Problem 1:

There are various stocks for which we have collected a data set, which all stocks are apparently similar in performance

Problem 2:

How many Unique patterns that exist in the historical stock data set, based on fluctuations in price.

Problem 3:

Identify which all stocks are moving together and which all stocks are different from each other.

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from sklearn import decomposition
from sklearn import datasets
import seaborn as sns
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
%matplotlib inline
```

In [2]:
```
df = pd.read_csv('data_stocks.csv')
```

In [3]:
```
df.head()
```

Out[3]:

	DATE	SP500	NASDAQ.AAL	NASDAQ.AAPL	NASDAQ.ADBE	NASDAQ.ADI	NASDAQ.A
0	1491226200	2363.6101	42.3300	143.6800	129.6300	82.040	102.2:
1	1491226260	2364.1001	42.3600	143.7000	130.3200	82.080	102.1‹
2	1491226320	2362.6799	42.3100	143.6901	130.2250	82.030	102.2'
3	1491226380	2364.3101	42.3700	143.6400	130.0729	82.000	102.1‹
4	1491226440	2364.8501	42.5378	143.6600	129.8800	82.035	102.0(

5 rows × 502 columns

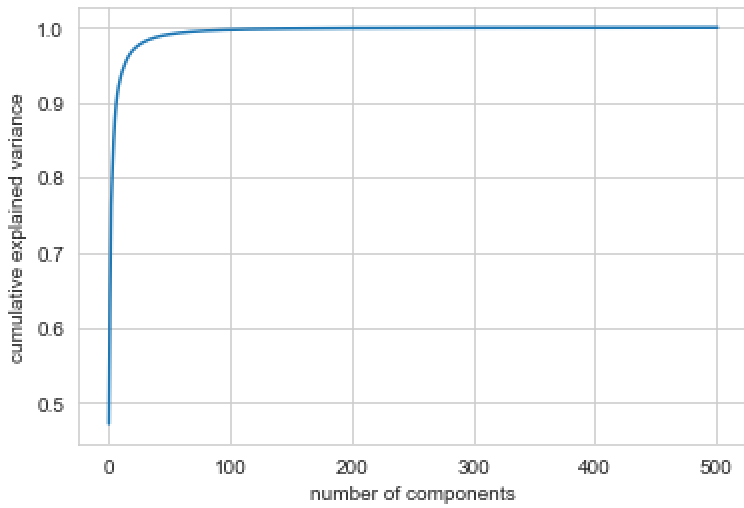
In [4]:
```
from sklearn.preprocessing import StandardScaler
features = df.values
sc = StandardScaler()
X_scaled = sc.fit_transform(features)
```

In [5]:
```
X_scaled.shape
```

Out[5]:
```
(41266, 502)
```

Determining optimal number of components for PCA looking at the explained variance as a function of the components

In [6]:
```
sns.set_style('whitegrid')
pca = PCA().fit(X_scaled)
plt.plot(np.cumsum(pca.explained_variance_ratio_))
plt.xlabel('number of components')
plt.ylabel('cumulative explained variance')
plt.show()
```



Here we see that we'd need about 100 components to retain 100% of the variance. Looking at this plot for a high-dimensional dataset can help us understand the level of redundancy present in multiple observations

Apply PCA to reduce the number of dimensions from 502 to 2 dimensions for better data visualization.

In [7]:
```
pca = PCA(n_components=2)
pca.fit(X_scaled)
print('explained variance :')
print('--------------------------------------------------------------------')
print(pca.explained_variance_)
print('--------------------------------------------------------------------')
print('PCA Components : ')
print('--------------------------------------------------------------------')
print(pca.components_)
print('--------------------------------------------------------------------')
X_transformed = pca.transform(X_scaled)
print('Transformed Feature values first five rows :')
print('--------------------------------------------------------------------')
print(X_transformed[:5,:])
print('--------------------------------------------------------------------')
print('Transformed Feature shape :')
print('--------------------------------------------------------------------')
print(X_transformed.shape)
print('--------------------------------------------------------------------')
print('Original Feature shape :')
print('--------------------------------------------------------------------')
print(X_scaled.shape)
print('--------------------------------------------------------------------')
print('Restransformed Feature shape :')
print('--------------------------------------------------------------------')
X_retransformed = pca.inverse_transform(X_transformed)
print(X_retransformed.shape)
print('--------------------------------------------------------------------')
print('Retransformed Feature values first five rows :')
print('--------------------------------------------------------------------')
print(X_retransformed[:5,:])
print('--------------------------------------------------------------------')
```

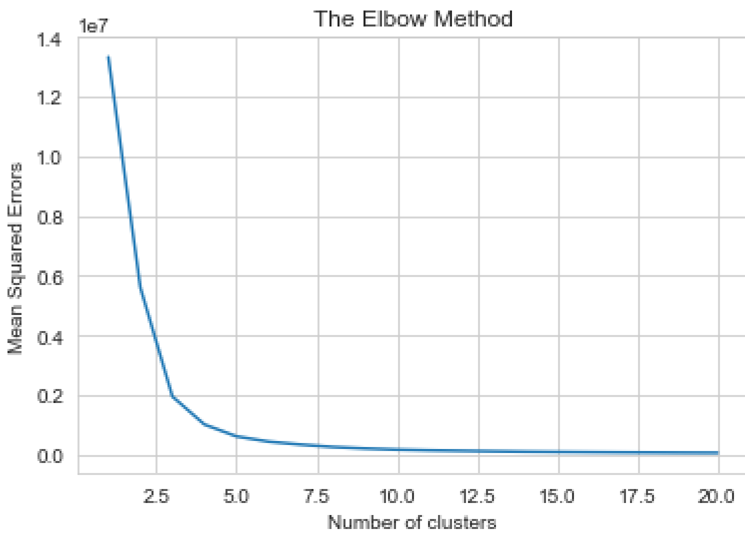
...

Problem 1:¶

There are various stocks for which we have collected a data set, which all stocks are apparently similar in performance

Finding optimum number of clusters for KMEANS cluster

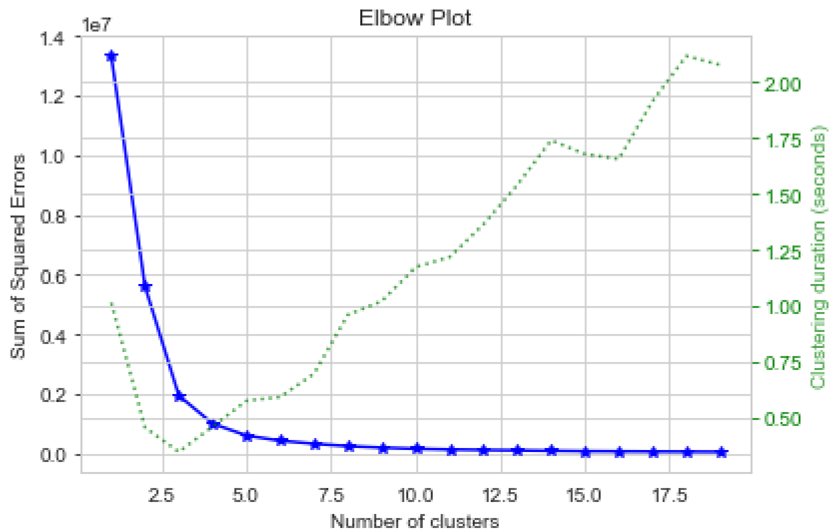
In [8]:
```
wcss=[]
for i in range(1, 21):
    kmeans = KMeans(n_clusters = i, init = 'k-means++', random_state = 0)
    kmeans.fit(X_transformed)
    wcss.append(kmeans.inertia_)
plt.plot(range(1, 21), wcss)
plt.title('The Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('Mean Squared Errors')
plt.figure(figsize=(10,8))
plt.show()
```




```
<Figure size 720x576 with 0 Axes>
```

In [9]:
```
import scikitplot
scikitplot.cluster.plot_elbow_curve(KMeans(),X_transformed,cluster_ranges=range(1
```

Out[9]:
```
<AxesSubplot:title={'center':'Elbow Plot'}, xlabel='Number of clusters', ylabel
='Sum of Squared Errors'>
```



Optimum number of cluster from the elbow method is determined to be 5

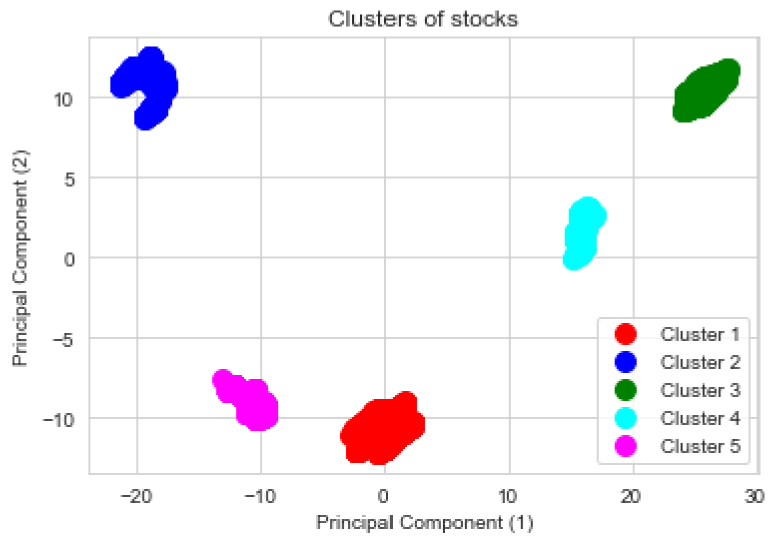
Applying K-Means Clustering to find stocks which are similar in performance

In [10]:
```
k_means = KMeans(n_clusters=5,random_state=0,init='k-means++')
k_means.fit(X_transformed)
y_kmeans = kmeans.fit_predict(X_transformed)
labels = k_means.labels_
```

In [11]:
```
len(labels)
```

Out[11]:
```
41266
```

In [12]:
```
plt.scatter(X_transformed[y_kmeans == 0, 0], X_transformed[y_kmeans == 0, 1], s =
plt.scatter(X_transformed[y_kmeans == 1, 0], X_transformed[y_kmeans == 1, 1], s =
plt.scatter(X_transformed[y_kmeans == 2, 0], X_transformed[y_kmeans == 2, 1], s =
plt.scatter(X_transformed[y_kmeans == 3, 0], X_transformed[y_kmeans == 3, 1], s =
plt.scatter(X_transformed[y_kmeans == 4, 0], X_transformed[y_kmeans == 4, 1], s =
#plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s = 30
plt.title('Clusters of stocks')
plt.xlabel('Principal Component (1)')
plt.ylabel('Principal Component (2)')
plt.legend()
plt.show()
```



The above 5 clusters shows the stocks which are similar in stock performance

Problem 2:

How many Unique patterns that exist in the historical stock data set, based on fluctuations in price.

In [13]:
```
df_comp = pd.DataFrame(pca.components_,columns=df.columns)
df_comp.head()
```

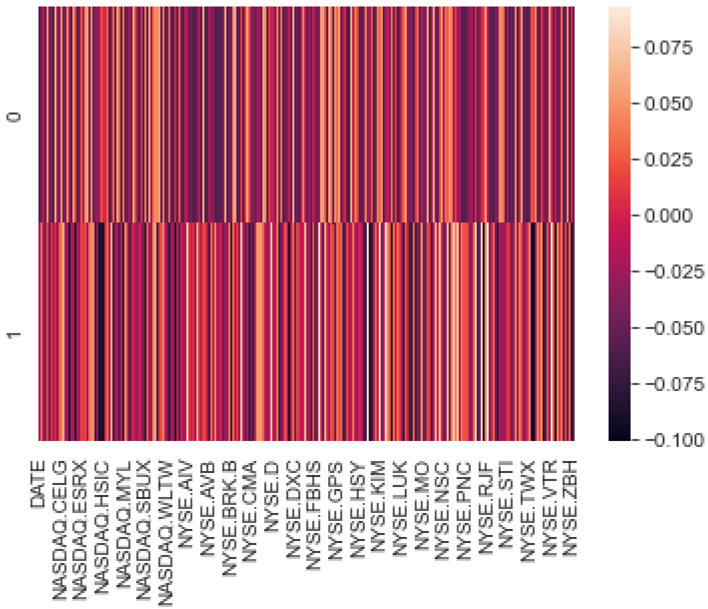
Out[13]:

	DATE	SP500	NASDAQ.AAL	NASDAQ.AAPL	NASDAQ.ADBE	NASDAQ.ADI	NASDAQ.ADP
0	-0.064116	-0.061006	-0.039128	-0.040896	-0.062662	-0.009756	-0.035746
1	0.013460	-0.017836	-0.064281	0.033885	0.001886	-0.032434	0.043464

2 rows × 502 columns

In [14]:
```
sns.heatmap(df_comp)
```

Out[14]: <AxesSubplot:>



Problem 3:

Identify which all stocks are moving together and which all stocks are different from each other.

In [15]:
```
df['labels'] = labels
```

In [30]:
```
df.labels.unique()
#df.value_counts
```

Out[30]:
```
array([1, 3, 4, 0, 2])
```

In [16]:
```
df.head()
```

Out[16]:

	DATE	SP500	NASDAQ.AAL	NASDAQ.AAPL	NASDAQ.ADBE	NASDAQ.ADI	NASDAQ.A
0	1491226200	2363.6101	42.3300	143.6800	129.6300	82.040	102.2
1	1491226260	2364.1001	42.3600	143.7000	130.3200	82.080	102.1
2	1491226320	2362.6799	42.3100	143.6901	130.2250	82.030	102.2
3	1491226380	2364.3101	42.3700	143.6400	130.0729	82.000	102.1
4	1491226440	2364.8501	42.5378	143.6600	129.8800	82.035	102.0

5 rows × 503 columns

In [17]:
```
df['labels'].unique().tolist()
```

Out[17]: [1, 3, 4, 0, 2]

In [18]:
```
for i in df['labels'].unique().tolist():
    count = df[df['labels'] == i].shape[0]
    print('\nFor lablel {} the number of similar stock performances is : {} '.for
```

```
For lablel 1 the number of similar stock performances is : 5872

For lablel 3 the number of similar stock performances is : 8624

For lablel 4 the number of similar stock performances is : 11162

For lablel 0 the number of similar stock performances is : 5870

For lablel 2 the number of similar stock performances is : 9738
```

In [19]:
```
from sklearn.cluster import SpectralClustering
hc = SpectralClustering(n_clusters = 5, affinity = 'nearest_neighbors')
hc.fit(X_transformed)
```

```
C:\Users\idofa\anaconda3\lib\site-packages\sklearn\manifold\_spectral_embeddin
g.py:236: UserWarning: Graph is not fully connected, spectral embedding may not
work as expected.
  warnings.warn("Graph is not fully connected, spectral embedding"
```

Out[19]: SpectralClustering(affinity='nearest_neighbors', n_clusters=5)

In [20]:
```
hc.fit_predict(X_transformed)
```

```
C:\Users\idofa\anaconda3\lib\site-packages\sklearn\manifold\_spectral_embeddin
g.py:236: UserWarning: Graph is not fully connected, spectral embedding may not
work as expected.
  warnings.warn("Graph is not fully connected, spectral embedding"
```

Out[20]: array([0, 0, 0, ..., 1, 1, 1])

In [21]:
```
y_labels = hc.labels_
```

In [22]:
```
len(y_labels),np.unique(y_labels)
```

Out[22]: (41266, array([0, 1, 2, 3, 4]))

In [23]:
```
# Visualising the clusters
X = X_transformed
plt.scatter(X[y_labels == 0, 0], X[y_labels == 0, 1], s = 100, c = 'red', label =
plt.scatter(X[y_labels == 1, 0], X[y_labels == 1, 1], s = 100, c = 'blue', label
plt.scatter(X[y_labels == 2, 0], X[y_labels == 2, 1], s = 100, c = 'green', label
plt.scatter(X[y_labels == 3, 0], X[y_labels == 3, 1], s = 100, c = 'cyan', label
plt.scatter(X[y_labels == 4, 0], X[y_labels == 4, 1], s = 100, c = 'magenta', lab
plt.title('Clusters of customers')
plt.xlabel('Annual Income (k$)')
plt.ylabel('Spending Score (1-100)')
plt.legend()
plt.show()
```

In [24]:
```
df2 = df.copy()
df2['labels'] = y_labels
for i in df2['labels'].unique().tolist():
    count = df2[df2['labels'] == i].shape[0]
    print('\nFor lablel {} the number of similar stock performances is : {} '.for
```

```
For lablel 0 the number of similar stock performances is : 5316

For lablel 1 the number of similar stock performances is : 24758

For lablel 2 the number of similar stock performances is : 921

For lablel 3 the number of similar stock performances is : 8479

For lablel 4 the number of similar stock performances is : 1792
```

For the given data set KMeans Clustering creates a better and distinct clustering compared to Spectral Clustering

In [ ]: