Loading Packages

In [1]:

```
import pandas as pd
import numpy as np

import matplotlib as mlp
import matplotlib.pyplot as plt
%matplotlib inline

import seaborn as sns
sns.set()

import bokeh as bk
from bokeh.io import output_notebook, show
output_notebook()

# custom
import helper
```

(https://bokeh.org) Loading BokehJS ...

Styling Tables

In [2]:

```
%%HTML
<style type='text/css'>
table.dataframe th, table.dataframe td{
    border: 3px solid purple !important;
    color: solid black !important;
}
</style>
```

Loading Dataset

In [3]:

```
dataset = 'Akosombo2.xlsx'

try:
    df = pd.read_excel(dataset,'Akosombo')
    print ("Successfully loaded dataset.")
except:
    print ("Something went wrong.")

df.head()
```

```
Successfully loaded dataset.
```

Out[3]:

	Date	Date.1	Upstream Elevation (feet)	Downstream Elevation (feet)	Discharge (cfs)	Generation (GWh)	Generation (W)	Upstrean Elevatio (m
0	1967-01-01	1967-01-01	255.80	44.64	5303.183746	1.31	5.458333e+07	77.96784
1	1967-01-02	1967-01-02	255.79	44.62	5553.333922	1.38	5.750000e+07	77.96479
2	1967-01-03	1967-01-03	255.78	44.71	4503.604240	1.68	7.000000e+07	77.96174
3	1967-01-04	1967-01-04	255.78	44.72	6003.604240	1.68	7.000000e+07	77.96174
4	1967-01-05	1967-01-05	255.75	44.72	4503.664311	1.69	7.041667e+07	77.95260

Saving Dataset Column Heads to `csv`

In [4]:

```
dataset_columns = pd.DataFrame({'column_names':list(df.columns)})
dataset_columns.to_csv("column_heads_of_dataset.csv", index=True)
dataset_columns
```

Out[4]:

	column_names
0	Date
1	Date.1
2	Upstream Elevation (feet)
3	Downstream Elevation (feet)
4	Discharge (cfs)
5	Generation (GWh)
6	Generation (W)
7	Upstream Elevation (m)
8	Downstreamstream Elevation (m)
9	Norminal Head (H)
10	Discharge (cms)
11	Efficiency
12	Unnamed: 12
13	general efficiency
14	ground acceleration

Dropping Unnecessary Columns

In [5]:

```
def drop_columns(column_names, df):
    df = df.drop(column_names, axis=1)
    return df
```

In [6]:

```
columns_to_drop = ['Date', 'Date.1', 'Upstream Elevation (feet)',
                   'Downstream Elevation (feet)', 'Discharge (cfs)', 'Generation (W)',
                   'Upstream Elevation (m)', 'Downstreamstream Elevation (m)', 'Efficie
                   'Unnamed: 12', 'general efficiency ', 'ground acceleration']

df = drop_columns(columns_to_drop, df)

df.head()
```

Out[6]:

	Generation (GWh)	Norminal Head (H)	Discharge (cms)
0	1.31	64.361568	150.08010
1	1.38	64.364616	157.15935
2	1.68	64.334136	127.45200
3	1.68	64.331088	169.90200
4	1.69	64.321944	127.45370

Formatting Column Heads

In [7]:

```
df.columns = df.columns.str.replace(' ', '_')
df.columns = df.columns.str.lower()
df.columns
```

Out[7]:

```
Index(['generation_(gwh)', 'norminal_head_(h)', 'discharge_(cms)'], dtype='o
bject')
```

Renaming Column Heads

In [8]:

```
df.rename({
    'generation_(gwh)':'generation',
    'norminal_head_(h)':'norminal_head',
    'discharge_(cms)':'discharge'
}, axis='columns', inplace=True)

df.columns
```

Out[8]:

```
Index(['generation', 'norminal_head', 'discharge'], dtype='object')
```

In [9]:

```
df.head()
```

Out[9]:

	generation	norminal_head	discharge
0	1.31	64.361568	150.08010
1	1.38	64.364616	157.15935
2	1.68	64.334136	127.45200
3	1.68	64.331088	169.90200
4	1.69	64.321944	127.45370

Checking For Missing Values and Saving Results to `csv`

In [10]:

```
missing_values = helper.missing_data(df)
missing_values.to_csv("missing_values.csv", index=True)
missing_values
```

Out[10]:

	generation	norminal_head	discharge
Total	0	0	0
Percent	0	0	0
Types	float64	float64	float64

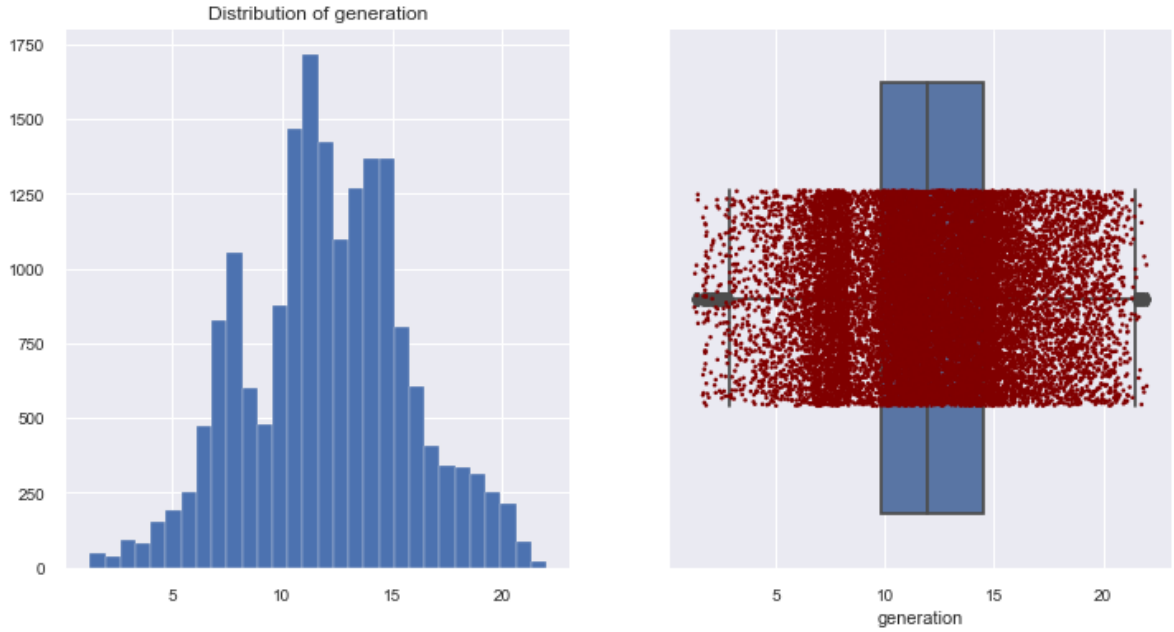
In [11]:

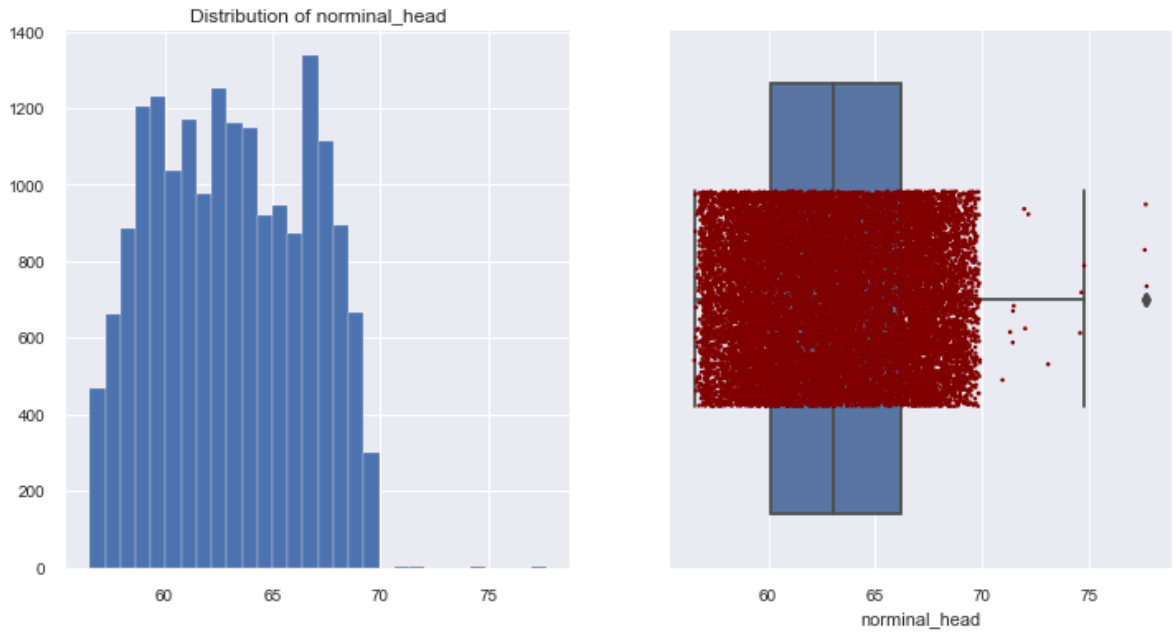
```
# Saving column heads to variable
column_names = df.columns
```

