Self Case Study-1_Pump it up-Data Mining the Water Table

Section-II

4. Feature Engineering and Baseline model

4.1 Importing required libraries & reading data

In [178]:
```
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import re
import time
import warnings
warnings.filterwarnings("ignore")
import numpy as np
from nltk.corpus import stopwords
from sklearn.preprocessing import normalize
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.manifold import TSNE
import seaborn as sns
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix
import sys
from category_encoders import OneHotEncoder
from sklearn.preprocessing import MinMaxScaler
from category_encoders import TargetEncoder, LeaveOneOutEncoder, WOEEncoder
from sklearn.preprocessing import RobustScaler
from sklearn.metrics import accuracy_score, balanced_accuracy_score
from sklearn.metrics import confusion_matrix
```

In [132]:
```
df1 = pd.read_csv('clean_df.csv')
```

In [133]:
```
df1.head()
```

Out[133]:

	Unnamed: 0	id	amount_tsh	funder	gps_height	installer	longitude	latitude	basin
0	0	69572	6000.0	other	1390	other	34.938093	-9.856322	Lake Nyasa
1	1	8776	0.0	other	1399	other	34.698766	-2.147466	Lake Victoria
2	2	34310	25.0	other	686	World Vision	37.460664	-3.821329	Pangani M
3	3	67743	0.0	Unicef	263	UNICEF	38.486161	-11.155298	Ruvuma / Southern Coast
4	4	19728	0.0	other	0	other	31.130847	-1.825359	Lake Victoria

5 rows × 24 columns

In [134]:
```
df1.drop(columns=['Unnamed: 0'],inplace=True )
```

In [135]:
```
df1['permit'] = df1['permit'].astype(bool).astype(int) #converting True/Flse into
df1['public_meeting'] = df1['public_meeting'].astype(bool).astype(int) #convertin
```

4.2 Converting categories of target variable into numeric values

In [136]:
```
df1['status_group'].unique()
```

Out[136]:
```
array(['functional', 'non functional', 'functional needs repair'],
      dtype=object)
```

In [137]:
```
numeric_target_values = {'functional':1, 'non functional':0, 'functional needs re
```

In [138]:
```
df1['status_group'] = df1['status_group'].replace(numeric_target_values)
```

In [139]:
```
df1['status_group'].value_counts()
```

Out[139]:
```
1    32259
0    22824
2     4317
Name: status_group, dtype: int64
```

4.3 Analysis of column 'amount_tsh' to conclude if the same adds any values

In [140]:
```
df1['amount_tsh'].value_counts()
```

Out[140]:
```
0.0         41639
500.0        3102
50.0         2472
1000.0       1488
20.0         1463
            ...
6300.0          1
120000.0        1
138000.0        1
350000.0        1
59.0            1
Name: amount_tsh, Length: 98, dtype: int64
```

In [141]:
```
df1['amount_tsh'].value_counts()/df1['amount_tsh'].value_counts().sum()
```

Out[141]:
```
0.0         0.700993
500.0       0.052222
50.0        0.041616
1000.0      0.025051
20.0        0.024630
              ...
6300.0      0.000017
120000.0    0.000017
138000.0    0.000017
350000.0    0.000017
59.0        0.000017
Name: amount_tsh, Length: 98, dtype: float64
```

In [142]:
```
df1.loc[df1['amount_tsh']==0].groupby('status_group').size()
```

Out[142]:
```
status_group
0    18885
1    19706
2     3048
dtype: int64
```

Here we can see that around 70% of all values of 'amount_tsh' feature has zero. At this point, it is very difficult to say if this feature will make any impact as it has very less information. we will try modelling with & without 'amount_tsh' feature.

