K-Means Clustering Technique

In [16]:

```
#Importing the libraries "pandas, numpy, matplotlib.pyplot, scipy.stats and seaborn" to m
y python script
#with the standard short name as "pd, np, plt and sns".

#Uploading the file on google colab and choosing the selected dataset by clicking "choose
files".

import pandas as pd
import numpy as np
import pylab
import scipy.stats as stats
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
%matplotlib inline

from google.colab import files
uploaded = files.upload()
```

Choose File **No file selected**

Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving New_Data4.csv to New_Data4 (1).csv
```

Data Input

In [17]:

```
#In the first step, importing the dataset in the project by the "read_csv" function
#and that reads the data into a pandas dataframe object.

dataframe = pd.read_csv('New_Data4.csv')
```

In [18]:

```
#head() method returns a particular rows from the top and where did not mention
#the number below hence returns first 5 rows

dataframe.head()
```

Out[18]:

	Product_Code	W0	W1	W2	W3	W4	W5	W6	W7	W8	...	Normalized 42	Normalized 43	Normalized 44	Normalized 45	Normalized 46
0	P1	11	12	10	8	13	12	14	21	6	...	0.06	0.22	0.28	0.39	0.50
1	P2	7	6	3	2	7	1	6	3	3	...	0.20	0.40	0.50	0.10	0.10
2	P3	7	11	8	9	10	8	7	13	12	...	0.27	1.00	0.18	0.18	0.36
3	P4	12	8	13	5	9	6	9	13	13	...	0.41	0.47	0.06	0.12	0.24
4	P5	8	5	13	11	6	7	9	14	9	...	0.27	0.53	0.27	0.60	0.20

5 rows × 107 columns

In [19]:

```
#.iloc is an integer index based method of selecting rows and columns
```

```
#from a Pandas dataframe of 52 columns

X = dataframe.iloc[:, 1:53]
X.head()
```

Out[19]:

	W0	W1	W2	W3	W4	W5	W6	W7	W8	W9	...	W42	W43	W44	W45	W46	W47	W48	W49	W50	W51
0	11	12	10	8	13	12	14	21	6	14	...	4	7	8	10	12	3	7	6	5	10
1	7	6	3	2	7	1	6	3	3	3	...	2	4	5	1	1	4	5	1	6	0
2	7	11	8	9	10	8	7	13	12	6	...	6	14	5	5	7	8	14	8	8	7
3	12	8	13	5	9	6	9	13	13	11	...	9	10	3	4	6	8	14	8	7	8
4	8	5	13	11	6	7	9	14	9	9	...	7	11	7	12	6	6	5	11	8	9

5 rows × 52 columns

In [20]:

```
#The describe() method gives description of the data in the dataframe
#The details are included in the description for each column if the dataframe includes nu
merical data.