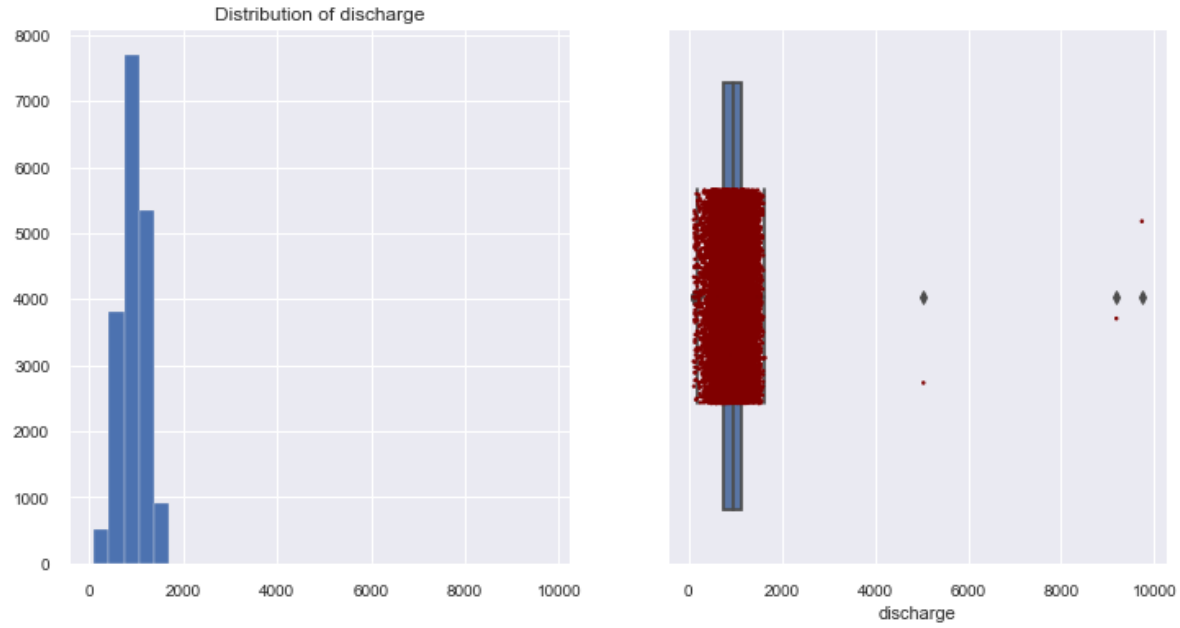
Checking For Outliers

In [12]:

```
helper.detect_outliers(df)
# plt.savefig('outliers.png', dpi=300, transparent=True)
```







Removing Outliers

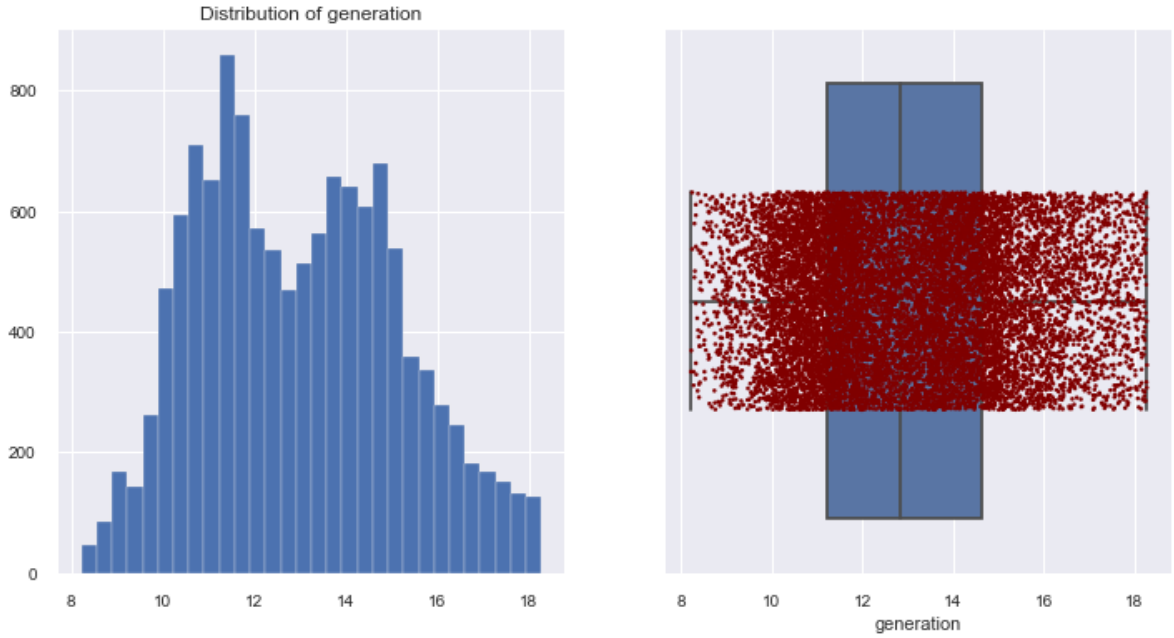
In [13]:

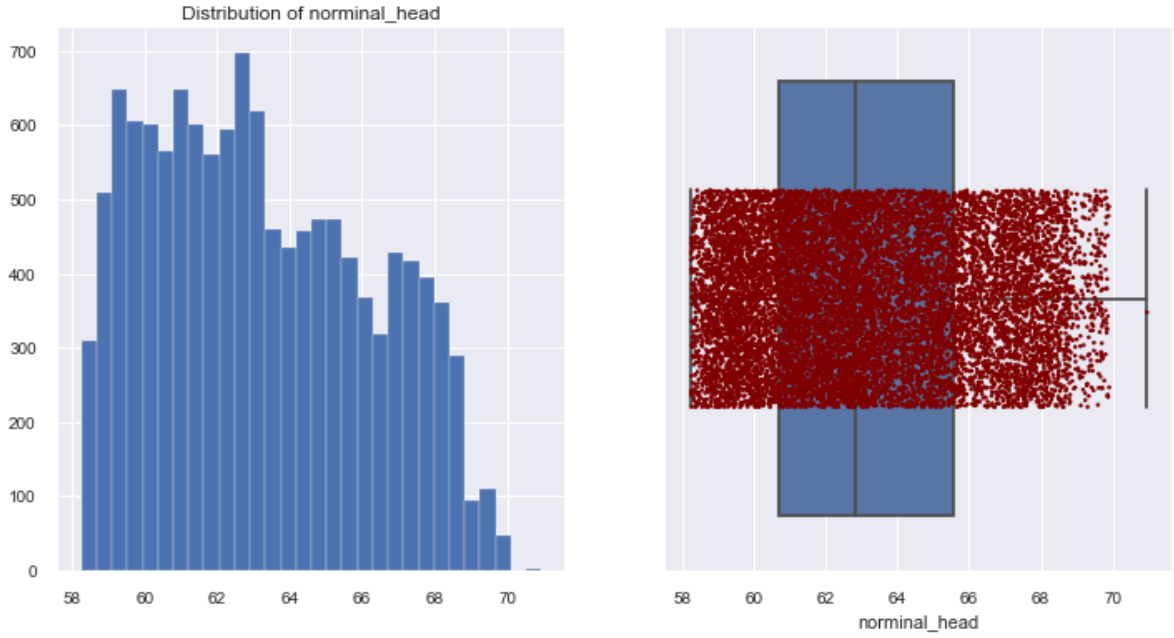
```
1 df = helper.outlier_remover(df)
```

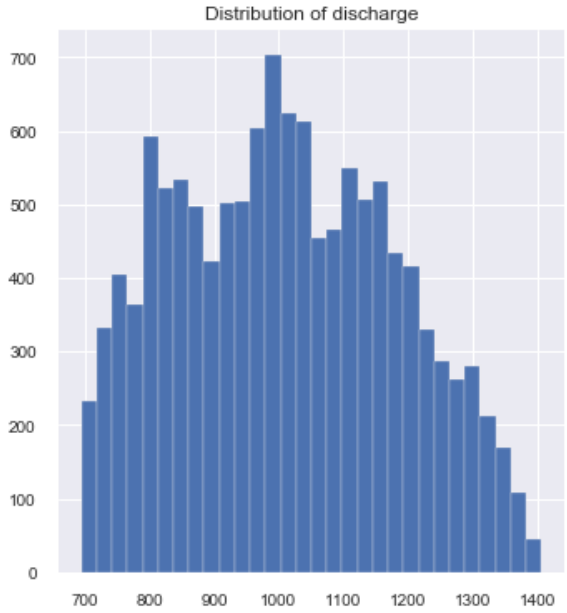
	Q1	Q3	IQR	lower_bound	upper_bound
generation	11.960000	14.48750	2.527500	8.16875	18.278750
norminal_head	63.044832	66.24066	3.195828	58.25109	71.034402
discharge	961.897756	1140.49000	178.592244	694.00939	1408.378366

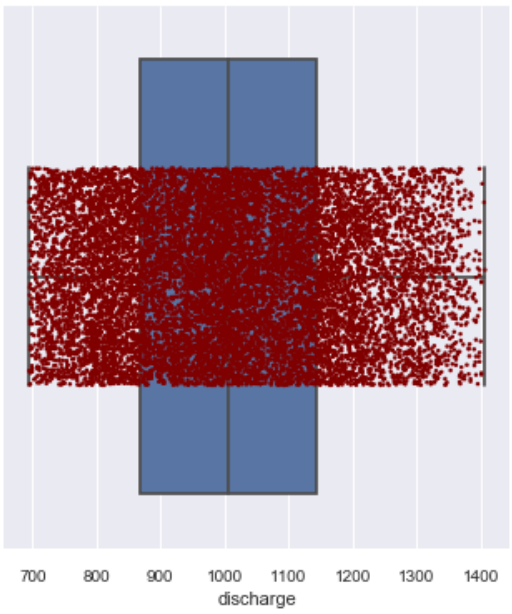
In [14]:

```
1 helper.detect_outliers(df)
```









In [15]:

```
1 df.head()
```

Out[15]:

	generation	norminal_head	discharge
1990	9.75	67.440048	736.2420
1992	9.89	67.418712	716.4371
1993	9.82	67.409568	713.6054
1996	9.65	67.403472	730.5786
1999	9.95	67.357752	722.1005

Correcting Data Index

In [16]:

```
1 df.index = range(len(df))
```

In [17]:

```
df = df[['norminal_head', 'discharge', 'generation']]
df.head()
```

Out[17]:

	norminal_head	discharge	generation
0	67.440048	736.2420	9.75
1	67.418712	716.4371	9.89
2	67.409568	713.6054	9.82
3	67.403472	730.5786	9.65
4	67.357752	722.1005	9.95

Checking Memory Usage

In [18]:

```
df.info(memory_usage='deep')
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12502 entries, 0 to 12501
Data columns (total 3 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   norminal_head  12502 non-null  float64
 1   discharge      12502 non-null  float64
 2   generation     12502 non-null  float64
dtypes: float64(3)
memory usage: 293.1 KB
```

