'water_quality']).count()
```

Out[9]:

quality_group	water_quality	id	amount_tsh	date_recorded	funder	gps_height	installer	longitude	latitude	wpt_name	num_private	ba
colored	coloured	490	490	490	391	490	391	490	490	490	490	
fluoride	fluoride	200	200	200	181	200	176	200	200	200	200	
	fluoride abandoned	17	17	17	17	17	17	17	17	17	17	
good	soft	50818	50818	50818	47945	50818	47948	50818	50818	50818	50818	50
milky	milky	804	804	804	788	804	785	804	804	804	804	
salty	salty	4856	4856	4856	4803	4856	4801	4856	4856	4856	4856	4
	salty abandoned	339	339	339	331	339	331	339	339	339	339	
unknown	unknown	1876	1876	1876	1309	1876	1296	1876	1876	1876	1876	1

Ввод [10]:
```
df.groupby(['payment_type', 'payment']).count()
```

Out[10]:

payment_type	payment	id	amount_tsh	date_recorded	funder	gps_height	installer	longitude	latitude	wpt_name	num_private	basin
annually	pay annually	3642	3642	3642	3533	3642	3529	3642	3642	3642	3642	3642
monthly	pay monthly	8300	8300	8300	8084	8300	8083	8300	8300	8300	8300	8300
never pay	never pay	25348	25348	25348	23440	25348	23443	25348	25348	25348	25348	25348
on failure	pay when scheme fails	3914	3914	3914	3869	3914	3872	3914	3914	3914	3914	3914
other	other	1054	1054	1054	1038	1054	1039	1054	1054	1054	1054	1054
per bucket	pay per bucket	8985	8985	8985	8717	8985	8711	8985	8985	8985	8985	8985
unknown	unknown	8157	8157	8157	7084	8157	7068	8157	8157	8157	8157	8157

Preprocessing. Data cleaning

```
Ввод [11]: df.groupby(['extraction_type_class', 'extraction_type_group', 'extraction_type']).count()
```

Out[11]:

extraction_type_class	extraction_type_group	extraction_type	id	amount_tsh	date_recorded	funder	gps_height
gravity	gravity	gravity	26780	26780	26780	24704	26780
handpump	afridev	afridev	1770	1770	1770	1668	1770
	india mark ii	india mark ii	2400	2400	2400	2358	2400
	india mark iii	india mark iii	98	98	98	98	98
	nira/tanira	nira/tanira	8154	8154	8154	7899	8154
	other handpump	other - mkulima/shinyanga	2	2	2	1	2
		other - play pump	85	85	85	85	85
		other - swn 81	229	229	229	219	229
		walimi	48	48	48	48	48
	swn 80	swn 80	3670	3670	3670	3596	3670
motorpump	mono	mono	2865	2865	2865	2577	2865
	other motorpump	cemo	90	90	90	90	90
		climax	32	32	32	32	32
other	other	other	6430	6430	6430	6010	6430
rope pump	rope pump	other - rope pump	451	451	451	448	451
submersible	submersible	ksb	1415	1415	1415	1411	1415
		submersible	4764	4764	4764	4409	4764
wind-powered	wind-powered	windmill	117	117	117	112	117

Preprocessing. Data cleaning

Ввод [12]:
```
df.groupby(['management_group', 'management']).count()
```

Out[12]:

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

Preprocessing. Data cleaning

Ввод [13]:
```
df.date_recorded = pd.to_datetime(X.date_recorded)
```

Ввод [14]:
```
df.drop(labels=['num_private', 'waterpoint_type_group', 'source_class', 'source', 'quantity_group', 'water_quality',
                'payment', 'extraction_type_class', 'extraction_type_group', 'region', 'management_group'],
        axis=1, inplace=True)
```

Ввод [15]:
```
df.head(3)
```

Out[15]:

	id	amount_tsh	date_recorded	funder	gps_height	installer	longitude	latitude	wpt_name	basin	subvillage	region_code	district_code	lg
0	69572	6000.0	2011-03-14	Roman	1390	Roman	34.938093	-9.856322	none	Lake Nyasa	Mnyusi B	11	5	Ludew
1	8776	0.0	2013-03-06	Grumeti	1399	GRUMETI	34.698766	-2.147466	Zahanati	Lake Victoria	Nyamara	20	2	Serenge
2	34310	25.0	2013-02-25	Lottery Club	686	World vision	37.460664	-3.821329	Kwa Mahundi	Pangani	Majengo	21	4	Simanjir

Preprocessing. Data cleaning

There are a lot geographic features, we will drop some of them. And 'recorded_by' is constatnt for a whole set. From 'date_recorded' we will create an age of a pump, then drop it.

Ввод [16]:
```
df.isnull().sum()
```

Out[16]:
```
id                     0
amount_tsh             0
date_recorded          0
funder              3635
gps_height             0
installer           3655
longitude              0
latitude               0
wpt_name               0
basin                  0
subvillage           371
region_code            0
district_code          0
lga                    0
ward                   0
population             0
public_meeting      3334
recorded_by            0
scheme_management   3877
scheme_name        28166
permit              3056
construction_year      0
extraction_type        0
management             0
payment_type           0
quality_group          0
quantity               0
source_type            0
waterpoint_type        0
status_group           0
dtype: int64
```

Preprocessing. Data cleaning

Ввод [17]:
```
# 'wpt_name' has a lot of unique classes, will drop it.
df.wpt_name.nunique()
```

Out[17]: 37400

Ввод [18]:
```
# 'subvillage' has a lot of unique classes, will drop it.
df.subvillage.nunique()
```

Out[18]: 19287

Ввод [19]:
```
# Have a lot of missing data, and a lot of unique classes, will drop it.
df.scheme_name.nunique()
```

Out[19]: 2696

Ввод [20]:
```
# 'lga' and 'ward' represent the same but 'lga' has large scale. Will drop 'ward'
df.groupby(['lga', 'ward']).count()[60:80]
```

Out[20]:

lga	ward	id	amount_tsh	date_recorded	funder	gps_height	installer	longitude	latitude	wpt_name	basin	subvillage	region_code	district_
Bagamoyo	Mbwewe	72	72	72	72	72	72	72	72	72	72	72	72	
	Miono	40	40	40	40	40	40	40	40	40	40	40	40	
	Mkange	13	13	13	12	13	12	13	13	13	13	13	13	
	Msata	73	73	73	73	73	73	73	73	73	73	73	73	
	Talawanda	5	5	5	5	5	5	5	5	5	5	5	5	
	Ubenazamozi	8	8	8	8	8	8	8	8	8	8	8	8	
	Vigwaza	26	26	26	26	26	26	26	26	26	26	26	26	
	Yombo	106	106	106	106	106	106	106	106	106	106	106	106	
	Zinga/Ikerege	141	141	141	141	141	141	141	141	141	141	141	141	
Bahi	Babayu	10	10	10	10	10	10	10	10	10	10	10	10	
	Bahi	17	17	17	17	17	17	17	17	17	17	17	17	
	Chali	5	5	5	5	5	5	5	5	5	5	5	5	
	Chibelela	12	12	12	12	12	12	12	12	12	12	12	12	
	Chikola	10	10	10	10	10	10	10	10	10	10	10	10	
	Chipanga	14	14	14	14	14	14	14	14	14	14	14	14	
	Ibihwa	9	9	9	9	9	9	9	9	9	9	9	9	
	Ibugule	6	6	6	6	6	6	6	6	6	6	6	6	
	Ilindi	4	4	4	4	4	4	4	4	4	4	4	4	
	Kigwe	26	26	26	26	26	26	26	26	26	26	26	26	
	Lamaiti	12	12	12	11	12	11	12	12	12	12	12	12	

Preprocessing. Data cleaning

Ввод [21]:
```
df.drop(labels=['recorded_by', 'wpt_name', 'subvillage', 'scheme_name', 'ward'], axis=1, inplace=True)
```

Ввод [22]:
```
df.head(2)
```

Out[22]:

	id	amount_tsh	date_recorded	funder	gps_height	installer	longitude	latitude	basin	region_code	district_code	lga	population	public_m
0	69572	6000.0	2011-03-14	Roman	1390	Roman	34.938093	-9.856322	Lake Nyasa	11	5	Ludewa	109	
1	8776	0.0	2013-03-06	Grumeti	1399	GRUMETI	34.698766	-2.147466	Lake Victoria	20	2	Serengeti	280	

Preprocessing. Filling NAs

```
Ввод [16]: df.isnull().sum()

Out[16]: id                     0
         amount_tsh             0
         date_recorded          0
         funder              3635
         gps_height             0
         installer           3655
         longitude              0
         latitude               0
         wpt_name               0
         basin                  0
         subvillage           371
         region_code            0
         district_code          0
         lga                    0
         ward                   0
         population             0
         public_meeting      3334
         recorded_by            0
         scheme_management   3877
         scheme_name        28166
         permit              3056
         construction_year      0
         extraction_type        0
         management             0
         payment_type           0
         quality_group          0
         quantity               0
         source_type            0
         waterpoint_type        0
         status_group           0
         dtype: int64
```

```
Ввод [25]: df.scheme_management.replace(to_replace='None', value=np.nan, inplace=True)
           df.scheme_management.value_counts()

Out[25]: VWC                 36793
         WUG                  5206
         Water authority      3153
         WUA                  2883
         Water Board          2748
         Parastatal           1680
         Private operator     1063
         Company              1061
         Other                 766
         SWC                    97
         Trust                  72
         Name: scheme_management, dtype: int64
```

```
Ввод [26]: print("True's part is", (51011 / (51011 + 5055)))
           df.public_meeting.value_counts()

           True's part is 0.909838404737274

Out[26]: True     51011
         False     5055
         Name: public_meeting, dtype: int64
```

```
Ввод [27]: df.public_meeting.fillna(value=True, inplace=True)
```

```
Ввод [28]: df.public_meeting.value_counts()

Out[28]: True     54345
         False     5055
         Name: public_meeting, dtype: int64
```

Preprocessing. Filling NAs

Dealing with construction year

Ввод [36]:
```
df.construction_year.value_counts()[:15]
```

Out[36]:
```
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
Name: construction_year, dtype: int64
```

Ввод [37]:
```
# 1/3 jf 'construction_year' is filled with zeros. So, it's NAs.
df.construction_year.replace(to_replace=0, value=np.nan, inplace=True)
```

Ввод [38]:
```
df.isnull().sum()
```

Out[38]:
```
id                     0
amount_tsh             0
date_recorded          0
funder              3635
gps_height             0
installer           3655
longitude              0
latitude               0
basin                  0
region_code            0
district_code          0
lga                    0
population             0
public_meeting         0
scheme_management   3878
permit              3056
construction_year  20709
extraction_type        0
management             0
payment_type           0
quality_group          0
quantity               0
source_type            0
waterpoint_type        0
status_group           0
dtype: int64
```

Preprocessing. Filling NAs

Ввод [41]:
```
df.groupby('region_code').construction_year.value_counts()[:20]
```

Out[41]:
```
region_code  construction_year
2            2000.0               484
             1990.0               169
             2010.0               168
             2012.0               167
             2002.0               160
             2008.0               150
             2007.0               144
             2009.0               144
             1998.0               133
             2011.0               115
             2003.0               107
             2005.0               104
             1978.0                95
             1980.0                94
             1970.0                84
             2006.0                84
             1999.0                77
             1995.0                58
             2013.0                39
             2001.0                38
Name: construction_year, dtype: int64
```

Ввод [42]:
```
# We will fill NAs in 'construction_year' with median with regard to 'region_code'
df.construction_year.fillna(df.groupby('region_code').construction_year.transform('median'), inplace=True)

# We don't cover all of them. Let's do it again with 'district_code'
df.construction_year.fillna(df.groupby('district_code').construction_year.transform('median'), inplace=True)

# Still have some. Now it's just median over all values
df.construction_year.fillna(df.construction_year.median(), inplace=True)
```

Preprocessing. Filling NAs

Dealing with geographic coordinates

Ввод [48]:
```
df.longitude.value_counts().sort_index(ascending=True)[:80]
```

Out[48]:
```
0.000000     1812
29.607122       1
29.607201       1
29.610321       1
29.610965       1
               ...
40.323402       1
40.325226       1
40.325240       1
40.344301       1
40.345193       1
Name: longitude, Length: 57516, dtype: int64
```

Ввод [49]:
```
df.latitude.value_counts().sort_index(ascending=True)[:80]
```

Out[49]:
```
-1.164944e+01       1
-1.164838e+01       1
-1.158630e+01       1
-1.156858e+01       1
-1.156680e+01       1
                  ...
-9.991170e-01       1
-9.990121e-01       1
-9.989160e-01       1
-9.984644e-01       1
-2.000000e-08    1812
Name: latitude, Length: 57517, dtype: int64
```

[52]:
```
# Let's combine well's coordinates accroding to region nad district and fill NAs with mean.
df.longitude.fillna(df.groupby(['region_code', 'district_code']).longitude.transform('mean'), inplace=True)
```

[53]:
```
df.isnull().sum()
```

[53]:
```
id                     0
amount_tsh             0
date_recorded          0
funder              3635
gps_height             0
installer           3655
longitude            488
latitude               0
basin                  0
region_code            0
district_code          0
lga                    0
population             0
public_meeting         0
scheme_management    3878
permit               3056
construction_year      0
extraction_type        0
management             0
payment_type           0
quality_group          0
quantity               0
source_type            0
waterpoint_type        0
status_group           0
dtype: int64
```

[54]:
```
# Still have sum. Let's group only by region.

df.longitude.fillna(df.groupby('region_code').longitude.transform('mean'), inplace=True)
```

[55]:
```
df.isnull().sum()
```

[55]:
```
id                     0
amount_tsh             0
date_recorded          0
funder              3635
gps_height             0
installer           3655
longitude              0
latitude               0
basin                  0
region_code            0
district_code          0
lga                    0
population             0
public_meeting         0
scheme_management    3878
permit               3056
construction_year      0
extraction_type        0
management             0
payment_type           0
quality_group          0
quantity               0
source_type            0
waterpoint_type        0
status_group           0
dtype: int64
```

Preprocessing. Filling NAs

As well we have to many classes in both 'funder' and 'installer',

let's create class 'other' for NAs.

[56]:
```
df.funder.fillna(value='other', inplace=True)
df.installer.fillna(value='other', inplace=True)
df.isnull().sum()
```

[56]:
```
id                     0
amount_tsh             0
date_recorded          0
funder                 0
gps_height             0
installer              0
longitude              0
latitude               0
basin                  0
region_code            0
district_code          0
lga                    0
population             0
public_meeting         0
scheme_management    3878
permit               3056
construction_year      0
extraction_type        0
management             0
payment_type           0
quality_group          0
quantity               0
source_type            0
waterpoint_type        0
status_group           0
dtype: int64
```

Preprocessing. Filling NAs

Let's group some managmenet features.

[57]:
```
print(df.scheme_management.nunique())
print(df.management.nunique())
```

```
11
12
```

[58]:
```
df[df.scheme_management.isnull() == True]['management'].unique()
```

[58]:
```
array(['other', 'wug', 'vwc', 'private operator', 'unknown', 'parastatal',
       'company', 'wua', 'water authority', 'trust'], dtype=object)
```

[59]:
```
df.groupby('scheme_management')['management'].value_counts()[:20]
```

[59]:
```
scheme_management  management
Company            company              674
                   private operator     224
                   vwc                  135
                   parastatal            25
                   other                  2
                   wug                    1
Other              other                519
                   wug                  113
                   private operator      64
                   vwc                   41
                   water authority       19
                   unknown                8
                   parastatal             1
                   trust                  1
Parastatal         parastatal          1568
                   private operator      59
                   vwc                   47
                   unknown                4
                   water authority        1
                   water board            1
Name: management, dtype: int64
```

[60]:
```
# There are a lot of cross-variable classes. I assume, that the most suitable fill there will be class 'other'

df.scheme_management.fillna(value='other', inplace=True)
```

[61]:
```
df.isnull().sum()
```

[61]:
```
id                     0
amount_tsh             0
date_recorded          0
funder                 0
gps_height             0
installer              0
longitude              0
latitude               0
basin                  0
region_code            0
district_code          0
lga                    0
population             0
public_meeting         0
scheme_management      0
permit              3056
construction_year      0
extraction_type        0
management             0
payment_type           0
quality_group          0
quantity               0
source_type            0
waterpoint_type        0
status_group           0
dtype: int64
```

Preprocessing. Filling NAs

Dealing with permit.

[62]:
```
print("True's part is", (38852 / (17492 + 38852) * 100))
print('This much is missing:', (59400 - 17492 - 38852) / 59400 * 100)
df.permit.value_counts()
```

```
True's part is 68.95499077097827
This much is missing: 5.144781144781145
True     38852
False    17492
Name: permit, dtype: int64
```

[63]:
```
df.groupby(['public_meeting']).permit.value_counts()
```

[63]:
```
public_meeting  permit
False           False      2380
                True       2308
True            True      36544
                False     15112
Name: permit, dtype: int64
```

[64]:
```
df[df.permit.isnull() == True].public_meeting.value_counts()
```

[64]:
```
True     2689
False     367
Name: public_meeting, dtype: int64
```

[65]:
```
# Will fill NA's with regard to obsrevations.

df.permit.fillna(method='pad', inplace=True)
```

[66]:
```
# Ratio of permit's classes with regard to public_meeting did not changed.

df.groupby(['public_meeting']).permit.value_counts()
```

[66]:
```
public_meeting  permit
False           True       2561
                False      2494
True            True      38396
                False     15949
Name: permit, dtype: int64
```

Preprocessing. Feature classes alignment

During observation of 'funder' and 'installer' we noted, that it has zeroes as classes. Also, there are some classes, that have different spelling, but they looks the same.

Let's put them into class 'other'. Unfortinantly, we cannot decrase amount of classes.

[79]:
```
df.installer.replace(to_replace='0', value='other', inplace=True)
df.funder.replace(to_replace='0', value='other', inplace=True)
```

[80]:
```
df.installer.value_counts()[:80].sum() / 5940
```

[80]:
```
7.651515151515151
```

[81]:
```
df.funder.value_counts()[:80].sum() / 59400
```

[81]:
```
0.771969696969697
```

[94]:
```
df.groupby('installer').funder.value_counts()[110:115]
```

[94]:
```
installer        funder
Adra /Community  Adra            55
Adra/ Community  Adra            17
Adra/Community   Adra            60
Adrs             Adra             1
Af               African Relie    15
Name: funder, dtype: int64
```

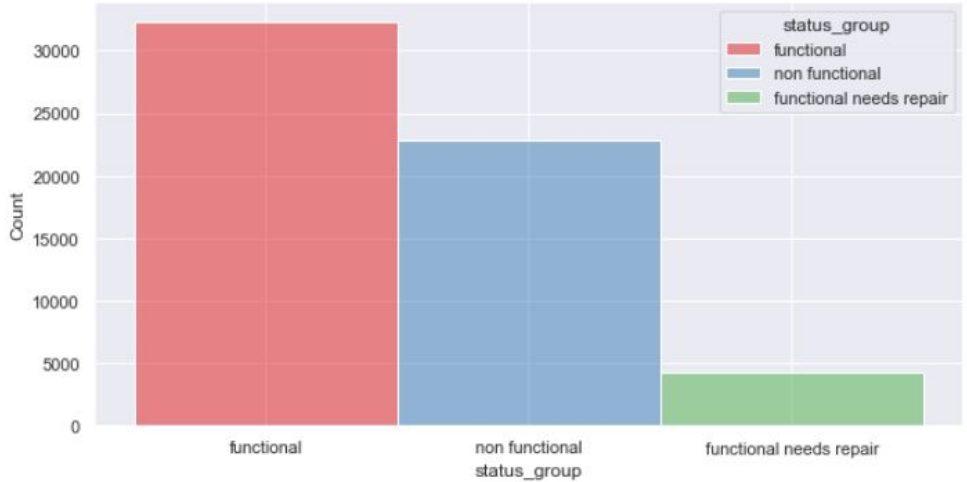
[82]:
```
# Wow, so many classes with different representations (check the groupby above). Let's work on it.

df.funder = df.funder.str.lower().str.replace(' ', '_').str.replace('/', '_').str.replace('.', '_').str.replace('___', '_').str.replace('__', '_')
df.installer = df.installer.str.lower().str.replace(' ', '_').str.replace('/', '_').str.replace('.', '_').str.replace('___', '_').str.replace('__', '_')
df.basin = df.basin.str.lower().str.replace(' ', '_').str.replace('/', '_').str.replace('.', '_').str.replace('___', '_').str.replace('__', '_')
df.lga = df.lga.str.lower().str.replace(' ', '_').str.replace('/', '_').str.replace('.', '_').str.replace('___', '_').str.replace('__', '_')
df.scheme_management = df.scheme_management.str.lower().str.replace(' ', '_').str.replace('/', '_').str.replace('.', '_').str.replace('___', '_').str.replace('__', '_')
df.extraction_type = df.extraction_type.str.lower().str.replace(' ', '_').str.replace('/', '_').str.replace('.', '_').str.replace('___', '_').str.replace('__', '_')
df.management = df.management.str.lower().str.replace(' ', '_').str.replace('/', '_').str.replace('.', '_').str.replace('___', '_').str.replace('__', '_')
df.payment_type = df.payment_type.str.lower().str.replace(' ', '_').str.replace('/', '_').str.replace('.', '_').str.replace('___', '_').str.replace('__', '_')
df.quality_group = df.quality_group.str.lower().str.replace(' ', '_').str.replace('/', '_').str.replace('.', '_').str.replace('___', '_').str.replace('__', '_')
df.quantity = df.quantity.str.lower().str.replace(' ', '_').str.replace('/', '_').str.replace('.', '_').str.replace('___', '_').str.replace('__', '_')
df.source_type = df.source_type.str.lower().str.replace(' ', '_').str.replace('/', '_').str.replace('.', '_').str.replace('___', '_').str.replace('__', '_')
df.waterpoint_type = df.waterpoint_type.str.lower().str.replace(' ', '_').str.replace('/', '_').str.replace('.', '_').str.replace('___', '_').str.replace('__', '_')
```

Exploratory Analysis via Visualization

Ввод [72]:
```
sns.set(rc={'figure.figsize': (10,5)})
sns.histplot(data=df, x='status_group', palette='Set1', hue='status_group')
```

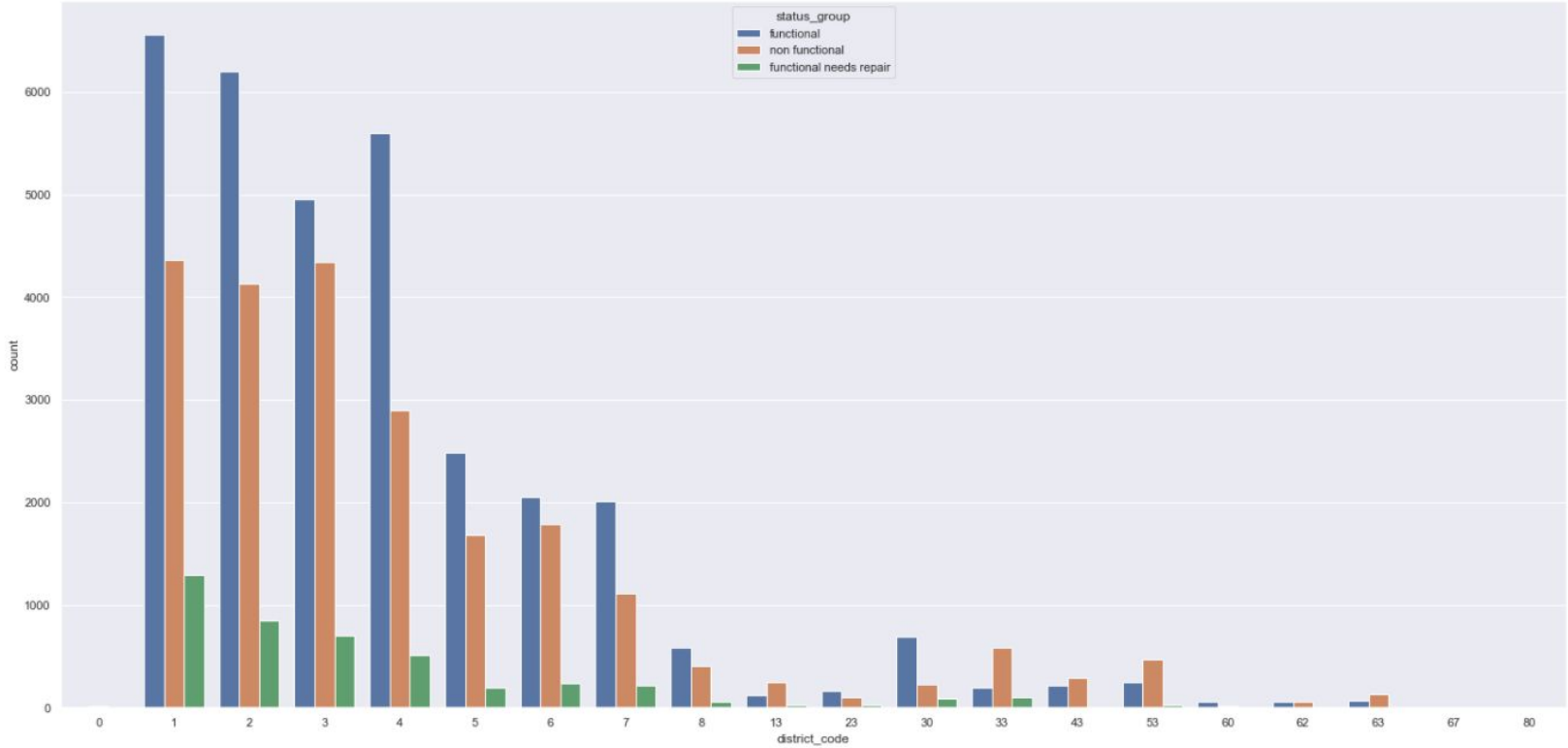
Out[72]: <AxesSubplot:xlabel='status_group', ylabel='Count'>



Exploratory Analysis via Visualization

[69]:
```
plt.figure(figsize=(25,12))
sns.countplot(data=df, x='district_code', hue='status_group')
```

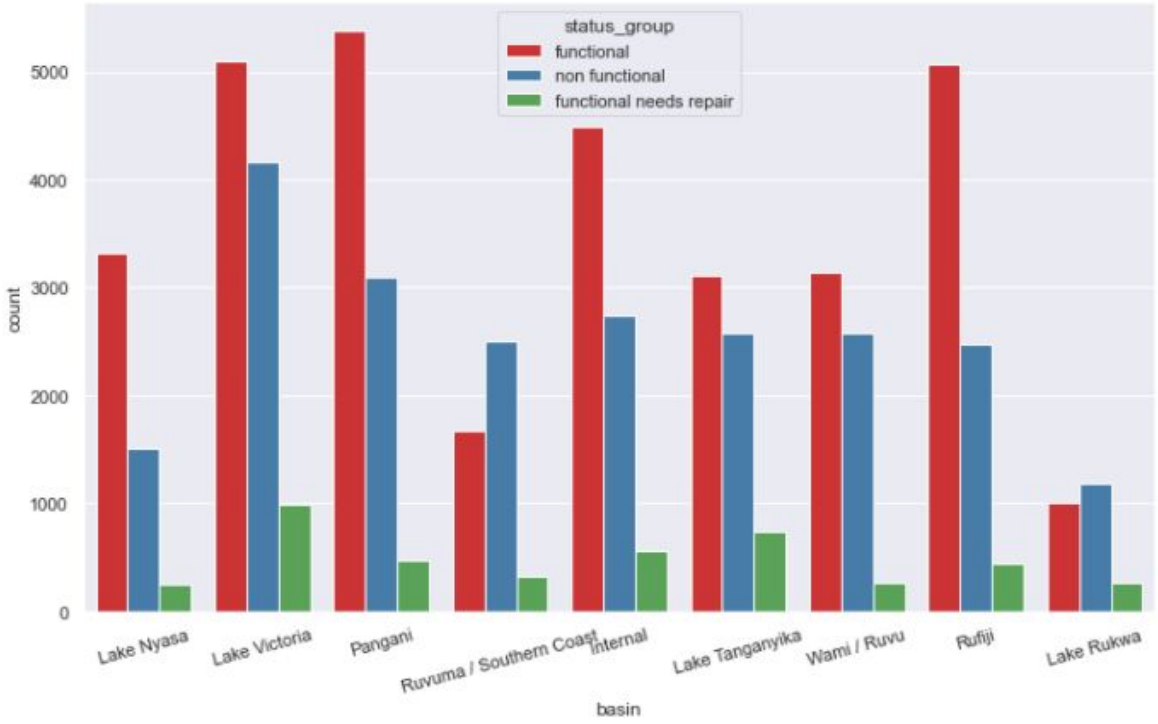
[69]: <AxesSubplot:xlabel='district_code', ylabel='count'>



Exploratory Analysis via Visualization

```
[77]: sns.set(rc={'figure.figsize': (12,7)})
      plt.xticks(rotation=15)
      sns.countplot(data=df, x='basin', palette='Set1', hue='status_group')
```

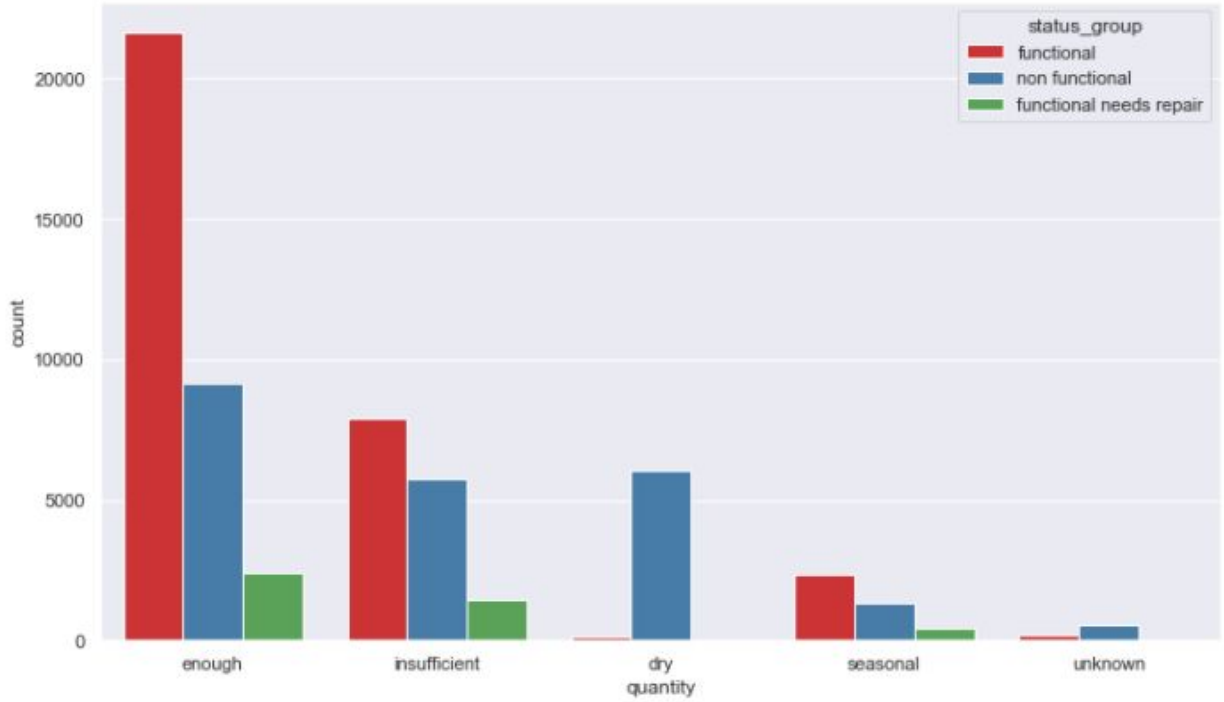
```
[77]: <AxesSubplot:xlabel='basin', ylabel='count'>
```



Exploratory Analysis via Visualization

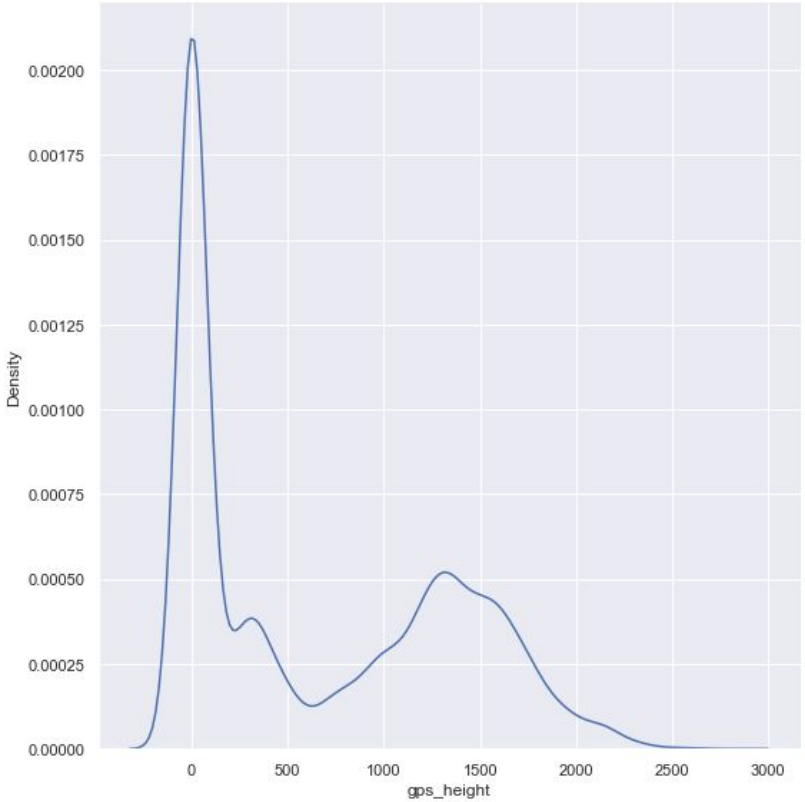
```
[75]: sns.set(rc={'figure.figsize': (12,7)})
      sns.countplot(data=df, x='quantity', palette='Set1', hue='status_group')
```

```
[75]: <AxesSubplot:xlabel='quantity', ylabel='count'>
```



Exploratory Analysis via Visualization

Ввод [70]:
```
sns.displot(data=df, x='gps_height', kind='kde', height=8);
```



Feature Engineering

Let's create variable 'age' from 'date_recorded' and 'construction_year'. Then drop them both

Ввод [84]:
```
df['age'] = df[['date_recorded', 'construction_year']].apply(lambda x: x[0].year - x[1], axis=1)
df.drop(labels=['date_recorded', 'construction_year'], axis=1, inplace=True)
```

Ввод [85]:
```
df.head(3)
```

Out[85]:

_type	management	payment_type	quality_group	quantity	source_type	waterpoint_type	status_group	age
ravity	vwc	annually	good	enough	spring	communal_standpipe	functional	12.0
ravity	wug	never_pay	good	insufficient	rainwater_harvesting	communal_standpipe	functional	3.0
ravity	vwc	per_bucket	good	enough	dam	communal_standpipe_multiple	functional	4.0

Data Exploration

Ввод [86]:
```
df.management.value_counts()
```

Out[86]:
```
vwc                 40507
wug                  6515
water_board          2933
wua                  2535
private_operator     1971
parastatal           1768
water_authority       904
other                 844
company               685
unknown               561
other_-_school         99
trust                  78
Name: management, dtype: int64
```

Ввод [87]:
```
df.scheme_management.value_counts()
```

Out[87]:
```
vwc                 36793
wug                  5206
other                4644
water_authority      3153
wua                  2883
water_board          2748
parastatal           1680
private_operator     1063
company              1061
swc                    97
trust                  72
Name: scheme_management, dtype: int64
```

Features 'management' and 'scheme_management' both has "classes-outliers" with not significant number of occurences.

Let's drop this rows (it's less then a percent). Also change class 'unknown' of feature 'managment' to class 'other'

Ввод [88]:
```
df.management.replace(to_replace='other_-_school', value=np.nan, inplace=True)
df.management.replace(to_replace='trust', value=np.nan, inplace=True)

df.scheme_management.replace(to_replace='swc', value=np.nan, inplace=True)
df.scheme_management.replace(to_replace='trust', value=np.nan, inplace=True)

df.dropna(axis=0, inplace=True)

df.management.replace(to_replace='unknown', value='other', inplace=True)
```

Ввод [89]:
```
df.management.value_counts()
```

Out[89]:
```
vwc                 40492
wug                  6514
water_board          2932
wua                  2535
private_operator     1970
parastatal           1767
other                1405
water_authority       903
company               684
Name: management, dtype: int64
```

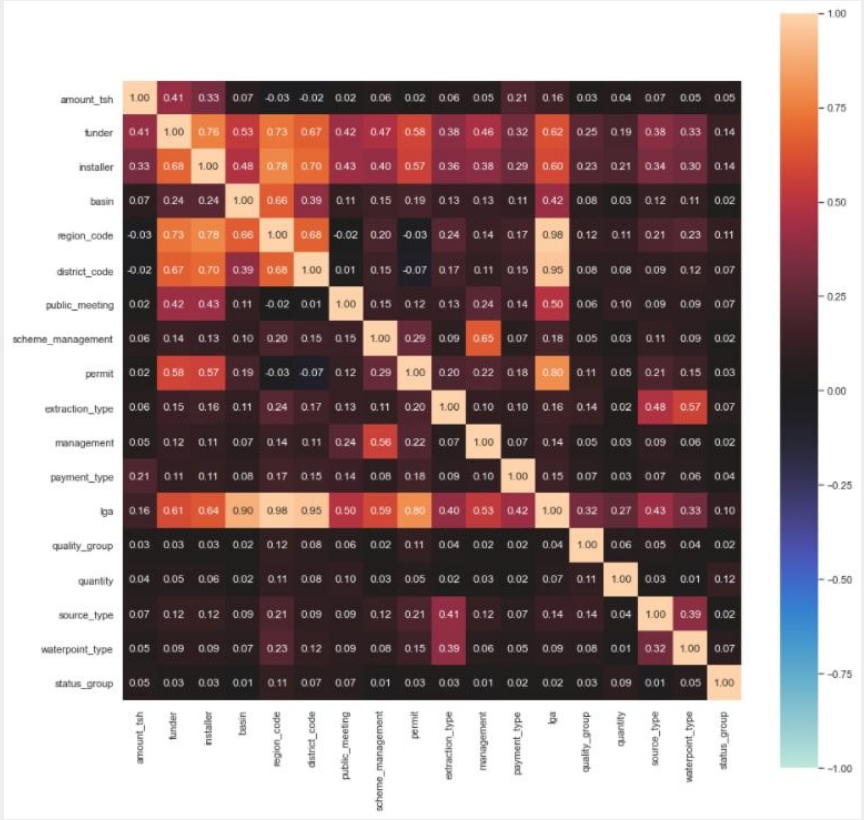
Ввод [90]:
```
df.scheme_management.value_counts()
```

Out[90]:
```
vwc                 36779
wug                  5206
other                4642
water_authority      3143
wua                  2883
water_board          2746
parastatal           1680
private_operator     1062
company              1061
Name: scheme_management, dtype: int64
```

Data Exploration

```
# Cramer's V - analog of the correlation but for
# categorical target and features.
import scipy.stats as ss
from dython.nominal import associations

cramer_v = ['amount_tsh', 'funder', 'installer',
            'basin', 'region_code', 'district_code',
            'public_meeting', 'scheme_management',
            'permit', 'extraction_type', 'management',
            'payment_type', 'lga', 'quality_group',
            'quantity', 'source_type', 'waterpoint_type',
            'status_group']
associations(df[cramer_v], nom_nom_assoc='theil', figsize=(15,15));
```



Train/Test Split. Model Building

```
from sklearn.model_selection import train_test_split

X = df.drop(labels=['id', 'status_group'], axis=1)
y = df[['status_group']]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, random_state=7)
```

```
print(f'Verification of splitiing: {X.shape[0]} / ({X_test.shape[0]} + {X_train.shape[0]}) = ', X_train.shape[0] / (X_test.shape[0] + X_train.shape[0]))
```

```
Verification of splitiing: 53460 / (5940 + 53460) =  0.9
```

```
y_train.status_group.unique()
```

```
array(['functional', 'non functional', 'functional needs repair'],
      dtype=object)
```

```
y_train[y_train.status_group == 'functional'].shape[0] / (y_test[y_test.status_group == 'functional'].shape[0] + y_train[y_train.status_group == 'functional'].shape[0])
```

```
0.9001208964940016
```

```
y_train[y_train.status_group == 'non functional'].shape[0] / (y_test[y_test.status_group == 'non functional'].shape[0] + y_train[y_train.status_group == 'non functional'].shape[0])
```

```
0.8997546442341395
```

```
y_train[y_train.status_group == 'functional needs repair'].shape[0] / (y_test[y_test.status_group == 'functional needs repair'].shape[0] + y_train[y_train.status_group == 'functional needs repair'].shape[0])
```

```
0.9003937919851749
```

[138]:

```
from catboost import CatBoostClassifier, Pool
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import f1_score

cat_1 = CatBoostClassifier(iterations=500, learning_rate=0.03, depth=6,
                           l2_leaf_reg=3, loss_function='MultiClass', border_count=32, ctr_target_border_count=50)

idx_cat_feat = [1, 3, 6, 7, 8, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19]

cat_train = Pool(X_train, y_train, cat_features=idx_cat_feat)
cat_test = Pool(X_test, y_test, cat_features=idx_cat_feat)
cat_1.fit(cat_train)
```

...

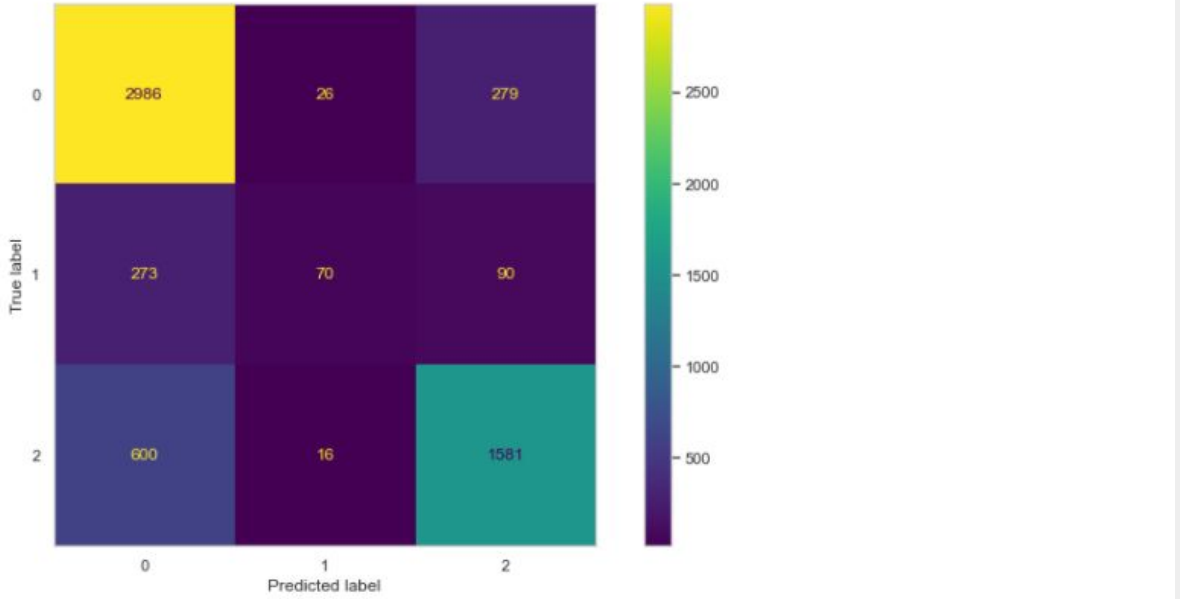
Dry Run Results

[140]:
```
y_hat = cat_1.predict(cat_test)
f1_score(y_test, y_hat, average='micro')
```

[140]:
```
0.7831447390643473
```

[155]:
```
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay

sns.set_style("whitegrid", {'axes.grid' : False})
conf_matr_1 = ConfusionMatrixDisplay(confusion_matrix(y_test, y_hat), display_labels=cat_1.classes_).plot()
```



Parameter Tuning of CatBoost

```
from catboost import CatBoostClassifier, Pool
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import f1_score

idx_cat_feat = [1, 3, 6, 7, 8, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19]
cat = CatBoostClassifier(iterations=500, cat_features=idx_cat_feat, task_type="GPU", devices='0:1',
                         thread_count=4, loss_function='MultiClass', ctr_target_border_count=50)

hyper_params = {'depth': list(range(6,9)),
                'learning_rate': [0.01, 0.1, 1],
                'l2_leaf_reg':[1, 3, 5],
                'border_count': [30, 35]}

grid = GridSearchCV(cat, hyper_params, cv=5)

grid.fit(X_train, y_train)
best_param = grid.best_params_
best_param
```

Show hidden output

```
[80] best_param = grid.best_params_
     best_param
```

```
{'border_count': 35, 'depth': 8, 'l2_leaf_reg': 1, 'learning_rate': 0.1}
```

```
[84] cat_opt = CatBoostClassifier(**best_param, iterations=500, cat_features=idx_cat_feat, task_type="GPU", devices='0:1',
                                  thread_count=4, loss_function='MultiClass', ctr_target_border_count=50)
     cat_opt.fit(X_train, y_train)
     f1_score(y_test, cat_opt.predict(X_test), average='micro')
```

Show hidden output

```
0.7976693126161122
```

Parameter Tuning of Random Forest

[143]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import f1_score

rfc = RandomForestClassifier()

col_cod = ['funder', 'installer', 'basin', 'region_code', 'district_code',
           'public_meeting', 'scheme_management', 'permit', 'extraction_type',
           'management', 'payment_type', 'quality_group', 'quantity',
           'source_type', 'waterpoint_type']

for col in col_cod:
    X_train[col] = LabelEncoder().fit_transform(X_train[col])
    X_test[col] = LabelEncoder().fit_transform(X_test[col])

hyper_param = {'max_depth': [80, 90],
               'max_features': ['auto', 'sqrt'],
               'min_samples_leaf': [3, 5],
               'min_samples_split': [8, 10],
               'n_estimators': [100, 200, 300]}


grid_rfc = GridSearchCV(rfc, hyper_param, scoring='f1_micro', n_jobs=-1)

grid_rfc.fit(X_train, y_train.values.ravel())
```

[143]:
```
GridSearchCV(estimator=RandomForestClassifier(), n_jobs=-1,
             param_grid={'max_depth': [80, 90],
                         'max_features': ['auto', 'sqrt'],
                         'min_samples_leaf': [3, 5],
                         'min_samples_split': [8, 10],
                         'n_estimators': [100, 200, 300]},
             scoring='f1_micro')
```

[145]:
```
y_hat = grid_rfc.predict(X_test)
print(grid_rfc.best_params_)
f1_score(y_test, y_hat, average='micro')
```

```
{'max_depth': 80, 'max_features': 'auto', 'min_samples_leaf': 3, 'min_samples_split': 8, 'n_estimators': 200}
```

[145]:
```
0.7802736024320216
```

Final Model

```
[85] hyper_params_2 = {'depth': list(range(6,9)),
                       'learning_rate': [0.05, 0.10, 0.15],
                       'l2_leaf_reg':[0.1, 1, 1.5],
                       'border_count': [34, 36]}

     grid_2 = GridSearchCV(cat, hyper_params_2, cv=5)

     grid_2.fit(X_train, y_train)
```

Show hidden output

```
[86] best_param_2 = grid_2.best_params_
     cat_opt_2 = CatBoostClassifier(**best_param_2, iterations=500, cat_features=idx_cat_feat, task_type="GPU", devices='0:1',
                                    thread_count=4, loss_function='MultiClass', ctr_target_border_count=50)
     cat_opt_2.fit(X_train, y_train)
```

Show hidden output

```
[88] print(best_param_2)
     f1_score(y_test, cat_opt_2.predict(X_test), average='micro')

     {'border_count': 36, 'depth': 8, 'l2_leaf_reg': 0.1, 'learning_rate': 0.15}
     0.8057760513426787
```

Final Model

Recall for ‘non-functional’: 76%

Recall for ‘need repair’: 28%

Recall for ‘functional’: 91%

```
[90] from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay

     sns.set(rc={'figure.figsize': (12,12)})
     sns.set_style("whitegrid", {'axes.grid' : False})
     conf_matr_2 = ConfusionMatrixDisplay(confusion_matrix(y_test, cat_opt_2.predict(X_test)), display_labels=cat_opt_2.classes_).plot()
```