X.describe()
```

Out[20]:

	W0	W1	W2	W3	W4	W5	W6	W7	W8	W
count	811.000000	811.000000	811.000000	811.000000	811.000000	811.000000	811.000000	811.000000	811.000000	811.0000
mean	8.902589	9.129470	9.389642	9.717633	9.574599	9.466091	9.720099	9.585697	9.784217	9.6818
std	12.067163	12.564766	13.045073	13.553294	13.095765	12.823195	13.347375	13.049138	13.550237	13.1379
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0000
25%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0000
50%	3.000000	3.000000	3.000000	4.000000	4.000000	3.000000	4.000000	4.000000	4.000000	4.0000
75%	12.000000	12.000000	12.000000	13.000000	13.000000	12.500000	13.000000	12.500000	13.000000	13.0000
max	54.000000	53.000000	56.000000	59.000000	61.000000	52.000000	56.000000	62.000000	63.000000	52.0000

8 rows × 52 columns

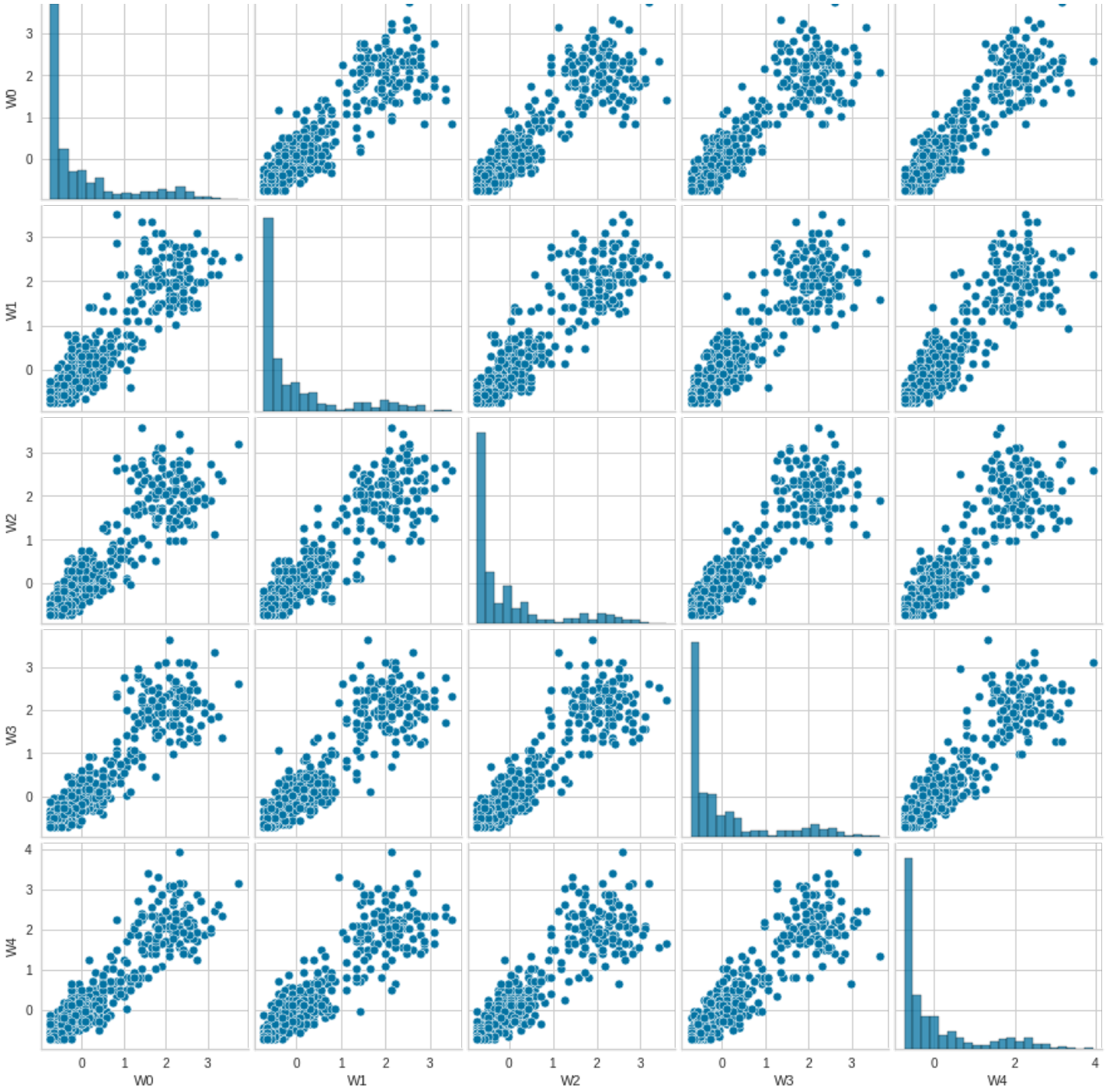
Feature Scaling and Transformation

In [21]:

```
#Importing standardscaler from sklearn.preprocessing.
#standardscaler() function used to remove the mean and scales of each
#feature or variable to unit variance.