In [19]:

```
df.memory_usage(deep='True')
```

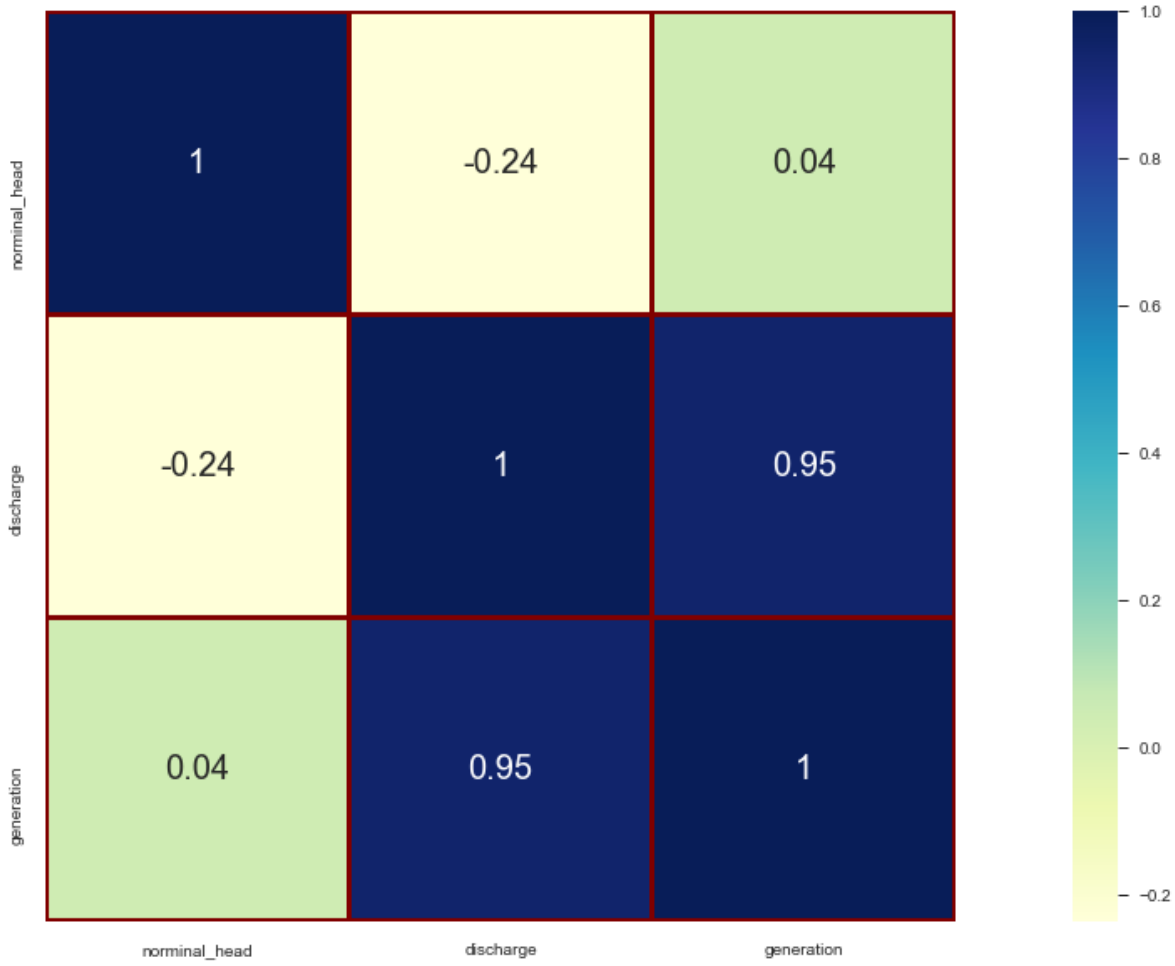
Out[19]:

```
Index              128
norminal_head    100016
discharge        100016
generation       100016
dtype: int64
```

Plotting Correlation Heat Map of Feature Variables

In [20]:

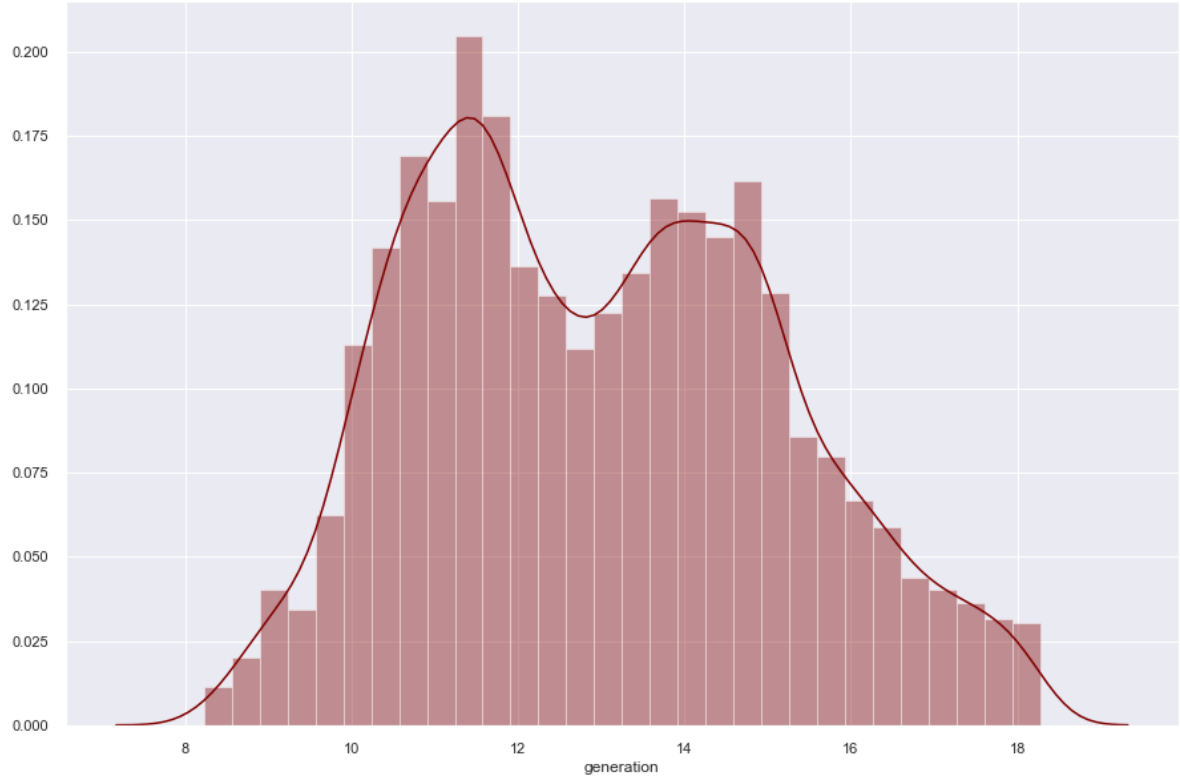
```
1 helper.correlation_viz(df, 'correlation_of_feature_variables', save=True, dpi=300, trar
```



Distribution of Target Variable

In [21]:

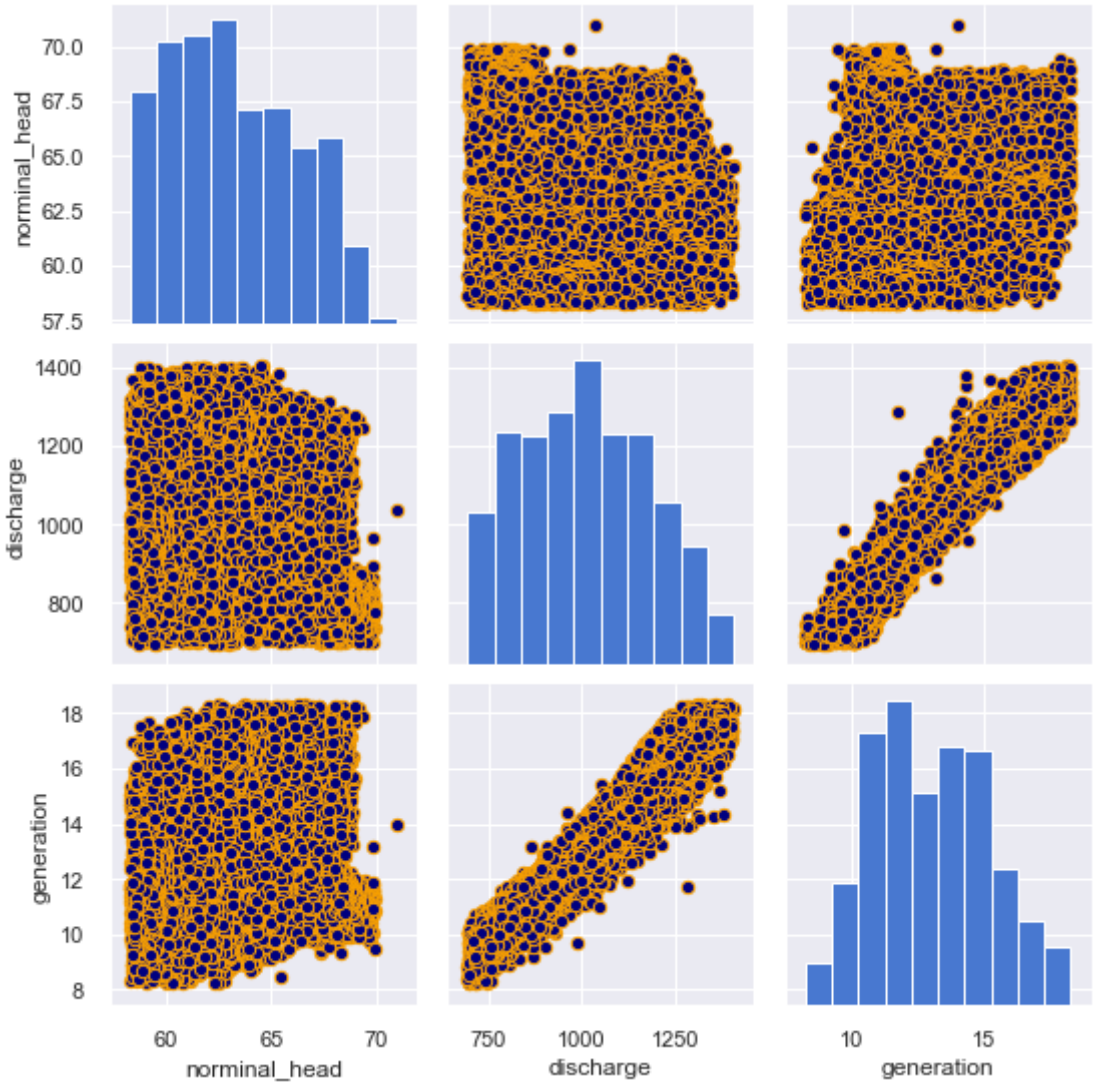
```
plt.figure(figsize=(15, 10))
sns.set(color_codes=True)
sns.set_palette(sns.color_palette('muted'))

sns.distplot(df['generation'], color='maroon', bins=30)
plt.savefig('Distribution_Plot_of_PowerGeneration.png', dpi=300, transparent=True)
```



Pairplot

In [22]:

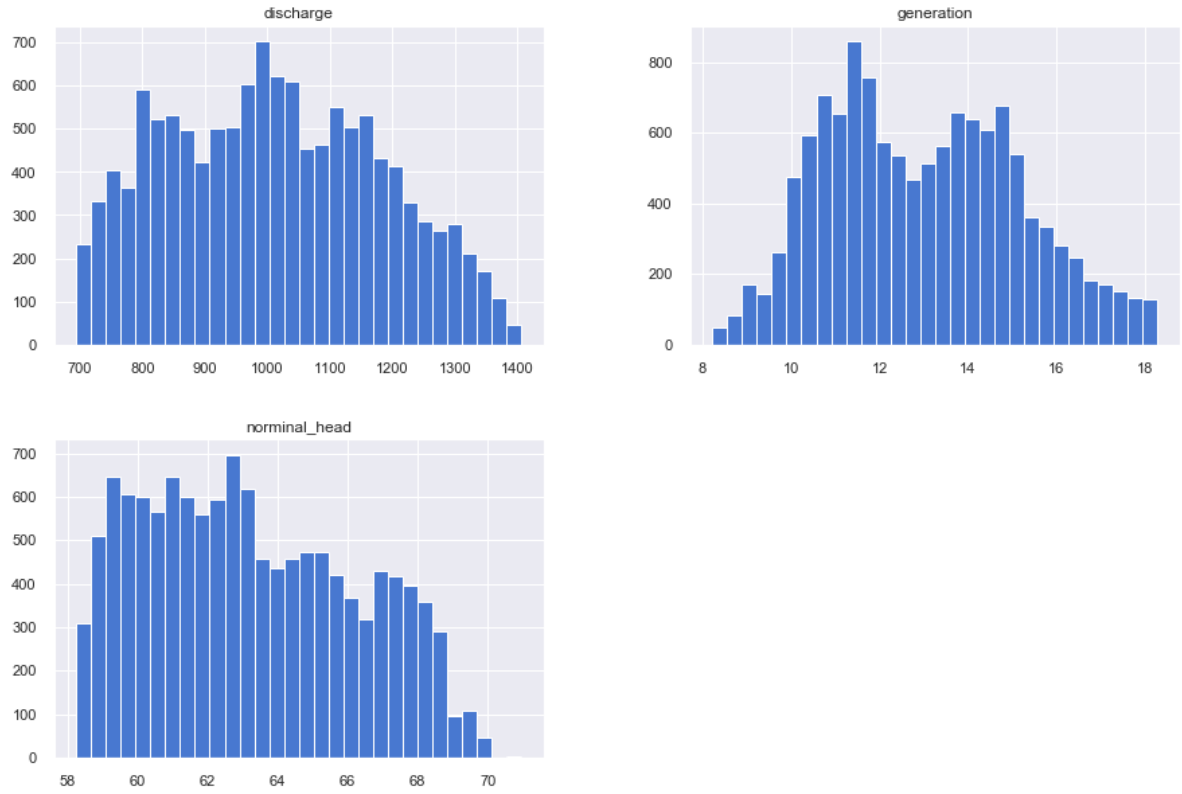
```
sns.pairplot(df, plot_kws=dict(s=40, edgecolor="orange", linewidth=1, facecolor='navy')
plt.savefig('PairPlot.png',dpi=300, transparent=True)
```



Histogram of Feature Variables

In [23]:

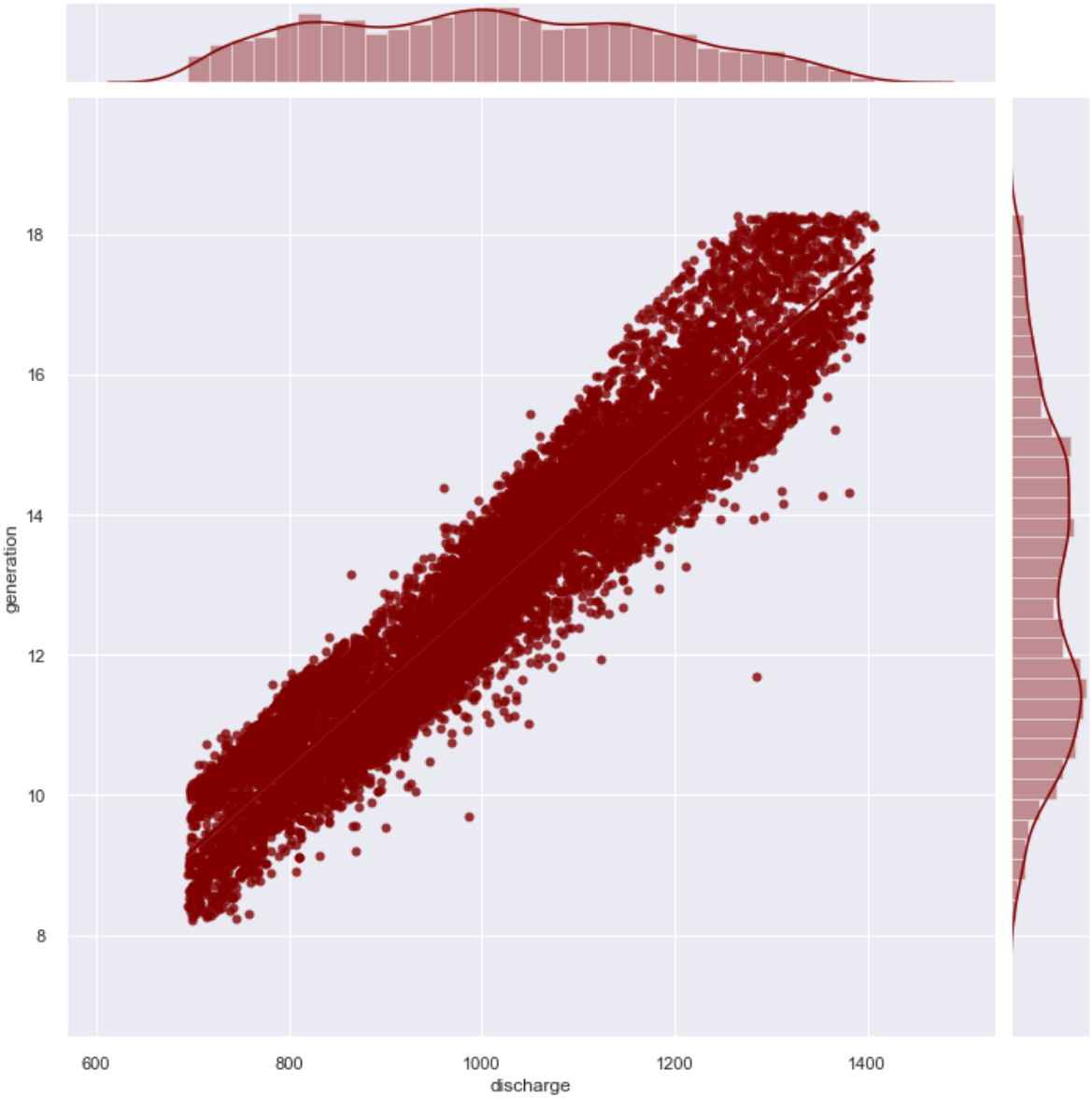
```
df.hist(bins=30, figsize=(15, 10),)
plt.savefig('Histogram_of_features_variables.png', dpi=300, transparent=True)
```



Joinplot of Feature Variables and Target Variable

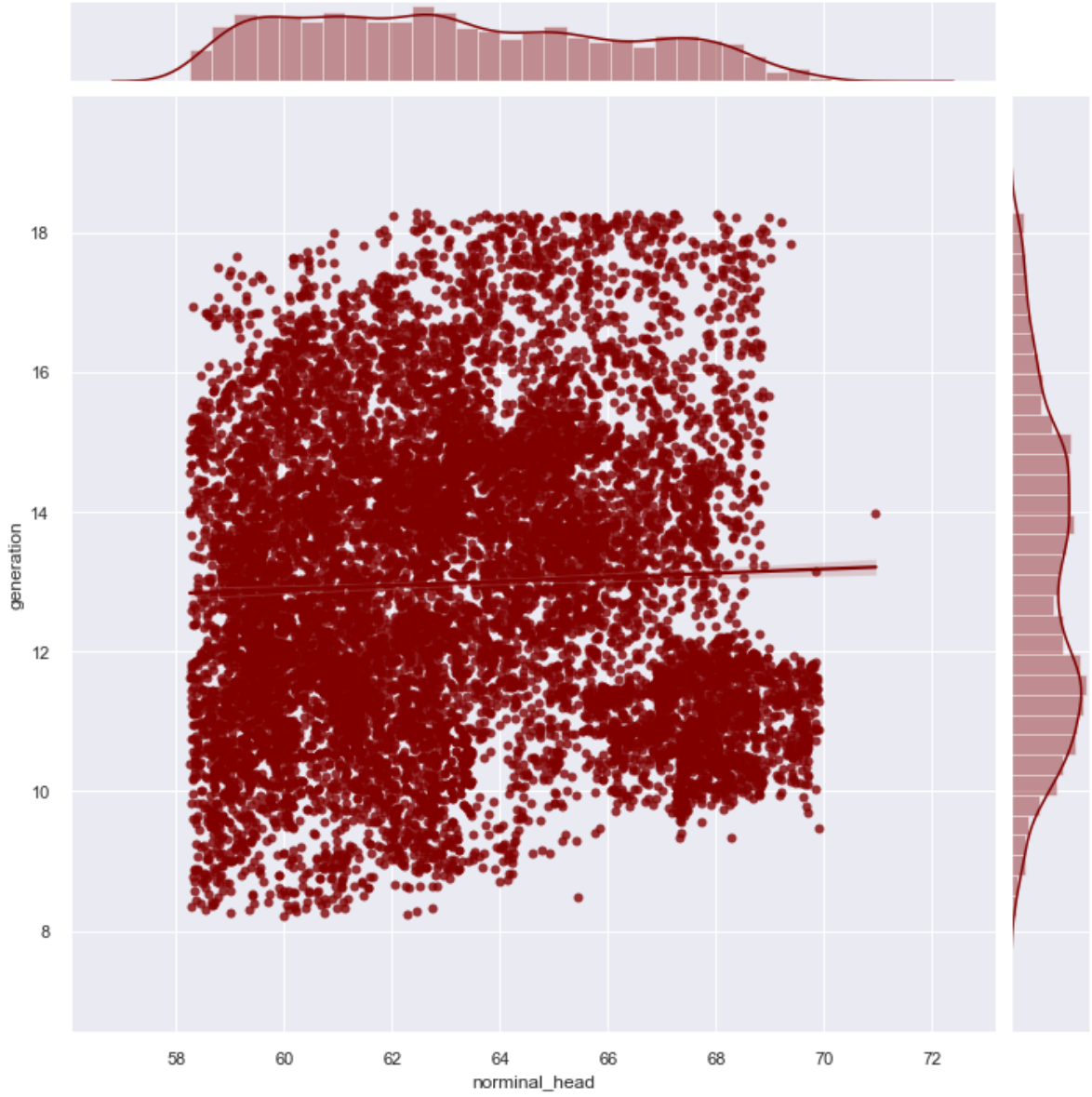
In [24]:

```
sns.jointplot('discharge', 'generation', df, kind='reg', height=10, ratio=10, color='ma
# annot_kws=dict(stat="r"), s=40, edgecolor="orange", facecolor='navy', linewidth=1,

plt.savefig('Jointplot_of_discharge_and_target_variable.png', dpi=300, transparent=True
```



In [25]:

```
sns.jointplot('norminal_head', 'generation', df, kind='reg', height=10, ratio=10, color
plt.savefig('Jointplot_of_norminal_head_and_target_variable.png', dpi=300, transparent=
```



Generating Data Report

In [26]:

```
import pandas_profiling as pp

profilling_report = pp.ProfileReport(df)
profilling_report.to_file('Akosombo_data_profile_report.html')
```

```
C:\ProgramData\Anaconda3\lib\site-packages\astropy\stats\bayesian_blocks.py:
429: RuntimeWarning: divide by zero encountered in log
  return N_k * (np.log(N_k) - np.log(T_k))
```

Saving Data to `csv`

In [27]:

```
1 df.to_csv('Clean_Akosombo_data.csv', index=True)
```

In [ ]:

```
1
```