In [1]:
```
pip install mlxtend
```

```
Requirement already satisfied: mlxtend in ./opt/anaconda3/envs/learn-env/
lib/python3.8/site-packages (0.18.0)
Requirement already satisfied: scipy>=1.2.1 in ./opt/anaconda3/envs/learn
-env/lib/python3.8/site-packages (from mlxtend) (1.5.2)
Requirement already satisfied: joblib>=0.13.2 in ./opt/anaconda3/envs/lea
rn-env/lib/python3.8/site-packages (from mlxtend) (0.17.0)
Requirement already satisfied: matplotlib>=3.0.0 in ./opt/anaconda3/envs/
learn-env/lib/python3.8/site-packages (from mlxtend) (3.3.1)
Requirement already satisfied: pandas>=0.24.2 in ./opt/anaconda3/envs/lea
rn-env/lib/python3.8/site-packages (from mlxtend) (1.1.3)
Requirement already satisfied: numpy>=1.16.2 in ./opt/anaconda3/envs/lear
n-env/lib/python3.8/site-packages (from mlxtend) (1.18.5)
Requirement already satisfied: setuptools in ./opt/anaconda3/envs/learn-e
nv/lib/python3.8/site-packages (from mlxtend) (50.3.0.post20201103)
Requirement already satisfied: scikit-learn>=0.20.3 in ./opt/anaconda3/en
vs/learn-env/lib/python3.8/site-packages (from mlxtend) (0.23.2)
Requirement already satisfied: pillow>=6.2.0 in ./opt/anaconda3/envs/lear
n-env/lib/python3.8/site-packages (from matplotlib>=3.0.0->mlxtend) (7.2.
0)
Requirement already satisfied: cycler>=0.10 in ./opt/anaconda3/envs/learn
-env/lib/python3.8/site-packages (from matplotlib>=3.0.0->mlxtend) (0.10.
0)
Requirement already satisfied: certifi>=2020.06.20 in ./opt/anaconda3/env
s/learn-env/lib/python3.8/site-packages (from matplotlib>=3.0.0->mlxtend)
(2020.12.5)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.3 i
n ./opt/anaconda3/envs/learn-env/lib/python3.8/site-packages (from matplo
tlib>=3.0.0->mlxtend) (2.4.7)
Requirement already satisfied: python-dateutil>=2.1 in ./opt/anaconda3/en
vs/learn-env/lib/python3.8/site-packages (from matplotlib>=3.0.0->mlxten
d) (2.8.1)
Requirement already satisfied: kiwisolver>=1.0.1 in ./opt/anaconda3/envs/
learn-env/lib/python3.8/site-packages (from matplotlib>=3.0.0->mlxtend)
(1.2.0)
Requirement already satisfied: pytz>=2017.2 in ./opt/anaconda3/envs/learn
-env/lib/python3.8/site-packages (from pandas>=0.24.2->mlxtend) (2020.1)
Requirement already satisfied: threadpoolctl>=2.0.0 in ./opt/anaconda3/en
vs/learn-env/lib/python3.8/site-packages (from scikit-learn>=0.20.3->mlxt
end) (2.1.0)
Requirement already satisfied: six in ./opt/anaconda3/envs/learn-env/lib/
python3.8/site-packages (from cycler>=0.10->matplotlib>=3.0.0->mlxtend)
(1.15.0)
Note: you may need to restart the kernel to use updated packages.
```

In [2]:
```
#Import relevant libraries here, more will be added later on, so we can see
import numpy as np
import pandas as pd
%matplotlib inline
import matplotlib.pyplot as plt
from sklearn.datasets import make_classification
from sklearn.metrics import accuracy_score, classification_report
from sklearn.linear_model import LogisticRegression
from mlxtend.plotting import plot_decision_regions
import warnings
from sklearn.metrics import recall_score
from sklearn.exceptions import DataConversionWarning
warnings.filterwarnings(action='ignore', category=DataConversionWarning)
```

OSEMN-Obtain

Here we will be using pandas to convert our data for readability

In [3]:
```
#Let's import and assign the data using pandas
sample = pd.read_csv('SubmissionFormat.csv')
df_test = pd.read_csv('702ddfc5-68cd-4d1d-a0de-f5f566f76d91.csv')
df_train_labels = pd.read_csv('0bf8bc6e-30d0-4c50-956a-603fc693d966.csv')
df_train_features = pd.read_csv('4910797b-ee55-40a7-8668-10efd5c1b960.csv')
```

In [4]:
```
#explore the data rows and column sizes
df_train_features.shape, df_test.shape, df_train_labels.shape
```

Out[4]:
```
((59400, 40), (14850, 40), (59400, 2))
```

In [5]:
```
df_test.sample()
```

Out[5]:

	id	amount_tsh	date_recorded	funder	gps_height	installer	longitude	latitude	wpt_na
5962	22134	0.0	2011-04-06	NaN	0	NaN	34.377662	-8.768062	Kwa M Mgo\

1 rows × 40 columns

In [6]:
```
df_test.head()
```

Out[6]:

	id	amount_tsh	date_recorded	funder	gps_height	installer	longitude	latitude	wp
0	50785	0.0	2013-02-04	Dmdd	1996	DMDD	35.290799	-4.059696	Se
1	51630	0.0	2013-02-04	Government Of Tanzania	1569	DWE	36.656709	-3.309214	k
2	17168	0.0	2013-02-01	NaN	1567	NaN	34.767863	-5.004344	Se
3	45559	0.0	2013-01-22	Finn Water	267	FINN WATER	38.058046	-9.418672	Kw
4	49871	500.0	2013-03-27	Bruder	1260	BRUDER	35.006123	-10.950412	Kw

5 rows × 40 columns

In [7]:
```
df_train_features.head()
```

Out[7]:

	id	amount_tsh	date_recorded	funder	gps_height	installer	longitude	latitude	wpt_na
0	69572	6000.0	2011-03-14	Roman	1390	Roman	34.938093	-9.856322	r
1	8776	0.0	2013-03-06	Grumeti	1399	GRUMETI	34.698766	-2.147466	Zaha
2	34310	25.0	2013-02-25	Lottery Club	686	World vision	37.460664	-3.821329	Mah
3	67743	0.0	2013-01-28	Unicef	263	UNICEF	38.486161	-11.155298	Zaha Nanyu
4	19728	0.0	2011-07-13	Action In A	0	Artisan	31.130847	-1.825359	Shu

5 rows × 40 columns

In [8]:
```
#this list will become important later on in the notebook, to calculate fea
list(df_train_features)
```

Out[8]:
```
['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']
```

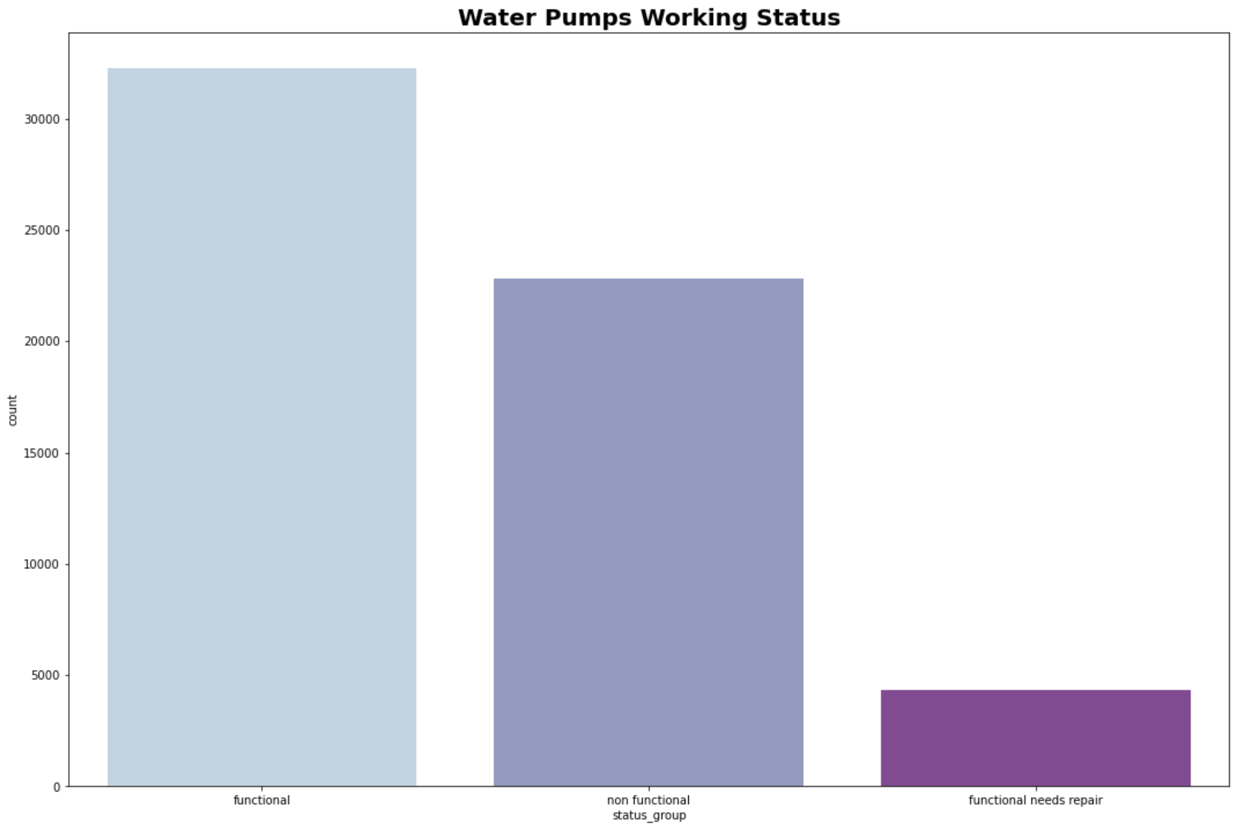
In [9]:
```
#Here we can see the number of wells recorded and their operational status
df_train_labels.status_group.value_counts()
```

Out[9]:
```
functional                 32259
non functional             22824
functional needs repair     4317
Name: status_group, dtype: int64
```

In [10]:
```
#converting this data to percentages
df_train_labels.status_group.value_counts(normalize=True)
```

Out[10]:
```
functional                 0.543081
non functional             0.384242
functional needs repair    0.072677
Name: status_group, dtype: float64
```

In [11]:
```
#Let's use seaborn to graph out the first and simplest aspect of our data!
import seaborn as sns
plt.figure(figsize=(18,12))
plt.title("Water Pumps Working Status",fontsize=20, fontweight='bold')
sns.countplot(x = df_train_labels['status_group'], data = df_train_labels,
```



In [12]:
```
#Here we can see which basins have the highest numbers of each well status
df_basins_status = pd.concat([df_train_features, df_train_labels], axis=1)
piv_df = df_basins_status[['basin','status_group','construction_year']]
piv_table = piv_df.pivot_table(index='basin',
                               columns='status_group', aggfunc='count')
piv_table
```

Out[12]:

	construction_year		
status_group	**functional**	**functional needs repair**	**non functional**
basin			
Internal	4482	557	2746
Lake Nyasa	3324	250	1511
Lake Rukwa	1000	270	1184
Lake Tanganyika	3107	742	2583
Lake Victoria	5100	989	4159
Pangani	5372	477	3091
Rufiji	5068	437	2471
Ruvuma / Southern Coast	1670	326	2497
Wami / Ruvu	3136	269	2582

In [13]:
```
#Next we can take a look at the water quality for the same metrics
df_water_quality_status = pd.concat([df_train_features, df_train_labels], a
piv_df = df_water_quality_status[['water_quality','status_group','basin']]
piv_table = piv_df.pivot_table(index='water_quality',
                               columns='status_group', aggfunc='count')
piv_table
```

Out[13]:

	basin		
status_group	**functional**	**functional needs repair**	**non functional**
water_quality			
coloured	246.0	54.0	190.0
fluoride	151.0	13.0	36.0
fluoride abandoned	6.0	NaN	11.0
milky	438.0	14.0	352.0
salty	2220.0	225.0	2411.0
salty abandoned	174.0	72.0	93.0
soft	28760.0	3904.0	18154.0
unknown	264.0	35.0	1577.0

In [14]:
```
#Let's combine our test and train features
full_df = pd.concat([df_train_features, df_test])
```

In [15]:
```
full_df.shape
```

Out[15]: (74250, 40)

In [16]:
```
#A closer look att our merged data to see what type of data we will be work
full_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 74250 entries, 0 to 14849
Data columns (total 40 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   id                     74250 non-null  int64  
 1   amount_tsh             74250 non-null  float64
 2   date_recorded          74250 non-null  object 
 3   funder                 69746 non-null  object 
 4   gps_height             74250 non-null  int64  
 5   installer              69718 non-null  object 
 6   longitude              74250 non-null  float64
 7   latitude               74250 non-null  float64
 8   wpt_name               74250 non-null  object 
 9   num_private            74250 non-null  int64  
 10  basin                  74250 non-null  object 
 11  subvillage             73780 non-null  object 
 12  region                 74250 non-null  object 
 13  region_code            74250 non-null  int64  
 14  district_code          74250 non-null  int64  
 15  lga                    74250 non-null  object 
 16  ward                   74250 non-null  object 
 17  population             74250 non-null  int64  
 18  public_meeting         70095 non-null  object 
 19  recorded_by            74250 non-null  object 
 20  scheme_management      69404 non-null  object 
 21  scheme_name            38992 non-null  object 
 22  permit                 70457 non-null  object 
 23  construction_year      74250 non-null  int64  
 24  extraction_type        74250 non-null  object 
 25  extraction_type_group  74250 non-null  object 
 26  extraction_type_class  74250 non-null  object 
 27  management             74250 non-null  object 
 28  management_group       74250 non-null  object 
 29  payment                74250 non-null  object 
 30  payment_type           74250 non-null  object 
 31  water_quality          74250 non-null  object 
 32  quality_group          74250 non-null  object 
 33  quantity               74250 non-null  object 
 34  quantity_group         74250 non-null  object 
 35  source                 74250 non-null  object 
 36  source_type            74250 non-null  object 
 37  source_class           74250 non-null  object 
 38  waterpoint_type        74250 non-null  object 
 39  waterpoint_type_group  74250 non-null  object 
dtypes: float64(3), int64(7), object(30)
memory usage: 23.2+ MB
```

In [17]:
```
full_df.head()
```

Out[17]:

	id	amount_tsh	date_recorded	funder	gps_height	installer	longitude	latitude	wpt_na
0	69572	6000.0	2011-03-14	Roman	1390	Roman	34.938093	-9.856322	r
1	8776	0.0	2013-03-06	Grumeti	1399	GRUMETI	34.698766	-2.147466	Zaha
2	34310	25.0	2013-02-25	Lottery Club	686	World vision	37.460664	-3.821329	Mah
3	67743	0.0	2013-01-28	Unicef	263	UNICEF	38.486161	-11.155298	Zaha Nanyu
4	19728	0.0	2011-07-13	Action In A	0	Artisan	31.130847	-1.825359	Shu

5 rows × 40 columns

OSEMN- Scrubbing our data

Dates are especially tricky and can throw wrenches into our algorithm later on, let's fix that before going any further

In [18]:
```
full_df['date_recorded_months'] = [(pd.to_datetime(date)-pd.to_datetime('20
```

Next we begin he very important task of dropping superfluous columns, and replacing missing data

In [19]:
```
full_df['scheme_name_duplicate'] = full_df['scheme_name']
```

In [20]:
```
#use the .mask() function to replace values that satisfy the conditions
full_df = full_df.apply(lambda x: x.mask(x.map(x.value_counts())<250, 'NaN')
```

In [21]:
```
full_df.scheme_name_duplicate.value_counts()
```

Out[21]:
```
NaN                            32724
K                                858
None                             794
Borehole                         704
Chalinze wate                    501
M                                490
DANIDA                           483
Government                       395
Ngana water supplied scheme      335
wanging'ombe water supply s      323
Bagamoyo wate                    296
wanging'ombe supply scheme       284
I                                281
Uroki-Bomang'ombe water sup      266
N                                258
Name: scheme_name_duplicate, dtype: int64
```

In [22]:
```
#Our mask fucntion worked pretty well, let's do the same for our other colu
full_df = full_df.apply(lambda x: x.mask(x.map(x.value_counts())<250, 'NaN'
full_df = full_df.apply(lambda x: x.mask(x.map(x.value_counts())<250, 'NaN'
full_df = full_df.apply(lambda x: x.mask(x.map(x.value_counts())<150, 'NaN'
full_df = full_df.apply(lambda x: x.mask(x.map(x.value_counts())<100, 'NaN'
full_df = full_df.apply(lambda x: x.mask(x.map(x.value_counts())<150, 'NaN'
```

In [23]:
```
full_df['public_meeting_missing'] = full_df['public_meeting'].isna()
```

In [24]:
```
full_df['public_meeting'] = full_df['public_meeting'].fillna(full_df['publi
```

In [25]:
```
full_df['scheme_management_missing'] = full_df['scheme_management'].isna()
```

In [26]:
```
full_df['scheme_management'] = full_df['scheme_management'].fillna(full_df[
```

In [27]:
```
#if the permit is missing, we can simply merge these columns and use NaN fe
full_df['permit_missing'] = full_df['permit'].isna()
```

In [28]:
```
full_df['permit'] = full_df['permit'].fillna(full_df['permit'].mode()[0])
full_df['construction_year_missing'] = (full_df['construction_year'] ==0)*1
mean_year = full_df[full_df['construction_year']>0]['construction_year'].me
full_df.loc[full_df['construction_year']==0, 'construction_year'] = int(mea
```

In [29]:
```
full_df = full_df.replace('none', np.NaN)
```

In [30]:
```
full_df = full_df.replace('0', np.NaN)
```

In [31]:
```
full_df['gps_height_missing'] = full_df['gps_height'].isna()
```

In [32]:
```
mean_gps_height = full_df[full_df['gps_height']>0]['gps_height'].mean()
full_df.loc[full_df['gps_height']==0, 'gps_height'] = int(mean_gps_height)
```

In [33]:
```
full_df['num_private_missing'] = full_df['num_private'].isna()
```

In [34]:
```
mean_num_private = full_df[full_df['num_private']>0]['num_private'].mean()
full_df.loc[full_df['num_private']==0, 'num_private'] = int(mean_num_privat
```

In [35]:
```
full_df['population_missing'] = full_df['population'].isna()
```

In [36]:
```
mean_population = full_df[full_df['population']>0]['population'].mean()
full_df.loc[full_df['population']==0, 'population'] = int(mean_population)
```

In [37]:
```
full_df['amount_tsh_missing'] = full_df['amount_tsh'].isna()
```

In [38]:
```
mean_amount = full_df[full_df['amount_tsh']>0]['amount_tsh'].mean()
full_df.loc[full_df['amount_tsh']==0, 'amount_tsh'] = int(mean_amount)
```

In [39]:
```
full_df_columns = full_df.drop(columns=['scheme_name','date_recorded','lga'
                                        'payment_type', 'management_
                                        ])
```

In [40]:
```
full_df.head()
```

Out[40]:

	id	amount_tsh	date_recorded	funder	gps_height	installer	longitude	latitude	wpt_nam
0	69572	6000.0	2011-03-14	Roman	1390	NaN	34.938093	-9.856322	Na
1	8776	1065.0	2013-03-06	NaN	1399	NaN	34.698766	-2.147466	Zahana
2	34310	25.0	2013-02-25	NaN	686	World vision	37.460664	-3.821329	Na
3	67743	1065.0	2013-01-28	Unicef	263	UNICEF	38.486161	-11.155298	Na
4	19728	1065.0	2011-07-13	NaN	1058	NaN	31.130847	-1.825359	Shule

5 rows × 50 columns

In [41]:
```
df_main = pd.get_dummies(full_df_columns)
```

In [42]:
```
df_main.head()
```

Out[42]:

	id	amount_tsh	gps_height	longitude	latitude	num_private	region_code	district_code
0	69572	6000.0	1390	34.938093	-9.856322	36	11	5
1	8776	1065.0	1399	34.698766	-2.147466	36	20	2
2	34310	25.0	686	37.460664	-3.821329	36	21	4
3	67743	1065.0	263	38.486161	-11.155298	36	90	63
4	19728	1065.0	1058	31.130847	-1.825359	36	18	1

5 rows × 293 columns

In [43]:
```
df_main.shape
```
Out[43]:
```
(74250, 293)
```

Now let's split the data once again so we can run a train test split, and model our data effectively

In [44]:
```
X_cleaned = df_main[:-14850]
X_test_main_cleaned = df_main[-14358:]
y = df_train_labels['status_group']
```

In [45]:
```
X_cleaned.shape, X_test_main_cleaned.shape, y.shape
```
Out[45]:
```
((59400, 293), (14358, 293), (59400,))
```

In [46]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X_cleaned, y, test_size
```

In [47]:
```
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```
Out[47]:
```
((44550, 293), (14850, 293), (44550,), (14850,))
```

In [48]:
```
X_train.head()
```
Out[48]:

	id	amount_tsh	gps_height	longitude	latitude	num_private	region_code	district
24947	33935	20.0	330	38.123839	-6.087137e+00	36	6	
22630	49654	1065.0	1058	0.000000	-2.000000e-08	36	17	
13789	39287	1065.0	1058	33.312321	-2.814100e+00	36	19	
15697	60510	1065.0	1542	34.783049	-4.842093e+00	36	13	
22613	24259	1065.0	523	34.660944	-1.070733e+01	36	10	

5 rows × 293 columns

OSEMN-Model

Now that our data is cleaned and organized, we can start modeling our dataset using Logistic Regression, Random Forests, and Gradient Boosting to check for feature importance. We can also create a heatmap of our data set, set on top of a map of Tanzania

In [49]:
```
from sklearn import tree

dtf = tree.DecisionTreeClassifier()
dtf.fit(X = X_train, y = y_train)
dtf.feature_importances_
dtf.score(X = X_test, y = y_test)
```
Out[49]:
```
0.7418855218855219
```

In [50]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from scipy.cluster import hierarchy as hc
```

In [51]:
```
AS = RandomForestClassifier(n_estimators=250,min_samples_leaf=3 ,n_jobs=-1,
%time AS.fit(X_train, y_train)
y_pred= AS.predict(X_test)
accuracy_score(y_test, y_pred)
```

```
CPU times: user 2min 14s, sys: 1.47 s, total: 2min 16s
Wall time: 1min 3s
```

Out[51]:
```
0.8103030303030303
```

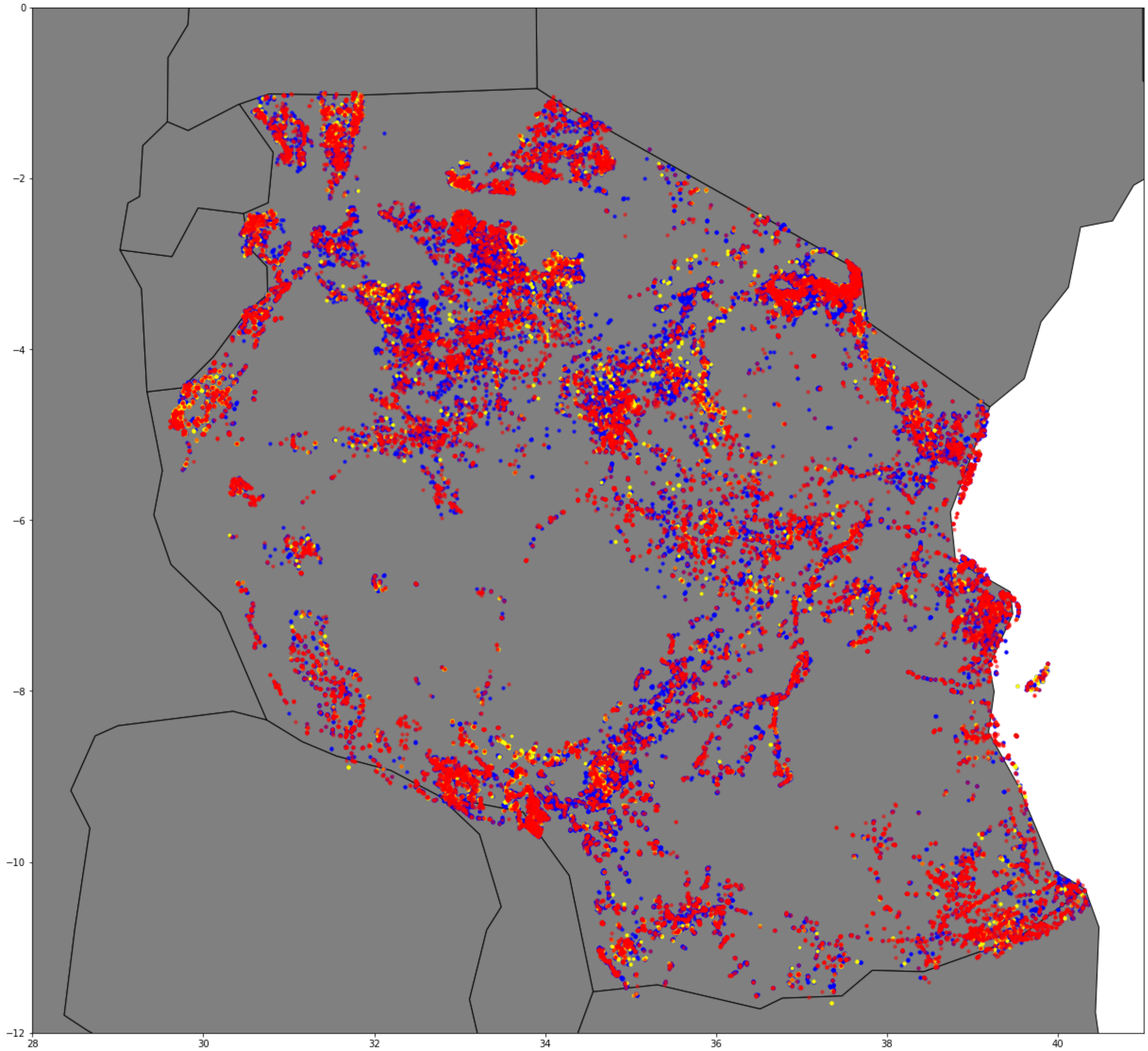
Not too bad! Let's install some extra packages that can show us the geographical distribution of the wells, so we can see how that relates to feature importance, as well as findings problematic area that may require additional attention

In [52]:
```
!pip install descartes
from pylab import rcParams
rcParams['figure.figsize'] = 30, 20
!pip install category_encoders
!pip install geopandas
import geopandas
```

```
Requirement already satisfied: descartes in ./opt/anaconda3/envs/learn-en
v/lib/python3.8/site-packages (1.1.0)
Requirement already satisfied: matplotlib in ./opt/anaconda3/envs/learn-e
nv/lib/python3.8/site-packages (from descartes) (3.3.1)
Requirement already satisfied: numpy>=1.15 in ./opt/anaconda3/envs/learn-
env/lib/python3.8/site-packages (from matplotlib->descartes) (1.18.5)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.3 i
n ./opt/anaconda3/envs/learn-env/lib/python3.8/site-packages (from matplo
tlib->descartes) (2.4.7)
Requirement already satisfied: kiwisolver>=1.0.1 in ./opt/anaconda3/envs/
learn-env/lib/python3.8/site-packages (from matplotlib->descartes) (1.2.
0)
Requirement already satisfied: python-dateutil>=2.1 in ./opt/anaconda3/en
vs/learn-env/lib/python3.8/site-packages (from matplotlib->descartes) (2.
8.1)
Requirement already satisfied: cycler>=0.10 in ./opt/anaconda3/envs/learn
-env/lib/python3.8/site-packages (from matplotlib->descartes) (0.10.0)
Requirement already satisfied: certifi>=2020.06.20 in ./opt/anaconda3/env
s/learn-env/lib/python3.8/site-packages (from matplotlib->descartes) (202
0.12.5)
Requirement already satisfied: pillow>=6.2.0 in ./opt/anaconda3/envs/lear
n-env/lib/python3.8/site-packages (from matplotlib->descartes) (7.2.0)
Requirement already satisfied: six>=1.5 in ./opt/anaconda3/envs/learn-en
v/lib/python3.8/site-packages (from python-dateutil>=2.1->matplotlib->des
cartes) (1.15.0)
Requirement already satisfied: category_encoders in ./opt/anaconda3/envs/
learn-env/lib/python3.8/site-packages (2.2.2)
Requirement already satisfied: pandas>=0.21.1 in ./opt/anaconda3/envs/lea
rn-env/lib/python3.8/site-packages (from category_encoders) (1.1.3)
Requirement already satisfied: numpy>=1.14.0 in ./opt/anaconda3/envs/lear
n-env/lib/python3.8/site-packages (from category_encoders) (1.18.5)
Requirement already satisfied: patsy>=0.5.1 in ./opt/anaconda3/envs/learn
-env/lib/python3.8/site-packages (from category_encoders) (0.5.1)
Requirement already satisfied: scipy>=1.0.0 in ./opt/anaconda3/envs/learn
-env/lib/python3.8/site-packages (from category_encoders) (1.5.2)
Requirement already satisfied: statsmodels>=0.9.0 in ./opt/anaconda3/env
s/learn-env/lib/python3.8/site-packages (from category_encoders) (0.12.0)
Requirement already satisfied: scikit-learn>=0.20.0 in ./opt/anaconda3/en
vs/learn-env/lib/python3.8/site-packages (from category_encoders) (0.23.
2)
Requirement already satisfied: python-dateutil>=2.7.3 in ./opt/anaconda3/
envs/learn-env/lib/python3.8/site-packages (from pandas>=0.21.1->category
_encoders) (2.8.1)
Requirement already satisfied: pytz>=2017.2 in ./opt/anaconda3/envs/learn
-env/lib/python3.8/site-packages (from pandas>=0.21.1->category_encoders)
(2020.1)
Requirement already satisfied: six in ./opt/anaconda3/envs/learn-env/lib/
python3.8/site-packages (from patsy>=0.5.1->category_encoders) (1.15.0)
Requirement already satisfied: joblib>=0.11 in ./opt/anaconda3/envs/learn
-env/lib/python3.8/site-packages (from scikit-learn>=0.20.0->category_enc
```

```
oders) (0.17.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in ./opt/anaconda3/en
vs/learn-env/lib/python3.8/site-packages (from scikit-learn>=0.20.0->cate
gory_encoders) (2.1.0)
Requirement already satisfied: geopandas in ./opt/anaconda3/envs/learn-en
v/lib/python3.8/site-packages (0.8.2)
Requirement already satisfied: shapely in ./opt/anaconda3/envs/learn-env/
lib/python3.8/site-packages (from geopandas) (1.7.1)
Requirement already satisfied: fiona in ./opt/anaconda3/envs/learn-env/li
b/python3.8/site-packages (from geopandas) (1.8.18)
Requirement already satisfied: pyproj>=2.2.0 in ./opt/anaconda3/envs/lear
n-env/lib/python3.8/site-packages (from geopandas) (3.0.0.post1)
Requirement already satisfied: pandas>=0.23.0 in ./opt/anaconda3/envs/lea
rn-env/lib/python3.8/site-packages (from geopandas) (1.1.3)
Requirement already satisfied: click<8,>=4.0 in ./opt/anaconda3/envs/lear
n-env/lib/python3.8/site-packages (from fiona->geopandas) (7.1.2)
Requirement already satisfied: certifi in ./opt/anaconda3/envs/learn-env/
lib/python3.8/site-packages (from fiona->geopandas) (2020.12.5)
Requirement already satisfied: attrs>=17 in ./opt/anaconda3/envs/learn-en
v/lib/python3.8/site-packages (from fiona->geopandas) (20.2.0)
Requirement already satisfied: cligj>=0.5 in ./opt/anaconda3/envs/learn-e
nv/lib/python3.8/site-packages (from fiona->geopandas) (0.7.1)
Requirement already satisfied: six>=1.7 in ./opt/anaconda3/envs/learn-en
v/lib/python3.8/site-packages (from fiona->geopandas) (1.15.0)
Requirement already satisfied: munch in ./opt/anaconda3/envs/learn-env/li
b/python3.8/site-packages (from fiona->geopandas) (2.5.0)
Requirement already satisfied: click-plugins>=1.0 in ./opt/anaconda3/env
s/learn-env/lib/python3.8/site-packages (from fiona->geopandas) (1.1.1)
Requirement already satisfied: python-dateutil>=2.7.3 in ./opt/anaconda3/
envs/learn-env/lib/python3.8/site-packages (from pandas>=0.23.0->geopanda
s) (2.8.1)
Requirement already satisfied: pytz>=2017.2 in ./opt/anaconda3/envs/learn
-env/lib/python3.8/site-packages (from pandas>=0.23.0->geopandas) (2020.
1)
Requirement already satisfied: numpy>=1.15.4 in ./opt/anaconda3/envs/lear
n-env/lib/python3.8/site-packages (from pandas>=0.23.0->geopandas) (1.18.
5)
```

In [53]:
```
gdf = geopandas.GeoDataFrame(
    df_main, geometry=geopandas.points_from_xy(df_main.longitude, df_main.l

functional = gdf.where(df_train_labels['status_group'] == 'functional')
repair = gdf.where(df_train_labels['status_group'] == 'functional needs rep
broken = gdf.where(df_train_labels['status_group'] == 'non functional')

world = geopandas.read_file(geopandas.datasets.get_path('naturalearth_lowre

ax = world[world.continent == 'Africa'].plot(
    color='grey', edgecolor='black')
ax.scatter(functional['longitude'], functional['latitude'],
           c='blue',alpha=.75, s=10)
ax.scatter(repair['longitude'], repair['latitude'],
           c='yellow', alpha=1, s=10)
ax.scatter(broken['longitude'], broken['latitude'],
           c='red', alpha=.5, s=10)

plt.ylim(-12, 0)
plt.xlim(28,41)

plt.show()

#blue is functional, yellow needs repair, red is broken
```



Next, we will create dummy variables for our y_train, this will allow us to test for feature importance across the X-train axis, since object datatypes stopped our model from working the first time

In [54]:
```
y_train
```

Out[54]:
```
24947    non functional
22630        functional
13789        functional
15697        functional
22613    non functional
              ...
54343        functional
38158        functional
860      non functional
15795        functional
56422    non functional
Name: status_group, Length: 44550, dtype: object
```

In [55]:
```
dummy_y = pd.get_dummies(y_train)
```

In [56]:
```
dummy_y.head()
```

Out[56]:

	functional	functional needs repair	non functional
24947	0	0	1
22630	1	0	0
13789	1	0	0
15697	1	0	0
22613	0	0	1

In [57]:
```
dummy_y.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 44550 entries, 24947 to 56422
Data columns (total 3 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   functional               44550 non-null  uint8
 1   functional needs repair  44550 non-null  uint8
 2   non functional           44550 non-null  uint8
dtypes: uint8(3)
memory usage: 478.6 KB
```

Now we can fit our model using RandoomForestRegressor, and use XGBClassifier to test for feature importance, then lastly use matplotlib and pyplot to graph that importance in respective order

In [58]:
```
from sklearn.ensemble import RandomForestRegressor
rfr = RandomForestRegressor(n_estimators=200)
rfr.fit(X_train, dummy_y)
```

Out[58]:
```
RandomForestRegressor(n_estimators=200)
```

In [59]:
```
y_train.shape
```

Out[59]:
```
(44550,)
```

We created quite a few extra columns when cleaning and merging our data, let's use the original column list for the sake of simplicity, this is where the list of X_train columns from earlier comes in handy!

In [60]:
```
X_train = X_train.filter(['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'])
```

In [61]:
```
!pip install xgboost
```

```
Requirement already satisfied: xgboost in ./opt/anaconda3/envs/learn-env/
lib/python3.8/site-packages (1.2.0)
Requirement already satisfied: numpy in ./opt/anaconda3/envs/learn-env/li
b/python3.8/site-packages (from xgboost) (1.18.5)
Requirement already satisfied: scipy in ./opt/anaconda3/envs/learn-env/li
b/python3.8/site-packages (from xgboost) (1.5.2)
```

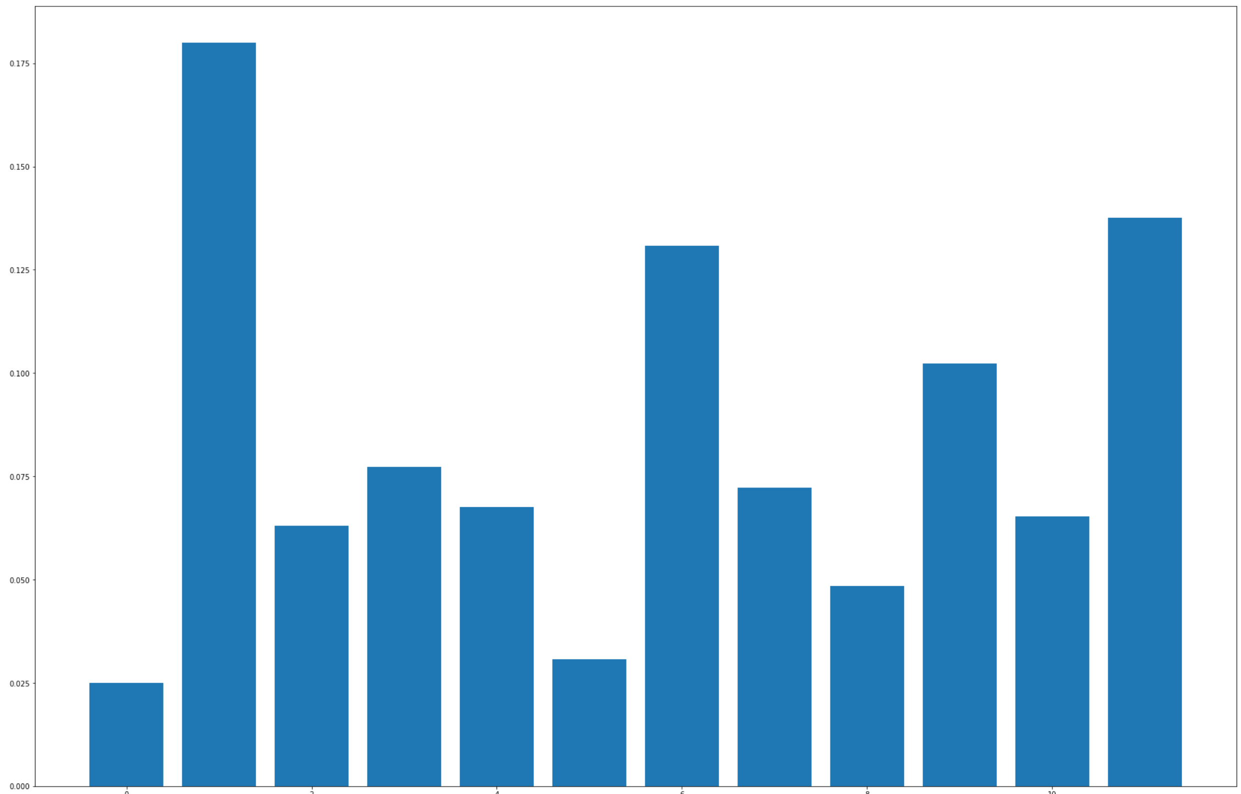
In [62]:
```
from xgboost import XGBClassifier
model = XGBClassifier()
model.fit(X_train, y_train)
print(model.feature_importances_)
```

```
[0.02500879 0.1799182  0.0629772  0.07722336 0.06762131 0.03079929
 0.13076232 0.07219969 0.04843081 0.10226837 0.06528778 0.13750286]
```

Neat! it worked, let's graph that out:

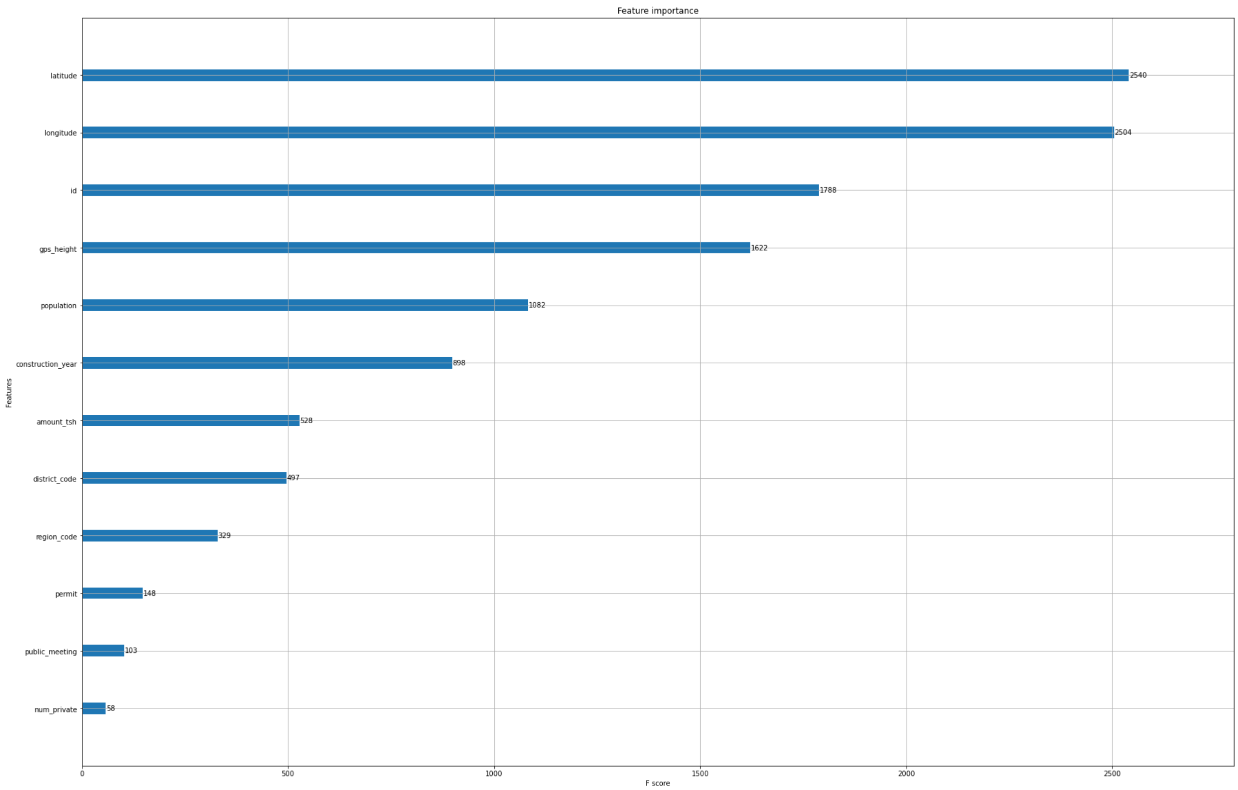
In [63]:
```
from matplotlib import pyplot
```

In [64]:
```
pyplot.bar(range(len(model.feature_importances_)), model.feature_importance
pyplot.show()
```



Looks ok, let's organize the results by importance, and add our labels for visibility.

In [65]:
```
from numpy import loadtxt
from xgboost import plot_importance
plot_importance(model)
pyplot.show()
```



OSEMN-iNterpret

Our Feature importance model made it statistically clear what we saw in our geopandas model, and our pivot tables from earlier. The location of a well plays a very significant factor in it's funcitonality, and the further north we go, the more functional wells we will find. This is most likely due to the proximity to Lake Victoria, Africas largest body of water, and the largest basin for wells in the country. "id" being third is most likely a statistical anomaly, as it's bearing on functionality would be purely coincidental (the ids are not in order).

We can see that our tree model performed rather well, with just over 81% accuracy in identifying functional wells.

In [ ]:

In [ ]: