In [1]:
```python
import pandas as pd
import numpy as np
```

In [2]:
```python
data = pd.read_csv(r'C:\Users\Tess-leslie\Documents\newcryptocurrency_p
rices.csv')
data
```

Out[2]:

	CryptoCurrency Type	Trade Time(US Time)	Daily High	Trade Date	Daily Low	Open	52-Week High	Prev.Close
0	BTC	06:13AM	8,231.15	6/10/2019	7,931.36	8,168.93	13,829.07	8,168.24
1	ETH	06:13AM	177.6656	6/10/2019	172.4018	177.1539	362.819	177.0681
2	XRP	06:13AM	0.2569	6/10/2019	0.2478	0.2543	0.5655	0.2543
3	BCC	06:13AM	226.3218	6/10/2019	218.3572	222.8465	642.8143	222.8102
4	LTC	06:13AM	57.3156	6/10/2019	55.4602	56.8784	145.8824	56.8571
...	...	...	...	...	...	...	...	...
615	ETC	04:40AM	4.8759	15/11/2019	4.6647	4.744	9.8644	4.7454
616	NEO	04:40AM	13.3447	15/11/2019	12.2852	12.6726	20.9391	12.6755
617	LINK	04:37AM	3.1058	15/11/2019	2.9763	3.1058	3.1346	3.0984
618	XEM	04:37AM	0.0422	15/11/2019	0.0398	0.0402	0.0975	0.0401
619	ZEC	04:40AM	37.0196	15/11/2019	35.7038	36.428	124.1407	36.4309

620 rows × 16 columns

In [3]:
```python
from pandas.api.types import is_string_dtype

def train_cats(df):
```

```
    for n,c in df.items():
        if is_string_dtype(c): df[n] = c.astype('category').cat.as_orde
red()

def apply_cats(df, trn):
    for n,c in df.items():
        if (n in trn.columns) and (trn[n].dtype.name=='category'):
            df[n] = pd.Categorical(c, categories=trn[n].cat.categories,
 ordered=True)
            df[n] = df[n].cat.codes
```

In [4]:

```
#first, create a copy where we apply the changes:
data_test = data.copy()

train_cats(data)
#this converts 'string' variables to categories datatype

apply_cats(data_test, data)
#this applys the numerical labelling onto categorical variables, but it
 is done on data (internally),
#u will not see it at the display. Then it uses the labeling as a refer
ence and apply it on 'data_test',
#which you will only see the actual labelling conversion
```

In [5]:

```
data_test
```

Out[5]:

	CryptoCurrency Type	Trade Time(US Time)	Daily High	Trade Date	Daily Low	Open	52-Week High	Prev.Close	52-Week Low	Price Direction	C
0	4	26	508	22	479	511	27	512	11	1	
1	8	26	216	22	215	216	37	218	20	1	
2	18	26	95	22	87	92	13	92	6	0	
3	1	26	298	22	297	299	44	300	19	1	
4	10	26	418	22	419	426	31	427	10	1	

	CryptoCurrency Type	Trade Time(US Time)	Daily High	Trade Date	Daily Low	Open	52-Week High	Prev.Close	52-Week Low	Price Direction	C
...	...	...	...	...	...	...	...	...	...	...	
615	7	1	386	8	382	386	46	388	12	0	
616	12	1	188	8	181	185	34	186	16	0	
617	9	0	320	8	282	324	36	325	4	1	
618	15	0	48	8	38	38	1	40	2	1	
619	19	1	353	8	345	350	26	352	13	0	

620 rows × 16 columns

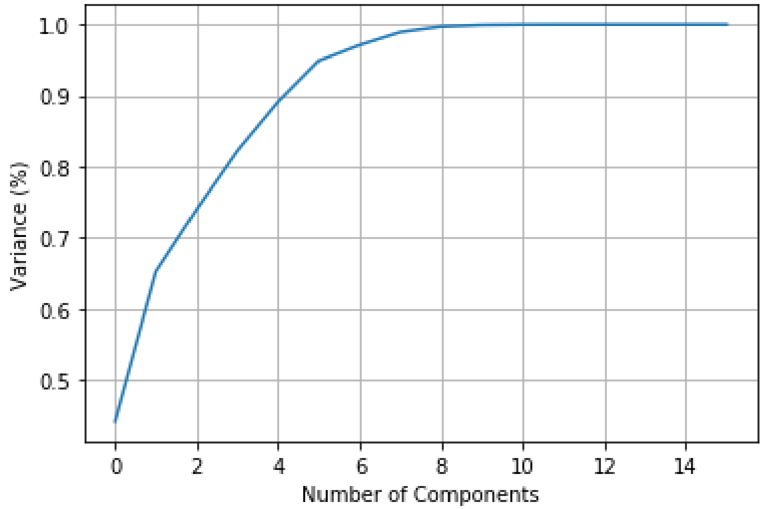
In [6]:
```
#Scaling the dataset
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
data_scaled = scaler.fit_transform(data_test.values)
```

In [7]:
```
data_scaled
```

Out[7]:
```
array([[2.10526316e-01, 6.66666667e-01, 9.37269373e-01, ...,
        9.17077902e-02, 5.19789957e-01, 5.18710622e-01],
       [4.21052632e-01, 6.66666667e-01, 3.98523985e-01, ...,
        8.73467223e-04, 1.03973307e-02, 1.06554029e-02],
       [9.47368421e-01, 6.66666667e-01, 1.75276753e-01, ...,
        6.26820604e-06, 1.42216388e-05, 4.60329508e-04],
       ...,
       [4.73684211e-01, 0.00000000e+00, 5.90405904e-01, ...,
        1.32652732e-05, 2.83338803e-04, 7.13248191e-04],
       [7.89473684e-01, 0.00000000e+00, 8.85608856e-02, ...,
        5.53934487e-06, 8.75177770e-07, 4.46764665e-04],
       [1.00000000e+00, 2.56410256e-02, 6.51291513e-01, ...,
        8.60639267e-04, 1.58450935e-03, 2.03691431e-03]])
```

In [18]:
```
#Using PCA to reduce dataset dimension
from sklearn.decomposition import PCA
data_pca = PCA().fit(data_scaled)
```

In [25]:
```
#Plot graph to check the number of components needed to explain the dat
aset
from matplotlib import pyplot as plt
from matplotlib.pyplot import figure
%matplotlib inline
plt.figure()
plt.plot(np.cumsum(data_pca.explained_variance_ratio_))
plt.xlabel('Number of Components')
plt.ylabel('Variance (%)') #for each component
plt.grid()
plt.show()
```

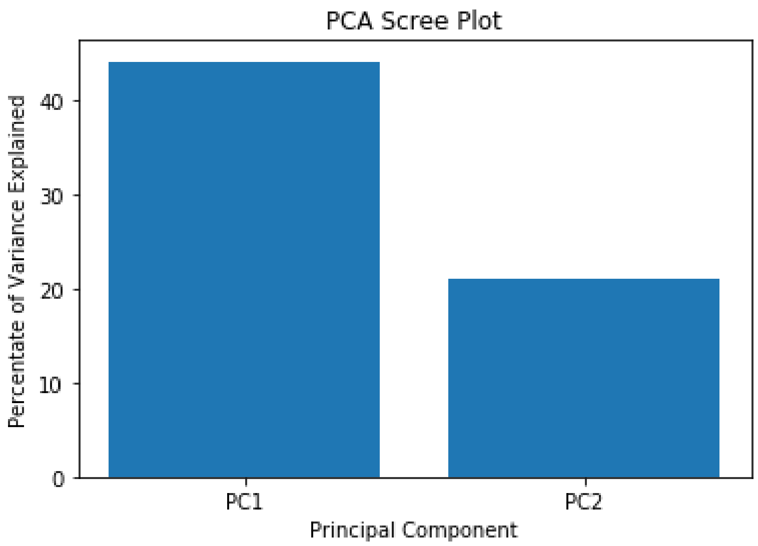


In [10]:
```
pca = PCA(n_components=2)

data_pca = pca.fit_transform(data_scaled)
```

In [11]:
```
#Plot barplot for variance explained
percent_variance = np.round(pca.explained_variance_ratio_* 100, decimal
```

```
s =2)
columns = ['PC1', 'PC2']
plt.bar(x= range(1,3), height=percent_variance, tick_label=columns)
plt.ylabel('Percentate of Variance Explained')
plt.xlabel('Principal Component')
plt.title('PCA Scree Plot')
plt.show()
```

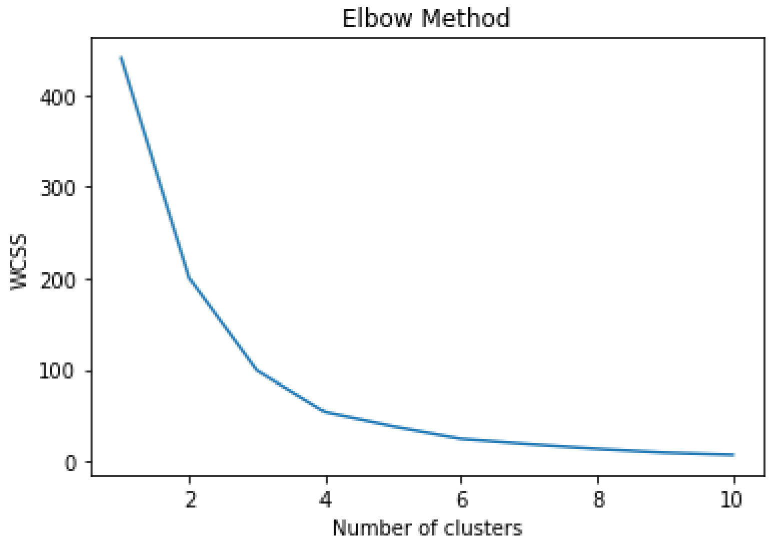


In [12]:
```
print('Explained variation per principal component: {}'.format(pca.expl
ained_variance_ratio_))
#The combined explained variance of PC1 and PC2 is 0.65 or 65%. That me
ans the two components was able to explain 65% of the entire dataset
```

```
Explained variation per principal component: [0.44111648 0.21112322]
```

In [13]:
```
#Find the number of cluster,k needed using the Elbow Plot
from sklearn.cluster import KMeans
wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, init='k-means++', max_iter=300, n_ini
t=10, random_state=0)
    kmeans.fit(data_pca)
```

```
    wcss.append(kmeans.inertia_)
plt.plot(range(1, 11), wcss)
plt.title('Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
```



In [14]:
```
data_pca_df = pd.DataFrame(data = data_pca, columns = ['PC1', 'PC2'])
data_pca_df.head()
```

Out[14]:

	PC1	PC2
0	-1.007772	0.357483
1	-0.288403	0.501530
2	0.840316	-0.227710
3	-0.628142	0.408959
4	-0.665149	0.422552

In [15]:
```
kmeans = KMeans(n_clusters=4).fit(data_pca_df)
```

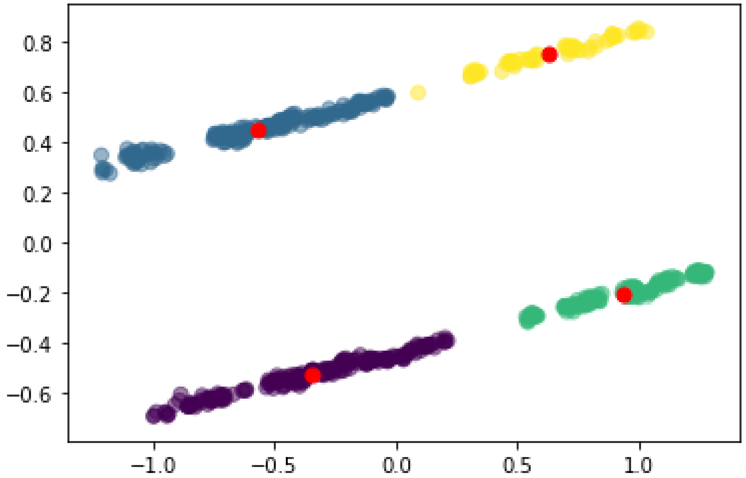
```
centroids = kmeans.cluster_centers_
print(centroids)# Centroid axis

plt.scatter(data_pca_df['PC1'], data_pca_df['PC2'], c= kmeans.labels_.a
stype(float), s=50, alpha=0.5)
plt.scatter(centroids[:, 0], centroids[:, 1], c='red', s=50)
figure(num=None, figsize=(8, 6), dpi=80, facecolor='w', edgecolor='k')
```

```
[[-0.3437467  -0.52348955]
 [-0.57187776  0.45269061]
 [ 0.93022956 -0.20140652]
 [ 0.62773645  0.75308469]]
```

Out[15]:

```
<Figure size 640x480 with 0 Axes>
```




```
<Figure size 640x480 with 0 Axes>
```

In [16]:

```
#From the cluster plot above, it can be observed that there are 4 clust
ers with their each centroids
```