#fit_transform is used on the training data so that we can scale the
#training data and also learn the scaling parameters of that data.

from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaled_X = pd.DataFrame(scaler.fit_transform(X))
scaled_X.columns = X.columns
```

In [22]:

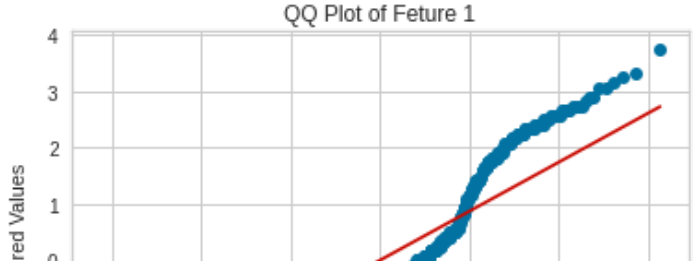
```
#.pairplot allows us to plot pairwise relationships between variables within a dataset.

_ = sns.pairplot(scaled_X.iloc[:,0:5])
```

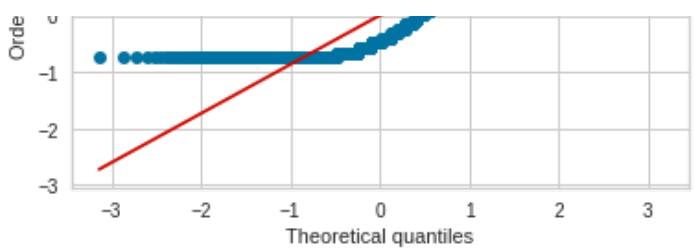


QQ Plot - Scaled Data

In [23]:

```
#QQ plot is the quantiles of the first data set against the quantiles of the second data
set.

#.probplot() function generates a probability plot of sample data against the
#quantiles of a specified theoretical distribution.

#Showing the theoretical quantiles in x axis and ordered values in y axis.

stats.probplot(scaled_X.iloc[:,0], dist="norm", plot=pylab)
plt.title('QQ Plot of Feture 1')
pylab.show()
```

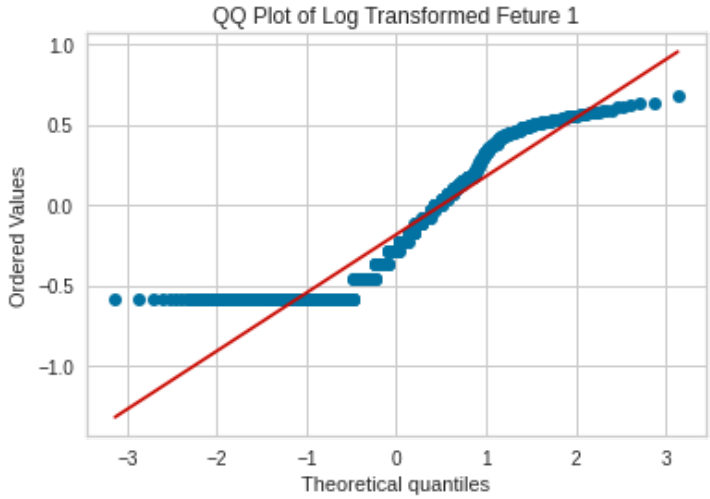




QQ Plot - Log Transformed Data

In [24]:

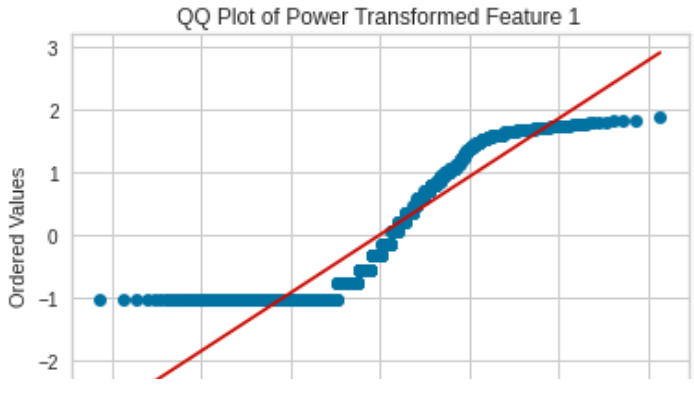
```
#Now here trying with log transformed data after the scaled data with some changes.

#Here the transformation showing the theoretical quantiles in x axis and ordered values i
n y axis.

X_LogTransformed = np.log10(scaled_X + 1)

stats.probplot(X_LogTransformed.iloc[:,0], dist="norm", plot=pylab)
plt.title('QQ Plot of Log Transformed Feture 1')
pylab.show()
```

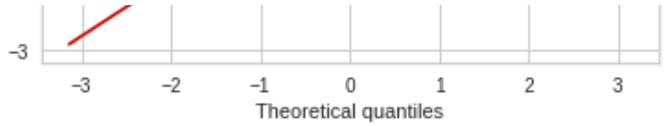


QQ Plot - Power Transformed Data

In [25]:

```
#No using the poertransformer() is applied to create a monotonic
#transformation of data using power functions.

#Importing poertransformer from sklearn.preprocessing for improvement.

from sklearn.preprocessing import PowerTransformer
pt = PowerTransformer()
X_PowerTransformed = pt.fit_transform(scaled_X)

stats.probplot(X_PowerTransformed[:,0], dist="norm", plot=pylab)
plt.title('QQ Plot of Power Transformed Feature 1')
pylab.show()
```





K-Means Clustering

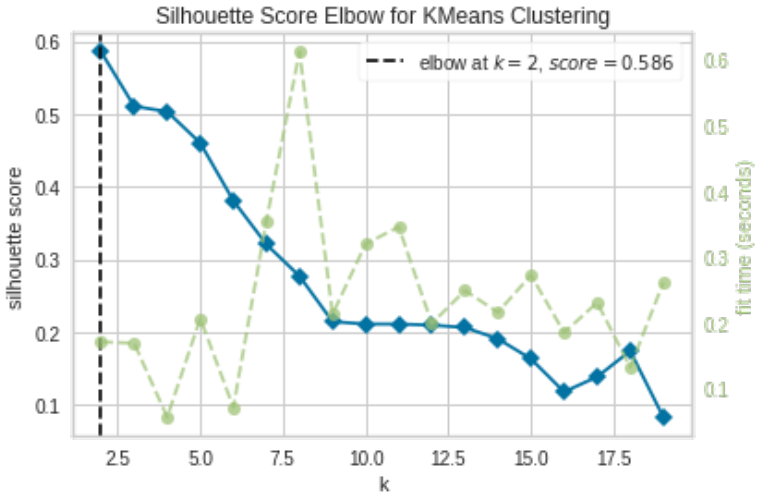
In [26]:

```
#Importing KElbowvisualizer which implements the "elbow" method of selecting the
#optimal number of clusters for K-means clustering.

#Importing silhouette_score is used for measuring the mean of the Silhouette Coefficient
#for each sample belonging to different clustersused for measuring the mean of the
#Silhouette Coefficient for each sample belonging to different clusters.

from yellowbrick.cluster import KElbowVisualizer
from sklearn.metrics import silhouette_score
```

Elbow Method

In [27]:

```
#An elbow method is a graphical representation of finding the optimal 'K' in a K-means cl
ustering.

#Kelbowvisualizer visualizes the clusters according to a scoring function, looking for an
"elbow" in the curve.

X = X_LogTransformed.to_numpy()
km = KMeans()
visualizer = KElbowVisualizer(km, k = (2,20), locate_elbow=True, metric = 'silhouette')
visualizer.fit(X)
_ = visualizer.poof()
```



Silhouette Scoring

In [28]:

```
#Silhouette score is a metric used to calculate the goodness of a clustering technique.

#The score for K-Means clustering is shown below.

print('Silhouette Score for K-Means Clustering: {}'.format(silhouette_score(X, km.labels
_, metric = 'euclidean')))
```

```
Silhouette Score for K-Means Clustering: 0.08333748185748018
```