Feature engineering

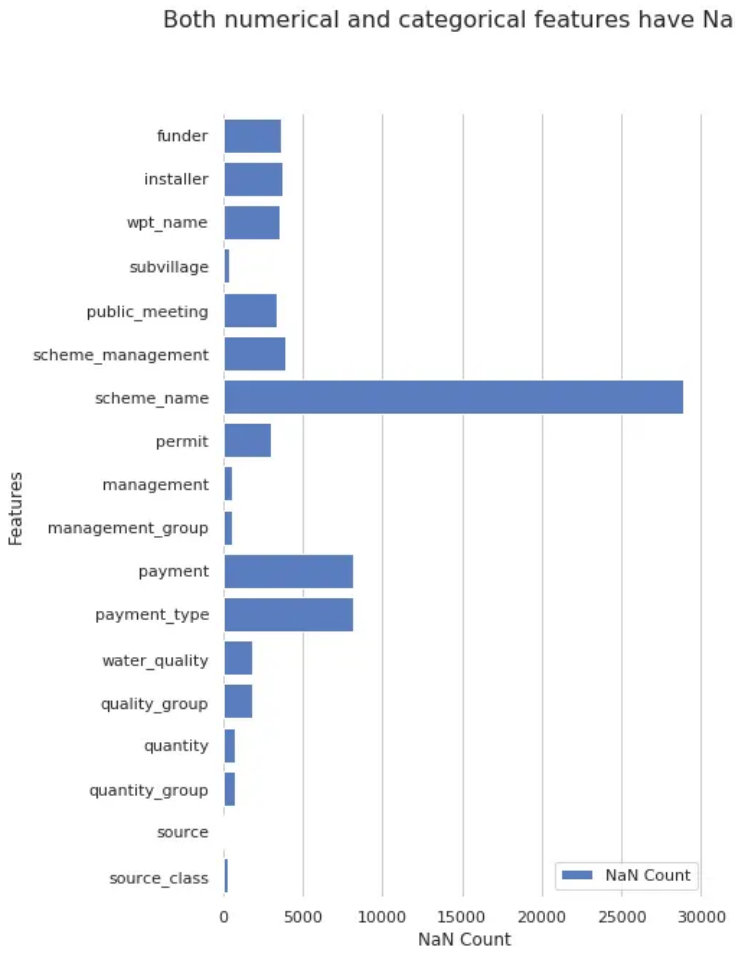
Feature engineering means identifying the relationships between independent and dependent features. Then the identified relationships we can add as polynomial or interaction features.

Feature engineering step is the point of entry for successive iterations. This is a critical step and plays a greater role in predictions as compared to model validation.

We can either drop columns with high cardinality and NaN values if needed. Also for numerical features, we can fill the NaN values with its mean, median or mode instead of dropping it. In the same way for categorical features, we can categorize the NaN as a separate category.

Also based on the water table data set, we can engineer few features like:

- NaN values in **longitude** and **latitude** updated with its mean.




```
# Apply mean for missing values of latitude
and longitude
```

```
mean_longitude =
df['longitude'].mean()
df['longitude'] =
df['longitude'].apply(lambda x:
mean_longitude if round(x, 2) == 0
else x)
mean_latitude = df['latitude'].mean()
df['latitude'] =
df['latitude'].apply(lambda x:
mean_latitude if round(x, 2) == 0
else x)
```

- Binary feature to identify if **funder** has at least funded 5 pumps. In the same was identify if **installer** has installed at least 5 pumps.

	# Identify items who have funded atleast 5 pumps
	if str(row) == "nan":
	return np.nan
	value_count = value_count_series.get(row)
	if value_count < count:
	return 0
	else:
	return 1
	# Create a column to indicate funder with atleast 5 pumps maintained.
	value_count_funder = df.funder.value_counts()
	df['funder_aleast_5'] = df['funder'].apply(atleast,
	args=(value_count_funder,))
	# Create a column to indicate installer with atleast 5 pumps maintained.
	value_count_installer = df.installer.value_counts()
	df['installer_aleast_5'] = df['installer'].apply(atleast,
	args=(value_count_installer,))

- Split the **date_recorded** into **year_recorded** and **month_recorded**. Even group in different bins.

```
# split year from date_recorded
return int(row.split('-')[0])

def compute_month_recorded(row):
# split year from date_recorded
return int(row.split('-')[1])

# Fetch Year and Month of date recorded
df['year_recorded'] = df['date_recorded'].apply(compute_year_recorded)
df['month_recorded'] = df['date_recorded'].apply(compute_month_recorded)
```

- Compute the age of pump based on **construction_year** and **year_recorded**.

Hyper-parameter tuning	