In [143]:
```
df1.loc[df1['amount_tsh']!=0, 'amount_tsh'].mean()
```

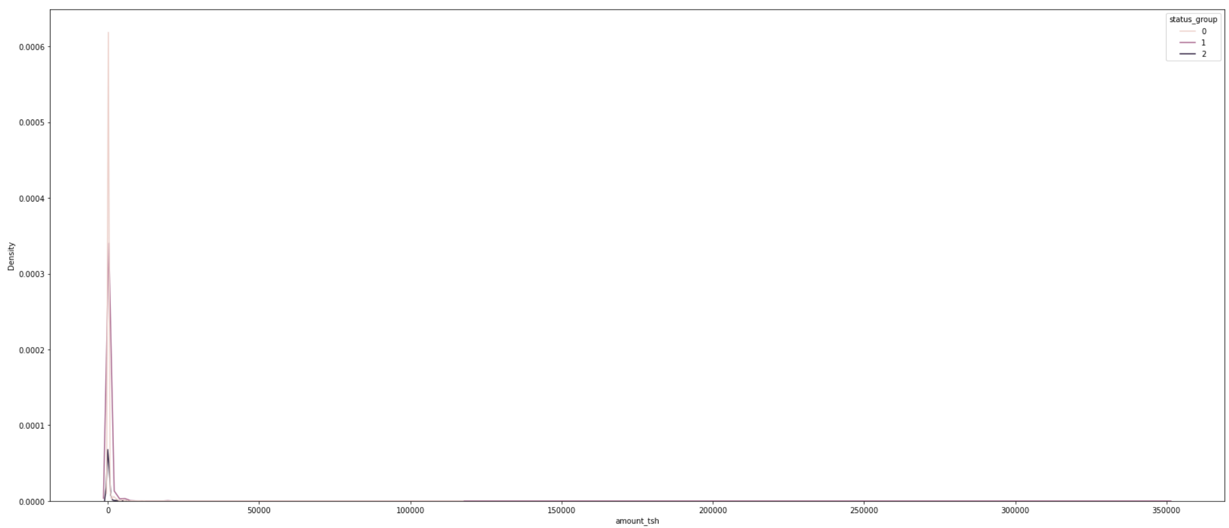
Out[143]:
```
1062.351942458195
```

In [144]:
```
df1.loc[df1['amount_tsh']!=0, 'amount_tsh'].describe()
```

Out[144]:
```
count     17761.000000
mean       1062.351942
std        5409.344940
min           0.200000
25%          50.000000
50%         250.000000
75%        1000.000000
max      350000.000000
Name: amount_tsh, dtype: float64
```

In [145]:
```
plt.figure(figsize=(28,12))
sns.kdeplot(data=df1, x='amount_tsh', hue="status_group", gridsize=200)
```

Out[145]: <AxesSubplot:xlabel='amount_tsh', ylabel='Density'>



we will try replacing the zero values with mean value

In [146]:
```
df2 = df1.copy() #creating copy of dataset
```

In [147]:
```
df2.shape
```

Out[147]: (59400, 23)

In [148]:
```
df2['amount_tsh'].describe()
```

Out[148]:
```
count     59400.000000
mean        317.650385
std        2997.574558
min           0.000000
25%           0.000000
50%           0.000000
75%          20.000000
max      350000.000000
Name: amount_tsh, dtype: float64
```

In [ ]:

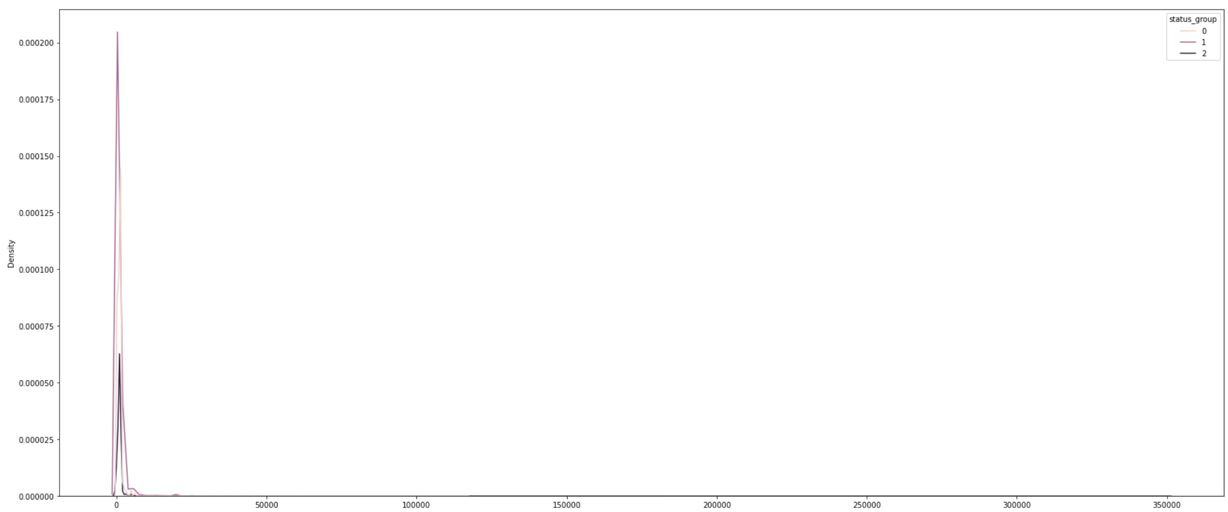
In [149]:
```
df2['amount_tsh'].replace(to_replace = 0 , value =1062.35, inplace=True)
```

In [150]:
```
df2['amount_tsh'].describe()  #checking after replacement of zero with mean
```

Out[150]:
```
count     59400.000000
mean       1062.350581
std        2957.853024
min           0.200000
25%        1062.350000
50%        1062.350000
75%        1062.350000
max      350000.000000
Name: amount_tsh, dtype: float64
```

In [151]:
```
plt.figure(figsize=(28,12))
sns.kdeplot(data=df2, x='amount_tsh', hue="status_group", gridsize=200)
```

Out[151]: <AxesSubplot:xlabel='amount_tsh', ylabel='Density'>



4.4 Modeling with 'amount_tsh'

4.4.1 Generating list of numerical and categorical values

In [152]:
```
df2.columns
```

Out[152]:
```
Index(['id', 'amount_tsh', 'funder', 'gps_height', 'installer', 'longitude',
       'latitude', 'basin', 'region', 'district_code', 'lga', 'population',
       'public_meeting', 'permit', 'extraction_type_group', 'management',
       'payment', 'water_quality', 'quantity', 'source', 'status_group',
       'operational_year', 'waterpoint_type'],
      dtype='object')
```

In [153]:
```
numerical_features = ['amount_tsh','gps_height','longitude', 'latitude', 'distric
```

In [154]:
```
categorical_features = ['funder','installer','basin', 'region', 'lga','public_mee
       'management', 'payment', 'water_quality', 'quantity', 'source','waterpoint
```

In [155]:
```
len(categorical_features)
```

Out[155]: 14

In [156]:
```
len(numerical_features)
```

Out[156]: 7

In [158]:
```
df2.head(1)
```

Out[158]:

	district_code	...	permit	extraction_type_group	management	payment	water_quality	quantity	so
	5	...	0	gravity	vwc	pay annually	soft	enough	s

4.4.2 Separating target column 'status_group' from other columns

In [159]:
```
y=df2['status_group']
X = df2.drop(columns = ['status_group', 'id'])
```

In [160]:
```
X.shape
```

Out[160]: (59400, 21)

In [161]:
```
y.shape
```

Out[161]: (59400,)

4.4.3 Splitting the data

In [162]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_s
```

4.4.4 Standardising the numerical features using RobustScaler

In [163]:
```
transformer = RobustScaler().fit(X_train[numerical_features])
```

In [164]:
```
transformer.transform(X_train[numerical_features])
```

Out[164]:
```
array([[ 0.00000000e+00, -2.79210926e-01, -8.96727076e-01, ...,
         3.33333333e-01,  0.00000000e+00,  0.00000000e+00],
       [ 0.00000000e+00, -2.79210926e-01, -6.00695420e-01, ...,
         0.00000000e+00,  0.00000000e+00,  2.22222222e-01],
       [ 0.00000000e+00, -2.64795144e-01,  1.11542716e+00, ...,
        -3.33333333e-01, -4.47779753e-01, -2.22222222e-01],
       ...,
       [ 0.00000000e+00, -2.79210926e-01, -1.07130384e+00, ...,
         9.00000000e+00,  0.00000000e+00,  0.00000000e+00],
       [ 0.00000000e+00, -2.79210926e-01, -2.13696508e-01, ...,
        -6.66666667e-01,  0.00000000e+00,  1.11111111e-01],
       [-1.05735000e+03,  5.72837633e-01,  6.02806549e-01, ...,
         6.66666667e-01, -1.27611012e+00, -1.11111111e-01]])
```

In [165]:
```
transformer.transform(X_test[numerical_features])
```

Out[165]:
```
array([[ 0.00000000e+00, -2.79210926e-01, -8.56100409e-01, ...,
         0.00000000e+00,  0.00000000e+00,  0.00000000e+00],
       [ 0.00000000e+00,  5.52352049e-01, -1.06803837e-01, ...,
        -6.66666667e-01, -5.25090588e-01, -8.88888889e-01],
       [ 0.00000000e+00, -2.79210926e-01, -2.70020041e-01, ...,
         3.33333333e-01,  0.00000000e+00,  0.00000000e+00],
       ...,
       [-5.62350000e+02,  9.97723824e-01, -1.59183129e-01, ...,
         0.00000000e+00,  0.00000000e+00,  1.11111111e-01],
       [-5.62350000e+02, -6.37329287e-02,  4.63123027e-01, ...,
         3.33333333e-01, -1.23745471e+00, -8.88888889e-01],
       [ 0.00000000e+00, -2.79210926e-01, -3.37846948e-01, ...,
        -3.33333333e-01,  0.00000000e+00,  1.11111111e-01]])
```

In [166]:
```
X_train[numerical_features] = transformer.transform(X_train[numerical_features])
```

In [167]:
```
X_test[numerical_features] = transformer.transform(X_test[numerical_features])
```

In [168]:
```
X_train.head()
```

Out[168]:

	amount_tsh	funder	gps_height	installer	longitude	latitude	basin	regio
51950	0.0	Hesawa	-0.279211	DWE	-0.896727	0.418023	Lake Tanganyika	Kager
33024	0.0	Rwssp	-0.279211	DWE	-0.600695	0.228799	Lake Victoria	Shinyang
33547	0.0	Undefined	-0.264795	Undefined	1.115427	-0.370306	Wami / Ruvu	Dar e Salaan
1813	0.0	other	-0.279211	other	-0.559189	-0.116281	Lake Tanganyika	Tabor
17664	0.0	Tcrs	0.756449	DWE	0.469989	0.453226	Internal	Arush

5 rows × 21 columns

In [169]:
```
X_test.head()
```

Out[169]:

	amount_tsh	funder	gps_height	installer	longitude	latitude	basin	region	dist
59379	0.00	other	-0.279211	other	-0.856100	0.613682	Lake Victoria	Kagera	
23853	0.00	other	0.552352	DWE	-0.106804	0.172762	Internal	Singida	-
4306	0.00	Undefined	-0.279211	Undefined	-0.270020	-0.830738	Lake Nyasa	Mbeya	
11714	0.00	Hesawa	0.775417	HESAWA	-0.101200	0.586696	Lake Victoria	Mara	-
36791	-62.35	Danida	0.485584	other	0.164856	-1.087846	Ruvuma / Southern Coast	Ruvuma	

5 rows × 21 columns

4.4.5 Encoding of categorical features

In [171]:
```
import category_encoders as ce
transformer_te = ce.TargetEncoder().fit(X_train[categorical_features], y_train)
```

In [172]:
```
X_train[categorical_features] = transformer_te.transform(X_train[categorical_feat
X_test[categorical_features] = transformer_te.transform(X_test[categorical_featur
```

In [173]:
```
X_train.head()
```

Out[173]:

	amount_tsh	funder	gps_height	installer	longitude	latitude	basin	region	distı
51950	0.0	0.628118	-0.279211	0.727951	-0.896727	0.418023	0.709981	0.693627	
33024	0.0	0.740807	-0.279211	0.727951	-0.600695	0.228799	0.686842	0.815691	
33547	0.0	0.752472	-0.264795	0.755051	1.115427	-0.370306	0.613160	0.571651	-
1813	0.0	0.715915	-0.279211	0.674986	-0.559189	-0.116281	0.709981	0.471591	
17664	0.0	0.522541	0.756449	0.727951	0.469989	0.453226	0.721076	0.791061	

5 rows × 21 columns

In [174]:
```
X_test.head()
```

Out[174]:

	amount_tsh	funder	gps_height	installer	longitude	latitude	basin	region	distı
59379	0.00	0.715915	-0.279211	0.674986	-0.856100	0.613682	0.686842	0.693627	
23853	0.00	0.715915	0.552352	0.727951	-0.106804	0.172762	0.721076	0.606509	-
4306	0.00	0.752472	-0.279211	0.755051	-0.270020	-0.830738	0.760453	0.720716	
11714	0.00	0.628118	0.775417	0.630941	-0.101200	0.586696	0.686842	0.506305	-
36791	-62.35	0.658088	0.485584	0.674986	0.164856	-1.087846	0.519412	0.686632	

5 rows × 21 columns

In [176]:
```
from sklearn.linear_model import LogisticRegression
clf_lr1 = LogisticRegression(class_weight = 'balanced', solver = 'lbfgs', random_
clf_lr1.fit(X_train, y_train)
```

Out[176]:
```
▸ LogisticRegression
```

In [199]:
```
# making prediction on train set
y_pred_train = clf_lr1.predict(X_train)

# making predictions on test set
y_pred_test = clf_lr1.predict(X_test)

# printing the result
print("Accuracy:")
print("*"*50)
print("TRAIN:", accuracy_score(y_train, y_pred_train))
print("TEST:", accuracy_score(y_test, y_pred_test))


print("\nBalanced Accuracy:")
print("*"*50)
print("TRAIN:", balanced_accuracy_score(y_train, y_pred_train))
print("TEST:", balanced_accuracy_score(y_test, y_pred_test))
```

```
Accuracy:
**************************************************
TRAIN: 0.4858585858585859
TEST: 0.49107744107744106

Balanced Accuracy:
**************************************************
TRAIN: 0.47277359029364857
TEST: 0.47372617664030797
```

4.5 Modeling without 'amount_tsh'

In [185]:
```
numerical_features1 = ['gps_height','longitude', 'latitude', 'district_code','pop
```

In [180]:
```
df3 = df1.copy() #creating copy of dataset
```

In [181]:
```
y1=df3['status_group']
X1 = df3.drop(columns = ['status_group', 'id','amount_tsh'])
```

In [182]:
```
from sklearn.model_selection import train_test_split
X_train1, X_test1, y_train1, y_test1 = train_test_split(X1, y1, test_size=0.2, ra
```

In [186]:
```
transformer1 = RobustScaler().fit(X_train1[numerical_features1])
```

In [188]:
```
X_train1[numerical_features1] = transformer1.transform(X_train1[numerical_feature
X_test1[numerical_features1] = transformer1.transform(X_test1[numerical_features1
```

In [189]:
```
import category_encoders as ce
transformer_te1 = ce.TargetEncoder().fit(X_train1[categorical_features], y_train1
```

In [190]:
```
X_train1[categorical_features] = transformer_te1.transform(X_train1[categorical_f
X_test1[categorical_features] = transformer_te1.transform(X_test1[categorical_fea
```

In [191]:
```
from sklearn.linear_model import LogisticRegression
clf_lr2 = LogisticRegression(class_weight = 'balanced', solver = 'lbfgs', random_
clf_lr2.fit(X_train1, y_train1)
```

Out[191]:
```
LogisticRegression
LogisticRegression(class_weight='balanced', random_state=30)
```

In [200]:
```
# making prediction on train set
y_pred_train1 = clf_lr2.predict(X_train1)

# making predictions on test set
y_pred_test1 = clf_lr2.predict(X_test1)

# printing the result
print("Accuracy:")
print("*"*50)
print("TRAIN:", accuracy_score(y_train1, y_pred_train1))
print("TEST:", accuracy_score(y_test1, y_pred_test1))


print("\nBalanced Accuracy:")
print("*"*50)
print("TRAIN:", balanced_accuracy_score(y_train1, y_pred_train1))
print("TEST:", balanced_accuracy_score(y_test1, y_pred_test1))
```

```
Accuracy:
**************************************************
TRAIN: 0.6254208754208754
TEST: 0.6262626262626263

Balanced Accuracy:
**************************************************
TRAIN: 0.607809164357186
TEST: 0.6043859306810534
```

From above result it is understood that accuracy has improved after removing the feature 'amount_tsh'. Hence we conclude that it is not adding any value in prediction and hence should be removed.

In [ ]: