phase_3 project

Analyzing the water pumps' functionality of wells in Tanzania

By:Esther Nyawera

Project overview

Tanzania, a country facing challenges in providing clean water to its over 57 million population, struggles with maintaining and repairing existing water wells. This project focuses on constructing a classifier to predict the condition of water wells across the country. By leveraging data encompassing pump types, water source,quality,quantity and other features, the objective is to aid NGOs or the Tanzanian government in identifying wells, their functionality and those in need of repair or maintenance. This initiative aims to enhance water accessibility and availability for the population, especially in remote or underprivileged areas.

Problem Understanding

Water wells in Tanzania are essential for providing clean water but often suffer breakdowns or inadequate maintenance, impacting water availability. With numerous dispersed wells, systematically identifying non-functional or deteriorating ones is challenging. This project seeks to address this issue by employing machine learning to predict well conditions, facilitating targeted interventions for enhancing water infrastructure.

Project objectives

The main aim is to create a machine learning classifier for predicting water well conditions in Tanzania, classifying wells as functional, needing repair, or non-functional based on historical data in respect to:-

1. What aspects of the well affect functionality?
2. Regions and the state of their wells
3. Water quality, quantity and effects to funtionality
4. Data provided and how well it can be used to assess and train our model

The data.

In this project i will use data from drivendata.org

- -https://www.drivendata.org/competitions/7/pump-it-up-data-mining-the-water-table/data/
- The data in the datasetcontains information from Taarifa and the Tanzanian Ministry of Water. With over 59,000 data points.

Import necessary libraries

In [849]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, OneHotEncoder, StandardScaler
from imblearn.over_sampling import SMOTE
from sklearn.compose import ColumnTransformer
from imblearn.pipeline import Pipeline as ImbPipeline

from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

from sklearn.metrics import plot_confusion_matrix
import pickle
```

Load the data

In [850]:

```
train_values_df= pd.read_csv('data/trainingset_values.csv')
train_labels_df = pd.read_csv('data/trainingset_labels.csv')
```

Start on Data analysis

In [851]:

```
train_labels_df.info
```

Out[851]:

```
<bound method DataFrame.info of            id    status_group
0       69572      functional
1        8776      functional
2       34310      functional
3       67743  non functional
4       19728      functional
...       ...             ...
59395   60739      functional
59396   27263      functional
59397   37057      functional
59398   31282      functional
59399   26348      functional

[59400 rows x 2 columns]>
```

In [852]:

```
train_labels_df.columns
```

Out[852]:

```
Index(['id', 'status_group'], dtype='object')
```

In [853]:

```
train_values_df.columns
```

Out[853]:

```
Index(['id', 'amount_tsh', 'date_recorded', 'funder', 'gps_height',
       'installer', 'longitude', 'latitude', 'wpt_name', 'num_private',
       'basin', 'subvillage', 'region', 'region_code', 'district_code', 'lga',
       'ward', 'population', 'public_meeting', 'recorded_by',
       'scheme_management', 'scheme_name', 'permit', 'construction_year',
       'extraction_type', 'extraction_type_group', 'extraction_type_class',
       'management', 'management_group', 'payment', 'payment_type',
       'water_quality', 'quality_group', 'quantity', 'quantity_group',
       'source', 'source_type', 'source_class', 'waterpoint_type',
       'waterpoint_type_group'],
      dtype='object')
```

In [854]:

```
train_df = pd.merge(train_labels_df, train_values_df, on='id')
train_df.head()
```

Out[854]:

	id	status_group	amount_tsh	date_recorded	funder	gps_height	installer	longitude	latitude	wpt_name	...	pa
0	69572	functional	6000.0	2011-03-14	Roman	1390	Roman	34.938093	-9.856322	none	...	
1	8776	functional	0.0	2013-03-06	Grumeti	1399	GRUMETI	34.698766	-2.147466	Zahanati	...	
2	34310	functional	25.0	2013-02-25	Lottery Club	686	World vision	37.460664	-3.821329	Kwa Mahundi	...	
3	67743	non functional	0.0	2013-01-28	Unicef	263	UNICEF	38.486161	-11.155298	Zahanati Ya Nanyumbu	...	
4	19728	functional	0.0	2011-07-13	Action In A	0	Artisan	31.130847	-1.825359	Shuleni	...	

5 rows × 41 columns

In [855]:

```
train_df.info()
train_df.columns
train_df.shape
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 59400 entries, 0 to 59399
Data columns (total 41 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   id                     59400 non-null  int64
 1   status_group           59400 non-null  object
 2   amount_tsh             59400 non-null  float64
 3   date_recorded          59400 non-null  object
 4   funder                 55765 non-null  object
 5   gps_height             59400 non-null  int64
 6   installer              55745 non-null  object
 7   longitude              59400 non-null  float64
 8   latitude               59400 non-null  float64
 9   wpt_name               59400 non-null  object
 10  num_private            59400 non-null  int64
 11  basin                  59400 non-null  object
 12  subvillage             59029 non-null  object
 13  region                 59400 non-null  object
 14  region_code            59400 non-null  int64
 15  district_code          59400 non-null  int64
```

```
 16  lga                    59400 non-null  object
 17  ward                   59400 non-null  object
 18  population             59400 non-null  int64
 19  public_meeting         56066 non-null  object
 20  recorded_by            59400 non-null  object
 21  scheme_management      55523 non-null  object
 22  scheme_name            31234 non-null  object
 23  permit                 56344 non-null  object
 24  construction_year      59400 non-null  int64
 25  extraction_type        59400 non-null  object
 26  extraction_type_group  59400 non-null  object
 27  extraction_type_class  59400 non-null  object
 28  management             59400 non-null  object
 29  management_group       59400 non-null  object
 30  payment                59400 non-null  object
 31  payment_type           59400 non-null  object
 32  water_quality          59400 non-null  object
 33  quality_group          59400 non-null  object
 34  quantity               59400 non-null  object
 35  quantity_group         59400 non-null  object
 36  source                 59400 non-null  object
 37  source_type            59400 non-null  object
 38  source_class           59400 non-null  object
 39  waterpoint_type        59400 non-null  object
 40  waterpoint_type_group  59400 non-null  object
dtypes: float64(3), int64(7), object(31)
memory usage: 19.0+ MB
```

Out[855]:

```
(59400, 41)
```

In [856]:

```
train_df.describe()
```

Out[856]:

	id	amount_tsh	gps_height	longitude	latitude	num_private	region_code	district_code	
count	59400.000000	59400.000000	59400.000000	59400.000000	5.940000e+04	59400.000000	59400.000000	59400.000000	59
mean	37115.131768	317.650385	668.297239	34.077427	-5.706033e+00	0.474141	15.297003	5.629747	
std	21453.128371	2997.574558	693.116350	6.567432	2.946019e+00	12.236230	17.587406	9.633649	
min	0.000000	0.000000	-90.000000	0.000000	-1.164944e+01	0.000000	1.000000	0.000000	
25%	18519.750000	0.000000	0.000000	33.090347	-8.540621e+00	0.000000	5.000000	2.000000	
50%	37061.500000	0.000000	369.000000	34.908743	-5.021597e+00	0.000000	12.000000	3.000000	
75%	55656.500000	20.000000	1319.250000	37.178387	-3.326156e+00	0.000000	17.000000	5.000000	
max	74247.000000	350000.000000	2770.000000	40.345193	-2.000000e-08	1776.000000	99.000000	80.000000	30

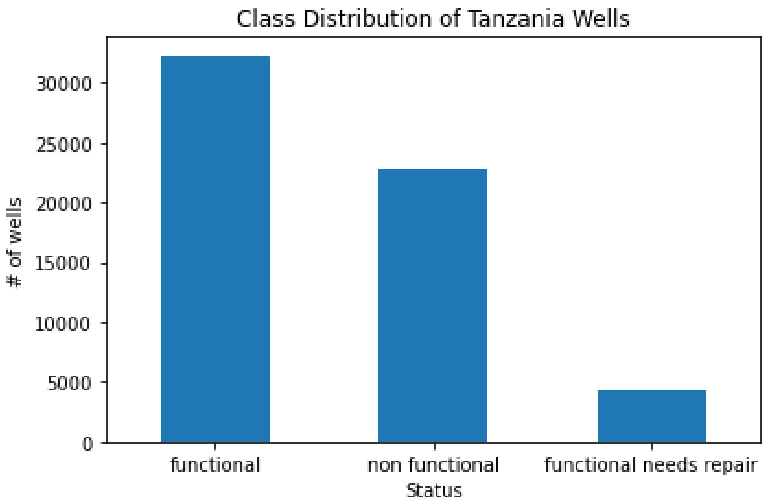
EDA

Since we are focusing on being able to distinguish functional wells from non-functional wells, we also want a general idea of how many wells in each data set are in each status group. Although "non-functional" and "functional needs repair" wells are both generally descriptive of wells that need to be repaired, I decided to keep those groups separate in case we need to prioritize wells that are still working over those that aren't for maintenance.

In [857]:

```
print(train_df['status_group'].value_counts())
train_df['status_group'].value_counts().plot(kind='bar', rot=0)
plt.title('Class Distribution of Tanzania Wells')
plt.xlabel('Status')
plt.ylabel('# of wells')
plt.tight_layout()
```

```
functional                 32259
non functional             22824
functional needs repair     4317
Name: status_group, dtype: int64
```



In [858]:

```
train_df = train_df.set_index('id')
```

In [859]:

```
train_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 59400 entries, 69572 to 26348
Data columns (total 40 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   status_group           59400 non-null  object
 1   amount_tsh             59400 non-null  float64
 2   date_recorded          59400 non-null  object
 3   funder                 55765 non-null  object
 4   gps_height             59400 non-null  int64
 5   installer              55745 non-null  object
 6   longitude              59400 non-null  float64
 7   latitude               59400 non-null  float64
 8   wpt_name               59400 non-null  object
 9   num_private            59400 non-null  int64
 10  basin                  59400 non-null  object
 11  subvillage             59029 non-null  object
 12  region                 59400 non-null  object
 13  region_code            59400 non-null  int64
 14  district_code          59400 non-null  int64
 15  lga                    59400 non-null  object
 16  ward                   59400 non-null  object
 17  population             59400 non-null  int64
 18  public_meeting         56066 non-null  object
 19  recorded_by            59400 non-null  object
 20  scheme_management      55523 non-null  object
 21  scheme_name            31234 non-null  object
 22  permit                 56344 non-null  object
 23  construction_year      59400 non-null  int64
 24  extraction_type        59400 non-null  object
 25  extraction_type_group  59400 non-null  object
 26  extraction_type_class  59400 non-null  object
 27  management             59400 non-null  object
 28  management_group       59400 non-null  object
```

```
 29  payment                59400 non-null  object
 30  payment_type           59400 non-null  object
 31  water_quality          59400 non-null  object
 32  quality_group          59400 non-null  object
 33  quantity               59400 non-null  object
 34  quantity_group         59400 non-null  object
 35  source                 59400 non-null  object
 36  source_type            59400 non-null  object
 37  source_class           59400 non-null  object
 38  waterpoint_type        59400 non-null  object
 39  waterpoint_type_group  59400 non-null  object
dtypes: float64(3), int64(6), object(31)
memory usage: 18.6+ MB
```

In [860]:

```python
#checking for duplicates

duplicate_rows = train_df[train_df.duplicated()]

# Displaying duplicate rows
print("Duplicate Rows:")
print(duplicate_rows)

# Counting duplicates
duplicate_count = train_df.duplicated().sum()
print(f"Number of duplicate rows: {duplicate_count}")

# Dropping duplicates
train_df = train_df.drop_duplicates()
```

```
Duplicate Rows:
                  status_group  amount_tsh date_recorded  \
id
23184               functional         0.0    2013-02-16
44709               functional         0.0    2012-10-25
69973           non functional         0.0    2012-11-04
8342   functional needs repair         0.0    2013-02-16
4307            non functional         0.0    2012-10-26
61256               functional         0.0    2013-02-16
25661               functional         0.0    2013-02-16
4532                functional         0.0    2011-07-27
11721               functional         0.0    2011-07-19
68204               functional         0.0    2011-07-18
13773           non functional         0.0    2012-10-26
17141               functional         0.0    2011-07-27
16417           non functional         0.0    2011-07-19
16967               functional         0.0    2012-10-25
3854            non functional         0.0    2012-10-26
73749           non functional         0.0    2011-08-02
16464               functional         0.0    2011-07-26
56859               functional         0.0    2011-07-19
2189                functional         0.0    2013-01-30
37998  functional needs repair         0.0    2013-01-20
18713               functional         0.0    2011-07-13
29329  functional needs repair         0.0    2012-10-25
28134               functional         0.0    2011-07-18
15716           non functional         0.0    2011-08-06
41029               functional         0.0    2012-10-26
52452               functional         0.0    2013-01-29
38894               functional         0.0    2013-02-16
39912           non functional         0.0    2012-10-26
626             non functional         0.0    2013-02-16
56194               functional         0.0    2012-10-25
21595               functional         0.0    2012-10-25
55001           non functional         0.0    2011-07-28
70312               functional         0.0    2011-07-18
58393           non functional         0.0    2012-10-25
1562                functional         0.0    2013-02-16
63207               functional         0.0    2012-10-26

                       funder  gps_height          installer  longitude  \
```

```
id
23184                    Dwsp            0                DWE    0.00000
44709                    Dwsp            0                DWE    0.00000
69973  Government Of Tanzania            0                RWE    0.00000
8342                      Pmo            0                DWE    0.00000
4307                  Holland            0            HOLLAND    0.00000
61256                   Rwssp            0                DWE    0.00000
25661                   Rwssp            0                DWE    0.00000
4532                   Hesawa            0                DWE    0.00000
11721  Government Of Tanzania            0         Government    0.00000
68204  Government Of Tanzania            0         Government    0.00000
13773                     Lwi            0                LWI    0.00000
17141                  Hesawa            0                DWE    0.00000
16417      Plan International            0  Plan Internationa    0.00000
16967                    Dwsp            0                DWE    0.00000
3854                  Holland            0            HOLLAND    0.00000
73749                  Hesawa            0             Hesawa    0.00000
16464                  Hesawa            0                DWE    0.00000
56859  Government Of Tanzania            0         Government    0.00000
2189                      Wvt            0                WVT    0.00000
37998                     Wvt            0                WVT    0.00000
18713                      He            0                 HE   31.61953
29329                    Dwsp            0                DWE    0.00000
28134  Government Of Tanzania            0         Government    0.00000
15716  Government Of Tanzania            0         Government    0.00000
41029                     Lwi            0                LWI    0.00000
52452                    Dwsp            0                DWE    0.00000
38894                    Dwsp            0                DWE    0.00000
39912                    Rwsp            0                DWE    0.00000
626                      Dwsp            0                DWE    0.00000
56194                    Dwsp            0                DWE    0.00000
21595                    Dwsp            0                DWE    0.00000
55001  Government Of Tanzania            0         Government    0.00000
70312  Government Of Tanzania            0         Government    0.00000
58393                 Holland            0            HOLLAND    0.00000
1562                     Dwsp            0                DWE    0.00000
63207                     Lwi            0                LWI    0.00000

           latitude                  wpt_name  num_private  ... payment_type  \
id                                                          ...
23184 -2.000000e-08                     Sango            0  ...      unknown
44709 -2.000000e-08                    Wazazi            0  ...      unknown
69973 -2.000000e-08                    School            0  ...    never pay
8342  -2.000000e-08                  Muungano            0  ...      unknown
4307  -2.000000e-08                     Jamii            0  ...      unknown
61256 -2.000000e-08                  Muungano            0  ...      unknown
25661 -2.000000e-08                      none            0  ...      unknown
4532  -2.000000e-08                   Bombani            0  ...    never pay
11721 -2.000000e-08                 Mulangila            0  ...   per bucket
68204 -2.000000e-08                  Hospital            0  ...    never pay
13773 -2.000000e-08                   Kwa Mdo            0  ...      unknown
17141 -2.000000e-08                   Bombani            0  ...    never pay
16417 -2.000000e-08              Elimu Maalum            0  ...    never pay
16967 -2.000000e-08                 Igunanilo            0  ...      unknown
3854  -2.000000e-08                     Jamii            0  ...      unknown
73749 -2.000000e-08                    Nyanza            0  ...      unknown
16464 -2.000000e-08                   Bombani            0  ...    never pay
56859 -2.000000e-08               K/Secondary            0  ...    never pay
2189  -2.000000e-08              Wvt Tanzania            0  ...        other
37998 -2.000000e-08  Shule Ya Msingi Mwanunui            0  ...    never pay
18713 -1.793342e+00                   Kahindu            0  ...    never pay
29329 -2.000000e-08                 Mwamusobi            0  ...      unknown
28134 -2.000000e-08                  Hospital            0  ...    never pay
15716 -2.000000e-08                   Bombani            0  ...      unknown
41029 -2.000000e-08                 Mwakuzuka            0  ...      unknown
52452 -2.000000e-08                  Isangijo            0  ...      unknown
38894 -2.000000e-08                Mwamahonza            0  ...      unknown
39912 -2.000000e-08                     Sanjo            0  ...      unknown
626   -2.000000e-08                 Serengeti            0  ...      unknown
56194 -2.000000e-08           Umoja Wa Vijana            0  ...      unknown
21595 -2.000000e-08                     Umoja            0  ...      unknown
55001 -2.000000e-08                  Mahakama            0  ...      unknown
```

```
70312 -2.000000e-08          Nersing College           0  ...    never pay
58393 -2.000000e-08                     none           0  ...      unknown
1562  -2.000000e-08                  Igolola           0  ...      unknown
63207 -2.000000e-08                  Msituni           0  ...      unknown

      water_quality quality_group      quantity  quantity_group  \
id
23184          soft          good        enough          enough
44709          soft          good        enough          enough
69973          soft          good  insufficient    insufficient
8342           soft          good        enough          enough
4307           soft          good        enough          enough
61256          soft          good        enough          enough
25661          soft          good        enough          enough
4532           soft          good  insufficient    insufficient
11721          soft          good  insufficient    insufficient
68204          soft          good  insufficient    insufficient
13773          soft          good        enough          enough
17141          soft          good  insufficient    insufficient
16417          soft          good  insufficient    insufficient
16967          soft          good        enough          enough
3854           soft          good        enough          enough
73749       unknown       unknown           dry             dry
16464          soft          good  insufficient    insufficient
56859          soft          good  insufficient    insufficient
2189           soft          good      seasonal        seasonal
37998          soft          good      seasonal        seasonal
18713          soft          good        enough          enough
29329          soft          good        enough          enough
28134          soft          good  insufficient    insufficient
15716       unknown       unknown           dry             dry
41029          soft          good        enough          enough
52452          soft          good        enough          enough
38894          soft          good        enough          enough
39912          soft          good        enough          enough
626            soft          good        enough          enough
56194          soft          good        enough          enough
21595          soft          good        enough          enough
55001       unknown       unknown           dry             dry
70312          soft          good  insufficient    insufficient
58393          soft          good        enough          enough
1562           soft          good        enough          enough
63207          soft          good        enough          enough

                     source           source_type  source_class  \
id
23184          shallow well          shallow well   groundwater
44709          shallow well          shallow well   groundwater
69973                  lake            river/lake       surface
8342           shallow well          shallow well   groundwater
4307           shallow well          shallow well   groundwater
61256          shallow well          shallow well   groundwater
25661          shallow well          shallow well   groundwater
4532           shallow well          shallow well   groundwater
11721           machine dbh              borehole   groundwater
68204           machine dbh              borehole   groundwater
13773          shallow well          shallow well   groundwater
17141          shallow well          shallow well   groundwater
16417           machine dbh              borehole   groundwater
16967          shallow well          shallow well   groundwater
3854           shallow well          shallow well   groundwater
73749          shallow well          shallow well   groundwater
16464          shallow well          shallow well   groundwater
56859           machine dbh              borehole   groundwater
2189   rainwater harvesting  rainwater harvesting       surface
37998  rainwater harvesting  rainwater harvesting       surface
18713                spring                spring   groundwater
29329          shallow well          shallow well   groundwater
28134           machine dbh              borehole   groundwater
15716                  lake            river/lake       surface
41029          shallow well          shallow well   groundwater
52452          shallow well          shallow well   groundwater
```

```
38894         shallow well         shallow well   groundwater
39912         shallow well         shallow well   groundwater
626                  other                other       unknown
56194         shallow well         shallow well   groundwater
21595         shallow well         shallow well   groundwater
55001                  dam                  dam       surface
70312          machine dbh             borehole   groundwater
58393         shallow well         shallow well   groundwater
1562          shallow well         shallow well   groundwater
63207         shallow well         shallow well   groundwater

                    waterpoint_type waterpoint_type_group
id
23184                     hand pump             hand pump
44709                     hand pump             hand pump
69973   communal standpipe multiple    communal standpipe
8342                      hand pump             hand pump
4307                          other                 other
61256                     hand pump             hand pump
25661                     hand pump             hand pump
4532                      hand pump             hand pump
11721   communal standpipe multiple    communal standpipe
68204            communal standpipe    communal standpipe
13773                     hand pump             hand pump
17141                     hand pump             hand pump
16417            communal standpipe    communal standpipe
16967                     hand pump             hand pump
3854                      hand pump             hand pump
73749                         other                 other
16464                     hand pump             hand pump
56859            communal standpipe    communal standpipe
2189             communal standpipe    communal standpipe
37998            communal standpipe    communal standpipe
18713               improved spring       improved spring
29329                     hand pump             hand pump
28134            communal standpipe    communal standpipe
15716   communal standpipe multiple    communal standpipe
41029                     hand pump             hand pump
52452                     hand pump             hand pump
38894                     hand pump             hand pump
39912                     hand pump             hand pump
626                           other                 other
56194                     hand pump             hand pump
21595                     hand pump             hand pump
55001   communal standpipe multiple    communal standpipe
70312                     hand pump             hand pump
58393                     hand pump             hand pump
1562                      hand pump             hand pump
63207                     hand pump             hand pump

[36 rows x 40 columns]
Number of duplicate rows: 36
```

In [861]:

```
train_df.shape
```

Out[861]:

```
(59364, 40)
```

In [862]:

```
# CHECKING MISSING DATA

# function for identifying with missing values
def missing_values(data):
    """
    Identify the missing values
    Drop values that have no missing values
    Return only dara with missing values
```

```
    """
    miss_val = data.isna().sum()
    percentage = (data.isna().sum() / len(data))
    missing_values = pd.DataFrame({"Missing Values": miss_val, "In Percentage": percentag
e})
    missing_values.drop(missing_values[missing_values["In Percentage"] == 0].index, inpl
ace=True)
    return missing_values
train_df_missing = missing_values(train_df)
train_df.head()
```

Out[862]:

	status_group	amount_tsh	date_recorded	funder	gps_height	installer	longitude	latitude	wpt_name	num_priv
id										
69572	functional	6000.0	2011-03-14	Roman	1390	Roman	34.938093	-9.856322	none	
8776	functional	0.0	2013-03-06	Grumeti	1399	GRUMETI	34.698766	-2.147466	Zahanati	
34310	functional	25.0	2013-02-25	Lottery Club	686	World vision	37.460664	-3.821329	Kwa Mahundi	
67743	non functional	0.0	2013-01-28	Unicef	263	UNICEF	38.486161	-11.155298	Zahanati Ya Nanyumbu	
19728	functional	0.0	2011-07-13	Action In A	0	Artisan	31.130847	-1.825359	Shuleni	

5 rows × 40 columns

In [863]:

```
train_df = train_df.dropna()
train_df.head()
```

Out[863]:

	status_group	amount_tsh	date_recorded	funder	gps_height	installer	longitude	latitude	wpt_name	num
id										
69572	functional	6000.0	2011-03-14	Roman	1390	Roman	34.938093	-9.856322	none	
34310	functional	25.0	2013-02-25	Lottery Club	686	World vision	37.460664	-3.821329	Kwa Mahundi	
9944	functional	20.0	2011-03-13	Mkinga Distric Coun	0	DWE	39.172796	-4.765587	Tajiri	
50495	functional	0.0	2013-03-15	Lawatefuka Water Supply	1368	Lawatefuka water sup	37.092574	-3.181783	Kwa John Izack Mmari	
53752	functional	0.0	2012-10-20	Biore	0	WEDECO	34.364073	-3.629333	Mwabasabi	

5 rows × 40 columns

In [864]:

```
# inspect after changes
train_df.describe()
```

Out[864]:

	amount_tsh	gps_height	longitude	latitude	num_private	region_code	district_code	population	co
count	27804.000000	27804.000000	27804.000000	2.780400e+04	27804.000000	27804.000000	27804.000000	27804.000000	
mean	480.653517	889.503201	35.211005	-5.826661e+00	0.684434	12.182564	4.721803	166.408430	
std	3537.870541	694.567669	4.579294	2.796712e+00	6.841412	16.077040	6.660303	371.883913	
min	0.000000	-90.000000	0.000000	-1.156451e+01	0.000000	1.000000	0.000000	0.000000	
25%	0.000000	80.000000	34.224465	-8.721481e+00	0.000000	3.000000	2.000000	1.000000	
50%	0.000000	1005.000000	35.865800	-4.935950e+00	0.000000	11.000000	3.000000	53.000000	
75%	150.000000	1459.000000	37.497144	-3.354595e+00	0.000000	16.000000	5.000000	200.000000	
max	250000.000000	2628.000000	40.323402	-2.000000e-08	280.000000	99.000000	80.000000	15300.000000	

Further EDA into specific columns

In [865]:

```
train_df
```

Out[865]:

	status_group	amount_tsh	date_recorded	funder	gps_height	installer	longitude	latitude	wpt_name	nu
id										
69572	functional	6000.0	2011-03-14	Roman	1390	Roman	34.938093	-9.856322	none	
34310	functional	25.0	2013-02-25	Lottery Club	686	World vision	37.460664	-3.821329	Kwa Mahundi	
9944	functional	20.0	2011-03-13	Mkinga Distric Coun	0	DWE	39.172796	-4.765587	Tajiri	
50495	functional	0.0	2013-03-15	Lawatefuka Water Supply	1368	Lawatefuka water sup	37.092574	-3.181783	Kwa John Izack Mmari	
53752	functional	0.0	2012-10-20	Biore	0	WEDECO	34.364073	-3.629333	Mwabasabi	
...	...	...	...	...	...	...	...	...	...	
67885	non functional	0.0	2011-03-16	Mkinga Distric Coun	0	DWE	38.835001	-4.880204	Mijohoroni	
47002	non functional	6.0	2013-08-03	Ces(gmbh)	1383	DWE	37.454759	-3.323599	Kwa Luka Msaki	
44885	non functional	0.0	2013-08-03	Government Of Tanzania	540	Government	38.044070	-4.272218	Kwa	
60739	functional	10.0	2013-05-03	Germany Republi	1210	CES	37.169807	-3.253847	Area Three Namba 27	
27263	functional	4700.0	2011-05-07	Cefa-njombe	1212	Cefa	35.249991	-9.070629	Kwa Yahona Kuvala	

27804 rows × 40 columns

In [866]:

```python
# Convert categorical columns to numeric
train_df['construction_year'] = pd.to_numeric(train_df['construction_year'], errors='coer
ce')

# Calculate median of non-zero values in 'construction_year'
median = train_df.loc[train_df['construction_year'] != 0, 'construction_year'].median()

# Replace 0s in 'construction_year' with the calculated median
train_df['construction_year'].replace(0, median, inplace=True)


# Convert 'date_recorded' to year only
train_df['date_recorded'] = pd.to_datetime(train_df['date_recorded']).dt.year

# Calculate 'Age' based on 'construction_year' and 'date_recorded'
train_df['Age'] = train_df['date_recorded'] - train_df['construction_year']

# Display the DataFrame with updated columns
train_df
```

Out[866]:

	status_group	amount_tsh	date_recorded	funder	gps_height	installer	longitude	latitude	wpt_name	nu
id										
69572	functional	6000.0	2011	Roman	1390	Roman	34.938093	-9.856322	none	
34310	functional	25.0	2013	Lottery Club	686	World vision	37.460664	-3.821329	Kwa Mahundi	
9944	functional	20.0	2011	Mkinga Distric Coun	0	DWE	39.172796	-4.765587	Tajiri	
50495	functional	0.0	2013	Lawatefuka Water Supply	1368	Lawatefuka water sup	37.092574	-3.181783	Kwa John Izack Mmari	
53752	functional	0.0	2012	Biore	0	WEDECO	34.364073	-3.629333	Mwabasabi	
...	...	...	...	...	...	...	...	...	...	
67885	non functional	0.0	2011	Mkinga Distric Coun	0	DWE	38.835001	-4.880204	Mijohoroni	
47002	non functional	6.0	2013	Ces(gmbh)	1383	DWE	37.454759	-3.323599	Kwa Luka Msaki	
44885	non functional	0.0	2013	Government Of Tanzania	540	Government	38.044070	-4.272218	Kwa	
60739	functional	10.0	2013	Germany Republi	1210	CES	37.169807	-3.253847	Area Three Namba 27	
27263	functional	4700.0	2011	Cefa-njombe	1212	Cefa	35.249991	-9.070629	Kwa Yahona Kuvala	

27804 rows × 41 columns

In [867]:

```python
train_df[['construction_year', 'Age', 'date_recorded']]
```

Out[867]:

	construction_year	Age	date_recorded
id			
69572	1999	12	2011
34310	2009	4	2013

id	construction_year	Age	date_recorded
9944	2009	2	2011
50495	2009	4	2013
53752	2000	12	2012
...	...	...	...
67885	1992	19	2011
47002	2008	5	2013
44885	1967	46	2013
60739	1999	14	2013
27263	1996	15	2011

27804 rows × 3 columns

In [868]:

```
train_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 27804 entries, 69572 to 27263
Data columns (total 41 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   status_group           27804 non-null  object 
 1   amount_tsh             27804 non-null  float64
 2   date_recorded          27804 non-null  int64  
 3   funder                 27804 non-null  object 
 4   gps_height             27804 non-null  int64  
 5   installer              27804 non-null  object 
 6   longitude              27804 non-null  float64
 7   latitude               27804 non-null  float64
 8   wpt_name               27804 non-null  object 
 9   num_private            27804 non-null  int64  
 10  basin                  27804 non-null  object 
 11  subvillage             27804 non-null  object 
 12  region                 27804 non-null  object 
 13  region_code            27804 non-null  int64  
 14  district_code          27804 non-null  int64  
 15  lga                    27804 non-null  object 
 16  ward                   27804 non-null  object 
 17  population             27804 non-null  int64  
 18  public_meeting         27804 non-null  object 
 19  recorded_by            27804 non-null  object 
 20  scheme_management      27804 non-null  object 
 21  scheme_name            27804 non-null  object 
 22  permit                 27804 non-null  object 
 23  construction_year      27804 non-null  int64  
 24  extraction_type        27804 non-null  object 
 25  extraction_type_group  27804 non-null  object 
 26  extraction_type_class  27804 non-null  object 
 27  management             27804 non-null  object 
 28  management_group       27804 non-null  object 
 29  payment                27804 non-null  object 
 30  payment_type           27804 non-null  object 
 31  water_quality          27804 non-null  object 
 32  quality_group          27804 non-null  object 
 33  quantity               27804 non-null  object 
 34  quantity_group         27804 non-null  object 
 35  source                 27804 non-null  object 
 36  source_type            27804 non-null  object 
 37  source_class           27804 non-null  object 
 38  waterpoint_type        27804 non-null  object 
 39  waterpoint_type_group  27804 non-null  object 
 40  Age                    27804 non-null  int64  
dtypes: float64(3), int64(8), object(30)
memory usage: 8.9+ MB
```

In [869]:

```
#changing amount_tsh to integer since the 0s in the float are redundant
train_df['amount_tsh'] = train_df['amount_tsh'].astype('int64')
train_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 27804 entries, 69572 to 27263
Data columns (total 41 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   status_group           27804 non-null  object 
 1   amount_tsh             27804 non-null  int64  
 2   date_recorded          27804 non-null  int64  
 3   funder                 27804 non-null  object 
 4   gps_height             27804 non-null  int64  
 5   installer              27804 non-null  object 
 6   longitude              27804 non-null  float64
 7   latitude               27804 non-null  float64
 8   wpt_name               27804 non-null  object 
 9   num_private            27804 non-null  int64  
 10  basin                  27804 non-null  object 
 11  subvillage             27804 non-null  object 
 12  region                 27804 non-null  object 
 13  region_code            27804 non-null  int64  
 14  district_code          27804 non-null  int64  
 15  lga                    27804 non-null  object 
 16  ward                   27804 non-null  object 
 17  population             27804 non-null  int64  
 18  public_meeting         27804 non-null  object 
 19  recorded_by            27804 non-null  object 
 20  scheme_management      27804 non-null  object 
 21  scheme_name            27804 non-null  object 
 22  permit                 27804 non-null  object 
 23  construction_year      27804 non-null  int64  
 24  extraction_type        27804 non-null  object 
 25  extraction_type_group  27804 non-null  object 
 26  extraction_type_class  27804 non-null  object 
 27  management             27804 non-null  object 
 28  management_group       27804 non-null  object 
 29  payment                27804 non-null  object 
 30  payment_type           27804 non-null  object 
 31  water_quality          27804 non-null  object 
 32  quality_group          27804 non-null  object 
 33  quantity               27804 non-null  object 
 34  quantity_group         27804 non-null  object 
 35  source                 27804 non-null  object 
 36  source_type            27804 non-null  object 
 37  source_class           27804 non-null  object 
 38  waterpoint_type        27804 non-null  object 
 39  waterpoint_type_group  27804 non-null  object 
 40  Age                    27804 non-null  int64  
dtypes: float64(2), int64(9), object(30)
memory usage: 8.9+ MB
```

Checking different columns which appear to have simmilar entries before determining which ones to drop

In [870]:

```
value_counts_combined = train_df[['quality_group', 'water_quality']].apply(lambda x: x.v
alue_counts())
value_counts_combined
```

Out[870]:

	quality_group	water_quality
colored	123.0	NaN
coloured	NaN	123.0
fluoride	125.0	124.0

	quality_group	water_quality
fluoride abandoned	NaN	1.0
good	25950.0	NaN
milky	39.0	39.0
salty	1182.0	1122.0
salty abandoned	NaN	60.0
soft	NaN	25950.0
unknown	385.0	385.0

In [871]:

```
value_counts_combined = train_df[['payment','payment_type' ]].apply(lambda x: x.value_counts())
value_counts_combined
```

Out[871]:

	payment	payment_type
annually	NaN	2266.0
monthly	NaN	5894.0
never pay	10034.0	10034.0
on failure	NaN	1210.0
other	205.0	205.0
pay annually	2266.0	NaN
pay monthly	5894.0	NaN
pay per bucket	6132.0	NaN
pay when scheme fails	1210.0	NaN
per bucket	NaN	6132.0
unknown	2063.0	2063.0

In [872]:

```
value_counts_combined = train_df[['quantity','quantity_group' ]].apply(lambda x: x.value_counts())
value_counts_combined
```

Out[872]:

	quantity	quantity_group
enough	16862	16862
insufficient	7060	7060
dry	2873	2873
seasonal	886	886
unknown	123	123

In [873]:

```
value_counts_combined = train_df[['waterpoint_type','waterpoint_type_group' ]].apply(lambda x: x.value_counts())
value_counts_combined
```

Out[873]:

	waterpoint_type	waterpoint_type_group
cattle trough	64	64.0

	waterpoint_type	waterpoint_type_group
communal standpipe	20303	25070.0
communal standpipe multiple	4767	NaN
dam	5	5.0
hand pump	1395	1395.0
improved spring	76	76.0
other	1194	1194.0

In [874]:

```
value_counts_combined = train_df[['source', 'source_type',
       'source_class' ]].apply(lambda x: x.value_counts())
value_counts_combined
```

Out[874]:

	source	source_type	source_class
borehole	NaN	4572.0	NaN
dam	458.0	458.0	NaN
groundwater	NaN	NaN	18782.0
hand dtw	117.0	NaN	NaN
lake	542.0	NaN	NaN
machine dbh	4455.0	NaN	NaN
other	139.0	155.0	NaN
rainwater harvesting	291.0	291.0	NaN
river	7576.0	NaN	NaN
river/lake	NaN	8118.0	NaN
shallow well	1089.0	1089.0	NaN
spring	13121.0	13121.0	NaN
surface	NaN	NaN	8867.0
unknown	16.0	NaN	155.0

In [875]:

```
value_counts_combined = train_df[['extraction_type',
       'extraction_type_group', 'extraction_type_class']].apply(lambda x: x.value_count
s())
value_counts_combined
```

Out[875]:

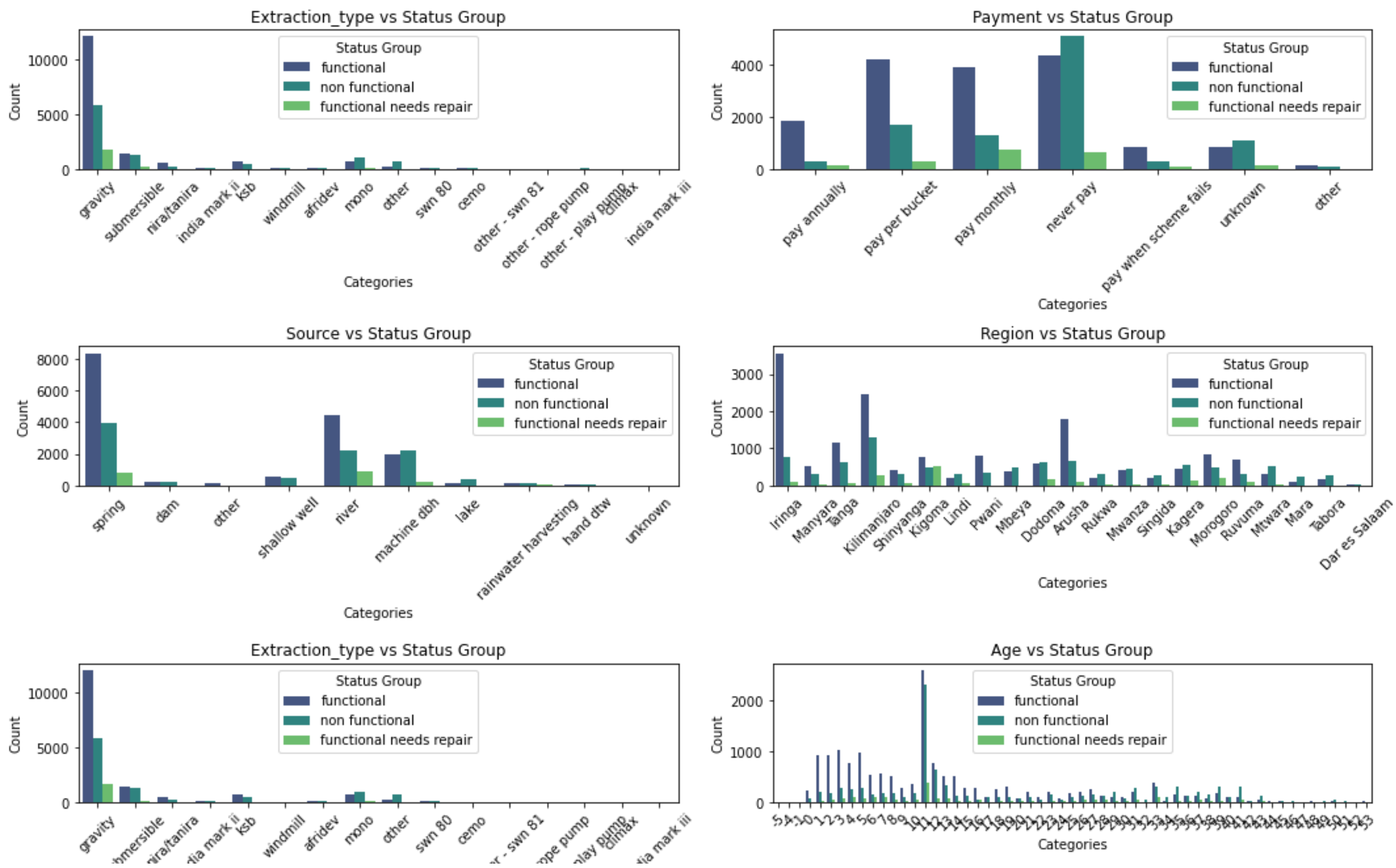
	extraction_type	extraction_type_group	extraction_type_class
afridev	198.0	198.0	NaN
cemo	89.0	NaN	NaN
climax	29.0	NaN	NaN
gravity	19610.0	19610.0	19610.0
handpump	NaN	NaN	1362.0
india mark ii	192.0	192.0	NaN
india mark iii	2.0	2.0	NaN
ksb	1159.0	NaN	NaN
mono	1777.0	1777.0	NaN
motorpump	NaN	NaN	1895.0
nira/tanira	747.0	747.0	NaN

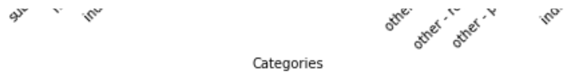
	extraction_type	extraction_type_group	extraction_type_class
other	844.0	844.0	844.0
other - play pump	66.0	NaN	NaN
other - rope pump	27.0	NaN	NaN
other - swn 81	10.0	NaN	NaN
other handpump	NaN	76.0	NaN
other motorpump	NaN	118.0	NaN
rope pump	NaN	27.0	27.0
submersible	2837.0	3996.0	3996.0
swn 80	147.0	147.0	NaN
wind-powered	NaN	70.0	70.0
windmill	70.0	NaN	NaN

Visualization in respect to the target variable that we are trying to inspect

In [876]:

```
columns_to_compare = ['extraction_type', 'payment','source' , 'region', 'extraction_type'
, 'Age']

# Create subplots for count plots
fig, axes = plt.subplots(nrows=3, ncols=2, figsize=(15, 10))

# Plotting count plots for each column against 'status_group' on different axes
for col, ax in zip(columns_to_compare, axes.flatten()):
    sns.countplot(data=train_df, x=col, hue='status_group', palette='viridis', ax=ax)
    ax.set_xlabel('Categories')
    ax.set_ylabel('Count')
    ax.set_title(f'{col.capitalize()} vs Status Group')
    ax.legend(title='Status Group')
    ax.tick_params(axis='x', rotation=45)

# Adjust layout
plt.tight_layout()
plt.show()
```





Interpretation

From above analysis the following features can be said about the functional water pumps:-

1. **Pumps that require frequent payments to use are more functional**
2. **Urban ares and cities have the highest number of functional pumps**
3. **Pump that run by gravity hence need the least maintainance are the most functional**
4. **Most of the functional pumps are below 15 yrs old**
5. **Soft water pumps are better**
6. **Pumps with underground sources perform better such as springs and shallow wells**

After inspection and analysis of several columns we drop the ones which are least expected to have an effect on the training model

In [877]:

```
cols_to_drop=['quantity_group','payment_type','quality_group','source_type','waterpoint_t
ype_group', 'source_type',
        'source_class',
        'extraction_type_group','num_private','subvillage','lga','longitude','latitude',
'ward']
train_df=train_df.drop(columns=cols_to_drop)
train_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 27804 entries, 69572 to 27263
Data columns (total 28 columns):
 #   Column                 Non-Null Count  Dtype 
---  ------                 --------------  ----- 
 0   status_group           27804 non-null  object
 1   amount_tsh             27804 non-null  int64 
 2   date_recorded          27804 non-null  int64 
 3   funder                 27804 non-null  object
 4   gps_height             27804 non-null  int64 
 5   installer              27804 non-null  object
 6   wpt_name               27804 non-null  object
 7   basin                  27804 non-null  object
 8   region                 27804 non-null  object
 9   region_code            27804 non-null  int64 
 10  district_code          27804 non-null  int64 
 11  population             27804 non-null  int64 
 12  public_meeting         27804 non-null  object
 13  recorded_by            27804 non-null  object
 14  scheme_management      27804 non-null  object
 15  scheme_name            27804 non-null  object
 16  permit                 27804 non-null  object
 17  construction_year      27804 non-null  int64 
 18  extraction_type        27804 non-null  object
 19  extraction_type_class  27804 non-null  object
 20  management             27804 non-null  object
 21  management_group       27804 non-null  object
 22  payment                27804 non-null  object
 23  water_quality          27804 non-null  object
 24  quantity               27804 non-null  object
 25  source                 27804 non-null  object
 26  waterpoint_type        27804 non-null  object
 27  Age                    27804 non-null  int64 
dtypes: int64(8), object(20)
memory usage: 7.4+ MB
```

In [878]:

```
train_df.columns
```

Out[878]:

```
Index(['status_group', 'amount_tsh', 'date_recorded', 'funder', 'gps_height',
       'installer', 'wpt_name', 'basin', 'region', 'region_code',
       'district_code', 'population', 'public_meeting', 'recorded_by',
       'scheme_management', 'scheme_name', 'permit', 'construction_year',
       'extraction_type', 'extraction_type_class', 'management',
       'management_group', 'payment', 'water_quality', 'quantity', 'source',
       'waterpoint_type', 'Age'],
      dtype='object')
```

Data Preprocessing

In this section i begin with splitting the data to training and test set. Given that we do have categorical data i use One hot encoder to transform the data and because previosly we saw the data given in the status group was leaning more toward the functional status i chose to use SMOTE and try rectify the imbalance.

1. **Splitting the data**

In [879]:

```
# Define X and y
y = train_df["status_group"]
X = train_df.drop(["status_group", "recorded_by", "construction_year", "funder", "install
er", "wpt_name", "basin",
                   "region", "scheme_management", "management", "scheme_name", "extracti
on_type",
                   "extraction_type_class", "management_group", "payment", "water_qualit
y", "quantity", "source",
                   "waterpoint_type"], axis=1)

# Transform the target variable into integers
label = LabelEncoder()
y_transformed = label.fit_transform(y)

# Perform train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y_transformed, test_size=0.3, str
atify=y_transformed,
                                                     random_state=42)

# Define preprocessing steps
numerical_cols = X.select_dtypes(include=["int64"]).columns.tolist()
categorical_cols = X.select_dtypes(include=["category"]).columns.tolist()

numeric_transformer = StandardScaler()
categorical_transformer = OneHotEncoder(handle_unknown='ignore')


# consolidate onehotencoder and standardscaler
preprocessor = ColumnTransformer(
    transformers=[
        ('num', numeric_transformer, numerical_cols),
        ('cat', categorical_transformer, categorical_cols)
    ])

# Apply the transformations to original_df to get the transformed DataFrame
transformed_data = preprocessor.fit_transform(train_df)
new_df = pd.DataFrame(transformed_data, columns=numerical_cols + categorical_cols)
```

In [880]:

```
new_df.columns
```

Out[880]:

```
Index(['amount_tsh', 'date_recorded', 'gps_height', 'region_code',
       'district_code', 'population', 'Age'],
      dtype='object')
```

2.Creating a base model

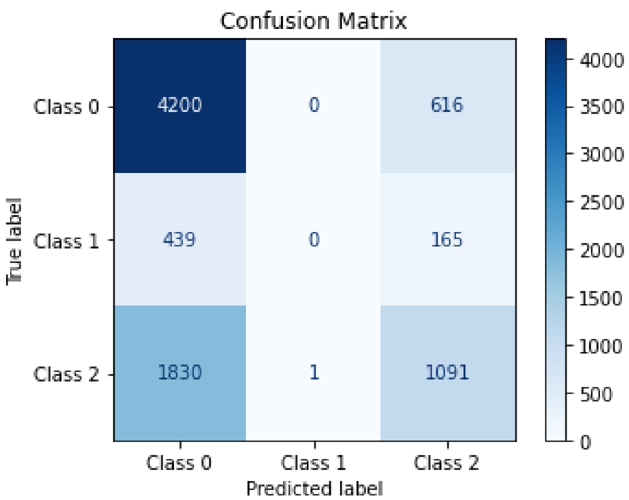
2.Creating a base model

In [881]:

```
base_model = LogisticRegression(random_state=42, max_iter=1000)
base_model.fit(X_train, y_train)

# Make predictions on the test set
y_pred = base_model.predict(X_test)

# evaluate
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='weighted')
recall = recall_score(y_test, y_pred, average='weighted')

print(f'Model: {base_model}')
print(f'Accuracy: {accuracy}')
print(f'Precision: {precision}')
print(f'Recall: {recall}')

# Plot confusion matrix
plot_confusion_matrix(base_model, X_test, y_test, cmap=plt.cm.Blues, display_labels=["Cl
ass 0", "Class 1", "Class 2"])
plt.title('Confusion Matrix')
plt.show()
```

```
/Users/esthernyawera/anaconda3/envs/learn-env/lib/python3.8/site-packages/sklearn/linear_
model/_logistic.py:762: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

```
Model: LogisticRegression(max_iter=1000, random_state=42)
Accuracy: 0.6342603692160154
Precision: 0.5789652847510164
Recall: 0.6342603692160154
```



MODELLING

Here i am going to build a pipeline with the preprocessed data and train it using the following classifiers till the on with better metrics is observed:-

1. **Decision tree classifier**
2. **KNN classifier**
3. **Random forest with different parameters**

In [882]:

```
# Define the pipeline with preprocessing and model
pipeline = ImbPipeline(steps=[
    ('preprocessor', preprocessor),
    ('smote', SMOTE(random_state=42)),
    ('classifier',DecisionTreeClassifier(
    criterion='entropy',
    max_depth=5,
    min_samples_split=2,
    min_samples_leaf=1,
    max_features=None
) )
])

# Fit the pipeline (preprocessing + model) on training data
pipeline.fit(X_train, y_train)

# Predict on the test set
y_pred = pipeline.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='weighted')
recall = recall_score(y_test, y_pred, average='weighted')
f1 = f1_score(y_test, y_pred, average='weighted')

# Display evaluation metrics for each model
print(f'classifier: {DecisionTreeClassifier}')
print(f'Accuracy: {accuracy}')
print(f'Precision: {precision}')
print(f'Recall: {recall}')
```

```
classifier: <class 'sklearn.tree._classes.DecisionTreeClassifier'>
Accuracy: 0.4889714696715416
Precision: 0.6215226784660483
Recall: 0.4889714696715416
```

In [883]:

```
# Define the pipeline with preprocessing and model
pipeline = ImbPipeline(steps=[
    ('preprocessor', preprocessor),
    ('smote', SMOTE(random_state=42)),
    ('classifier', KNeighborsClassifier(n_neighbors=5, metric='euclidean'))
])

# Fit the pipeline (preprocessing + model) on training data
pipeline.fit(X_train, y_train)

# Predict on the test set
y_pred = pipeline.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='weighted')
recall = recall_score(y_test, y_pred, average='weighted')
f1 = f1_score(y_test, y_pred, average='weighted')

# Display evaluation metrics for each model
print(f'classifier: {knn}')
print(f'Accuracy: {accuracy}')
print(f'Precision: {precision}')
print(f'Recall: {recall}')
```

```
classifier: KNeighborsClassifier(metric='euclidean')
Accuracy: 0.6364181251498442
Precision: 0.6920359372631527
Recall: 0.6364181251498442
```

In [884]:

```
# Define the pipeline with preprocessing and model
pipeline = ImbPipeline(steps=[
```

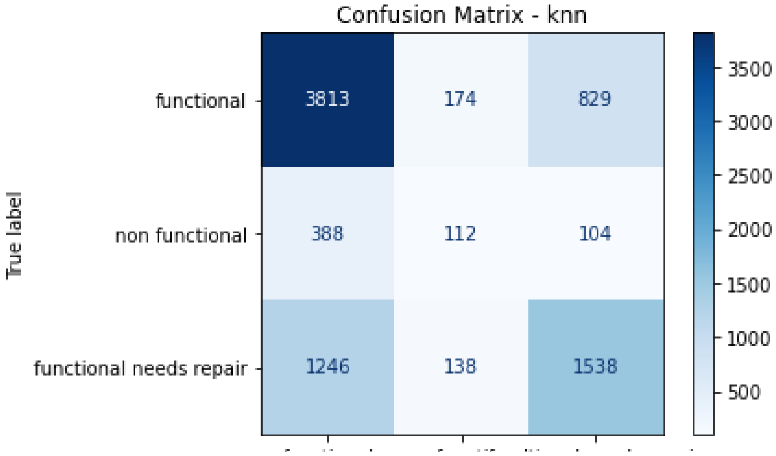
```
    ('preprocessor', preprocessor),
    ('smote', SMOTE(random_state=42)),
    ('classifier', RandomForestClassifier(
    criterion= 'gini',
    max_depth=10,
    min_samples_split=2,
    min_samples_leaf=1,
    bootstrap=True,
    random_state=42  ))
])

# Fit the pipeline (preprocessing + model) on training data
pipeline.fit(X_train, y_train)

# Predict on the test set
y_pred = pipeline.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='weighted')
recall = recall_score(y_test, y_pred, average='weighted')
f1 = f1_score(y_test, y_pred, average='weighted')

# Display evaluation metrics for each model
print(f'classifier: {RandomForestClassifier}')
print(f'Accuracy: {accuracy}')
print(f'Precision: {precision}')
print(f'Recall: {recall}')
```

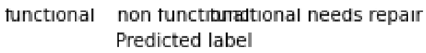
```
classifier: <class 'sklearn.ensemble._forest.RandomForestClassifier'>
Accuracy: 0.6520019180052745
Precision: 0.7198692196672536
Recall: 0.6520019180052745
```

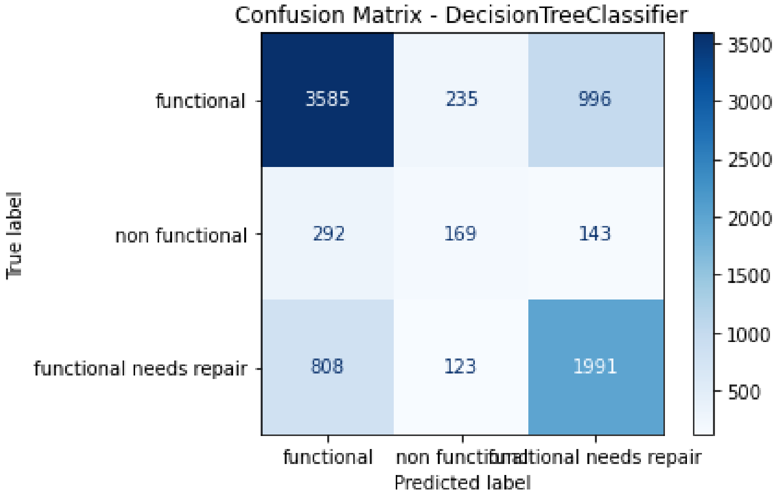
PLOTTING CONFUSION MATRIX

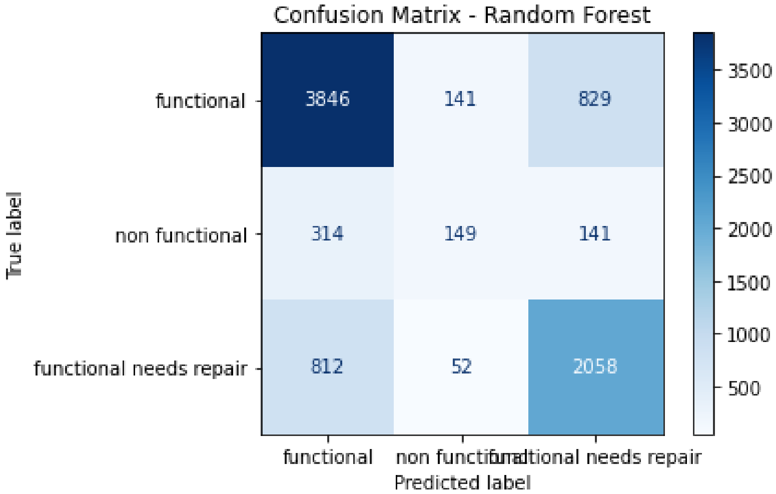
In [885]:

```
# Define models and their names
models = [KNeighborsClassifier(), DecisionTreeClassifier(), RandomForestClassifier()]
model_names = ['knn', 'DecisionTreeClassifier', 'Random Forest']

# Fit models, make predictions, and plot confusion matrix for each model
for model, name in zip(models, model_names):
    model.fit(X_train, y_train)  # Fit the model with training data

    predictions = model.predict(X_test)  # Make predictions on the test data

    # Plot confusion matrix for each model
    disp = plot_confusion_matrix(model, X_test, y_test,
                                 display_labels=["functional", "non functional", "functi
onal needs repair"],
                                 cmap=plt.cm.Blues)
    disp.ax_.set_title(f"Confusion Matrix - {name}")
    plt.show()
```









PICKLE BEST CLASSIFIER

Random forest classifier gives us the best metrics therefore at this point it is advisable to save and store it for future retrival when making predictions on sample data.

In [886]:

```
# Save the trained model to a file using pickle
with open('random_forest_model.pkl', 'wb') as file:
    pickle.dump(RandomForestClassifier, file)
```

INTERPRETATION

1. **The Random Forest Classifier outperforms the other models, exhibiting the highest accuracy, precision, and recall rates.**

   #### RandomForestClassifier

Accuracy: 65%

Precision: 72%

Recall: 65%

CONCLUSION

- **Most of the functional wells are found in the city where people mostly pay to use them. More wells similar to those in the city should be built in other region with focus on pumps that will work on gravity.**
- **The wells that were recorded a long time ago most are non functional and in need of repairs this is to show they have been neglected.**

they have been neglected.

- **Wells that run on soft water sources the most functional.**

RECOMMENDATION

- **Build more wells resembling those in urban areas in other regions that experience water shortage, especially focusing on implementing pumps that operate using gravity and have soft water sources. This replication strategy might enhance the functionality of wells in other areas**
- **Increase attention and maintenance of older wells to prevent deterioration and improve their functionality.**

NEXT STEPS

- **Further exploration into wider scopes of data and analysis of different features on how they affect functionality**
- **Develop a strong strategy on well maintenance and repairs**
- **Implement recommendations**