NC STATE UNIVERSITY

North Carolina State University

Department of Financial Mathematics

FIM 590 003 Machine Learning in Finance

Stock Classification with K-means and KNN

Author:
Haozhe Cui
Jinjia Peng
Jinyi Yang
Zhen He

October 26, 2024

FIM 590-003 Machine Learning in Finance

Project 02. Stock classification with K-means and KNN

Objective

1. Using market capitalization and price per earnings growth as metrics run clustering using K-means algorithm on companies traded in Russell 2000 index with 9 clusters. Compare the cluster positioning of your favorite companies with that of the Morningstar rating for those companies. Plot these clusters and see how these boundaries are matching with your expectations.

2. Assign labels to the clusters in the companies with their growth and capitalization based on K-means analysis or Morningstar analysis. Select a new company stock that you may be interested in and classify the stock using KNN technique.

Data Sourse

Data was collected from Bloomberg Terminal and Morningstar Investors

Data Processing

Import

In [67]:
```
import pandas as pd
import numpy as np
from sklearn.cluster import KMeans
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.neighbors import KNeighborsClassifier
import warnings
warnings.filterwarnings('ignore')
```

In [68]:
```
df = pd.read_csv('Final_data.csv')
```

Feature Scaling

In [69]:
```
# df.dropna(inplace=True)
# df['PEG'] = df['PEG'].apply(lambda x: np.log(x) if x > 0 else (-np.log(-x) if x < 0 else 0))
# df['Market Cap'] = df['Market Cap'].apply(lambda x: np.log(x) if x > 0 else(-np.log(-x) if x < (
```

To better visualize our results, we use log function to scale our features.

Data Analysis

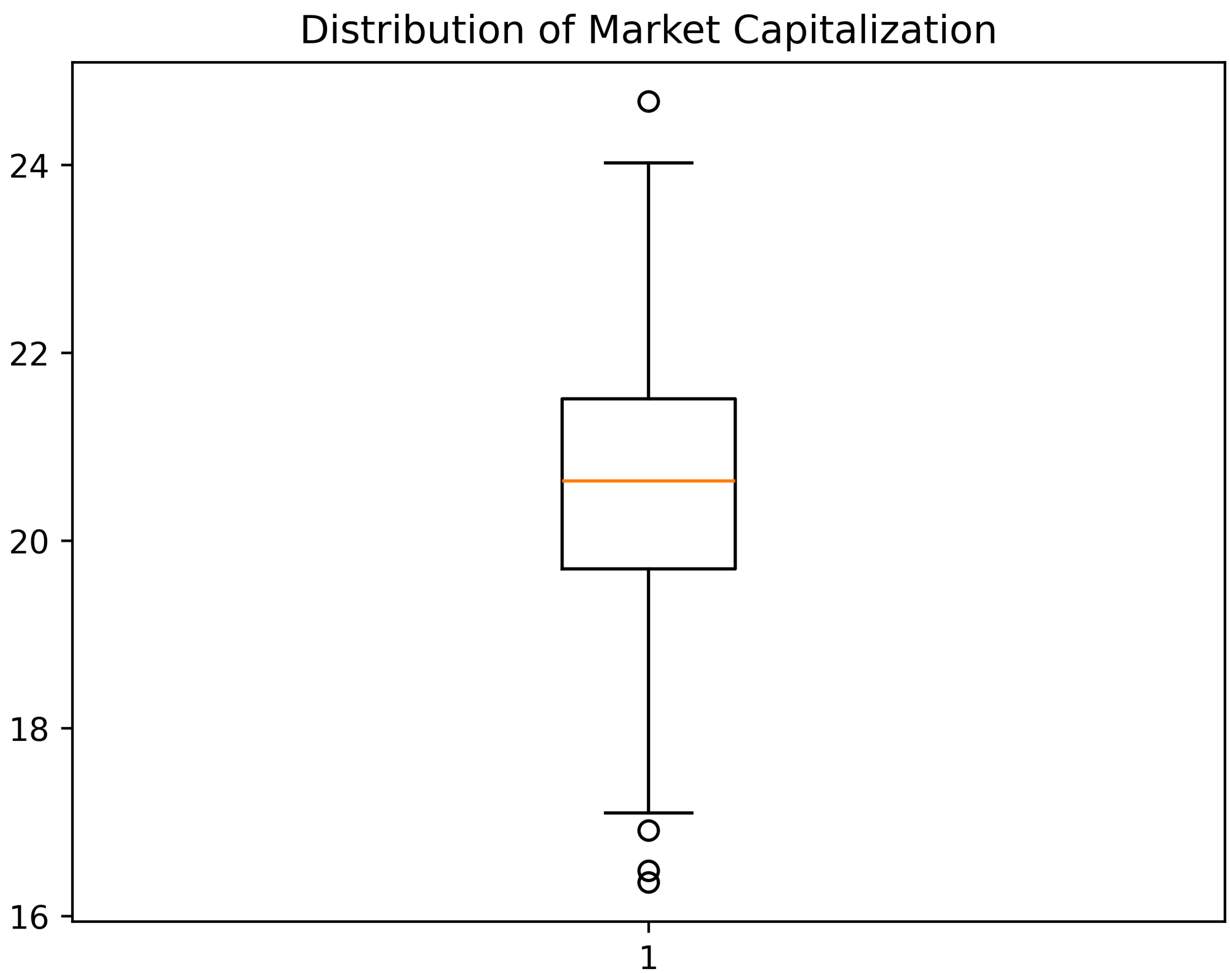
In [70]:
```
print(df[['PEG', 'Market Cap']].describe())
```

```
                PEG   Market Cap
count  1750.000000  1750.000000
mean     -0.059434    20.576889
std       4.309564     1.191997
min     -22.193954    16.360230
25%      -3.731173    19.697831
50%       0.141460    20.635041
75%       3.537406    21.509763
max      22.076735    24.677129
```

Checking outlier

In [71]:
```
#Creating a box gragh to check distribution
plt.figure(dpi=600)
plt.boxplot(df["Market Cap"])
plt.title("Distribution of Market Capitalization")

# fig = px.box(
#     data_frame=df_no_log,
#     x=["Market Cap"],
#     title="Distribution of Market Capitalization"
# )
# fig.update_layout(xaxis_title="Market Capitalization")
# offline.plot(fig, filename='interactive_line_plot.html', auto_open=False)
# fig.show()
```

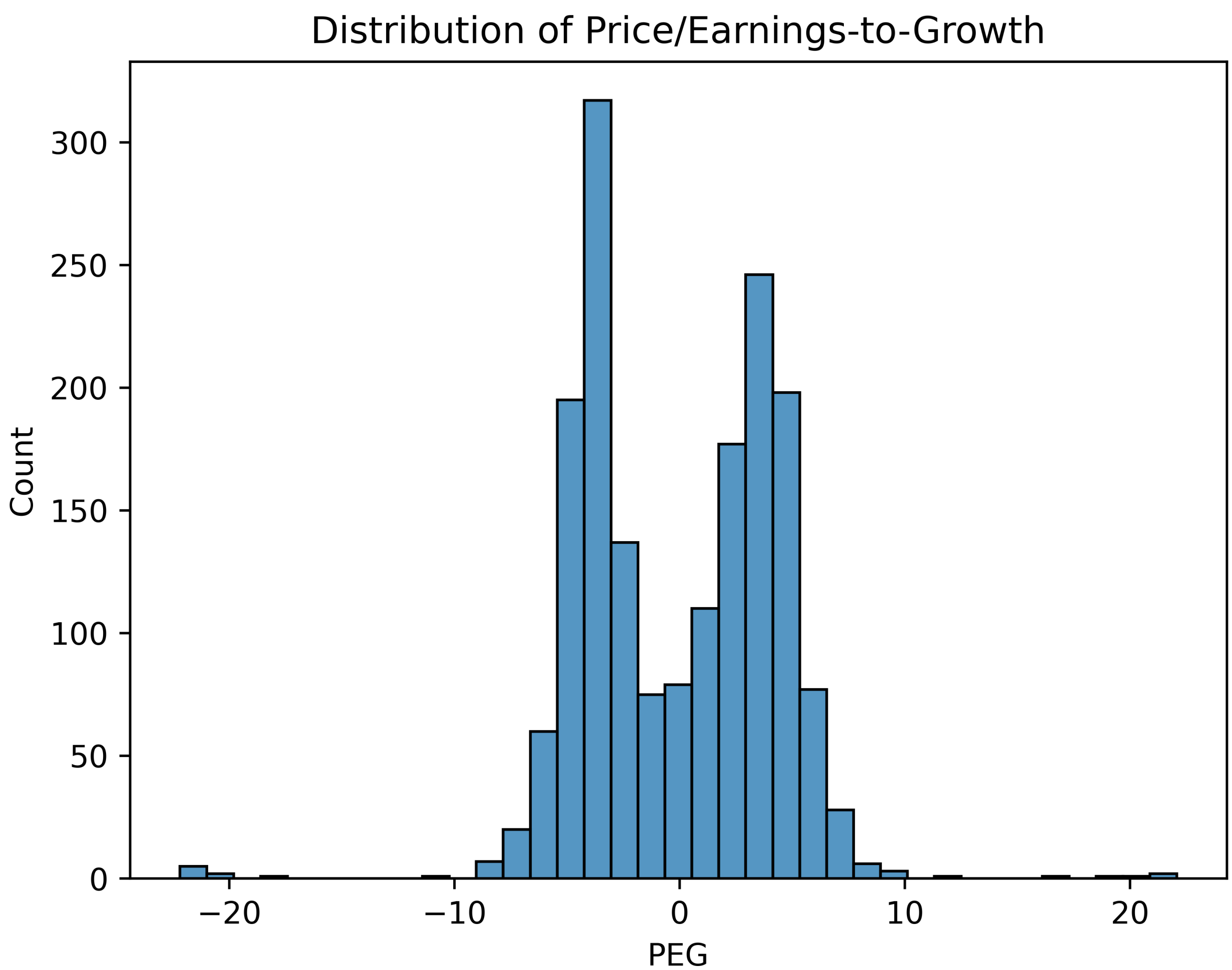
Out[71]:
```
Text(0.5, 1.0, 'Distribution of Market Capitalization')
```



The box plot shows how Market Capitalization is spread out with data points falling within the box area but with some notable outliers, towards the right side that belong to companies with very high market capitalizations. These outliers hint at a distribution. Could potentially affect the results of statistical analysis or machine learning models. To ensure an analysis and account, for these values better might involve adjusting the data or finding ways to lessen their impact.

In [72]:
```
#Creating a box gragh to check distribution
plt.figure(dpi=600)
sns.histplot(df["PEG"])
plt.title('Distribution of Price/Earnings-to-Growth')
# fig = px.box(
#     data_frame=df_no_log,
#     x=["PEG"],
#     title="Distribution of Price/Earnings-to-Growth"
# )
# fig.update_layout(xaxis_title="PEG")
# offline.plot(fig, filename='interactive_line_plot.html', auto_open=False)
# fig.show()
```

Out[72]:
```
Text(0.5, 1.0, 'Distribution of Price/Earnings-to-Growth')
```



Price/Earnings-to-Growth is centered around zero

K-means Clustering

Purpose

K-means is an unsupervised learning algorithm used for clustering data into groups based on feature similarity. Here we used PEG and Market Cap as two features and applied K-means to catch the pattern among stocks.

How it Works

1. Initialization: Choose the number of clusters n and randomly initialize n centroids. (n=9)
2. Assignment: Assign each data point to the nearest centroid based on the Euclidean distance.
3. Update: Recalculate the centroids by taking the mean of all data points assigned to each centroid.
4. Repeat: Repeat the assignment and update steps until the centroids no longer change significantly or a maximum number of iterations is reached.

Model

In [73]:
```
kmeans = KMeans(n_clusters = 9, random_state = 42)
kmeans.fit(df[['PEG','Market Cap']])
df['Cluster'] = kmeans.predict(df[['PEG','Market Cap']])
print(df)
```

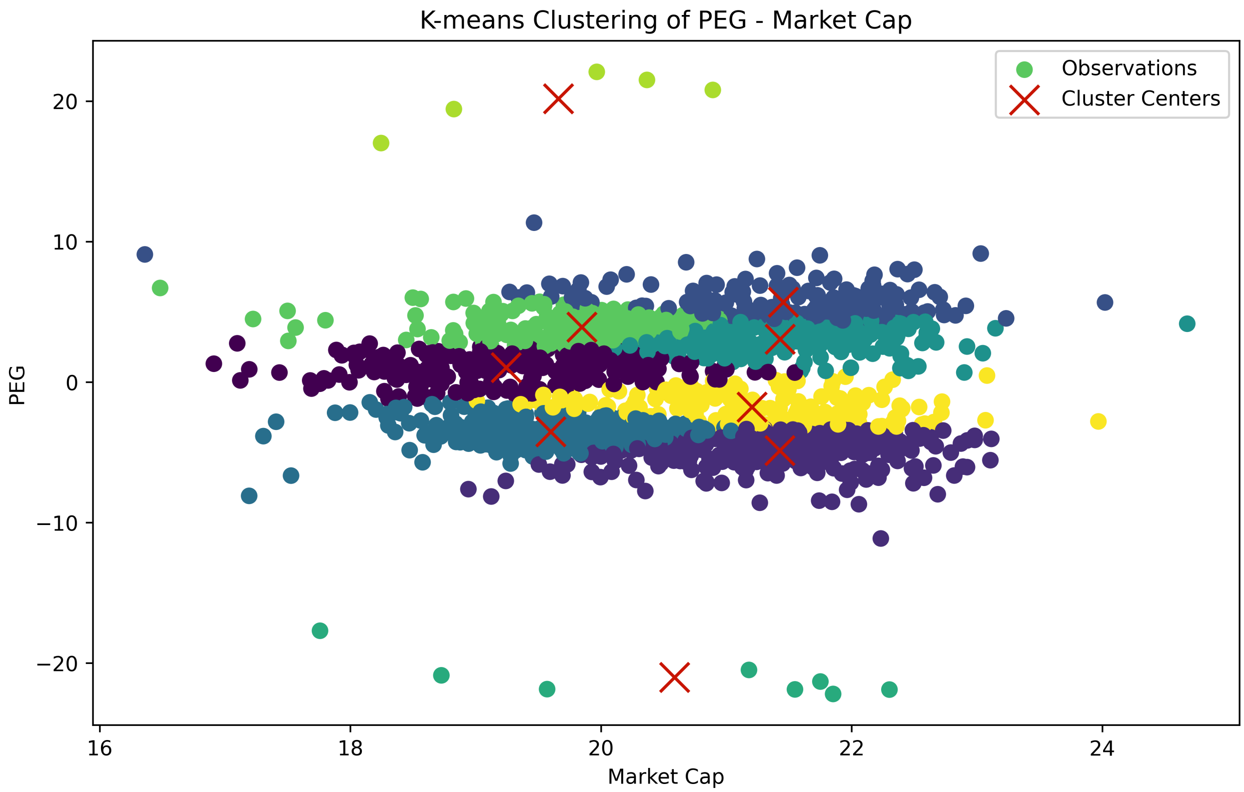
```
     Ticker       PEG  Market Cap  Cluster Stock Style Box
0      FLWS  4.558316   20.221569        6      Small Core
1      SRCE -7.175842   20.962696        1      Small Core
2      TSVT  0.805642   19.079673        0     Small Value
3     TWOUQ  6.697148   16.482134        6     Small Value
4      SCWO -3.672502   18.834407        3     Small Value
...     ...       ...         ...      ...             ...
1745    ZUO  3.458929   21.077243        4      Small Core
1746   ZURA  1.690996   19.367633        0             NaN
1747    ZWS  3.868356   22.339846        4    Small Growth
1748   ZYME  1.023755   20.203447        0      Small Core
1749   ZYXI -4.363376   19.474748        3      Small Core

[1750 rows x 5 columns]
```

In [74]:
```
# Calculate centers
df['Cluster'] = kmeans.labels_
centers = kmeans.cluster_centers_
print("Clustrer Centers", centers)
```

```
Clustrer Centers [[  1.04252814  19.24307574]
 [ -4.87466528  21.42677949]
 [  5.69518292  21.45390259]
 [ -3.53878168  19.59858646]
 [  3.05679313  21.42902354]
 [-21.01951892  20.5855837 ]
 [  3.90903094  19.8459462 ]
 [ 20.16138     19.65901243]
 [ -1.7786558   21.20471676]]
```

Visualization of K-maens

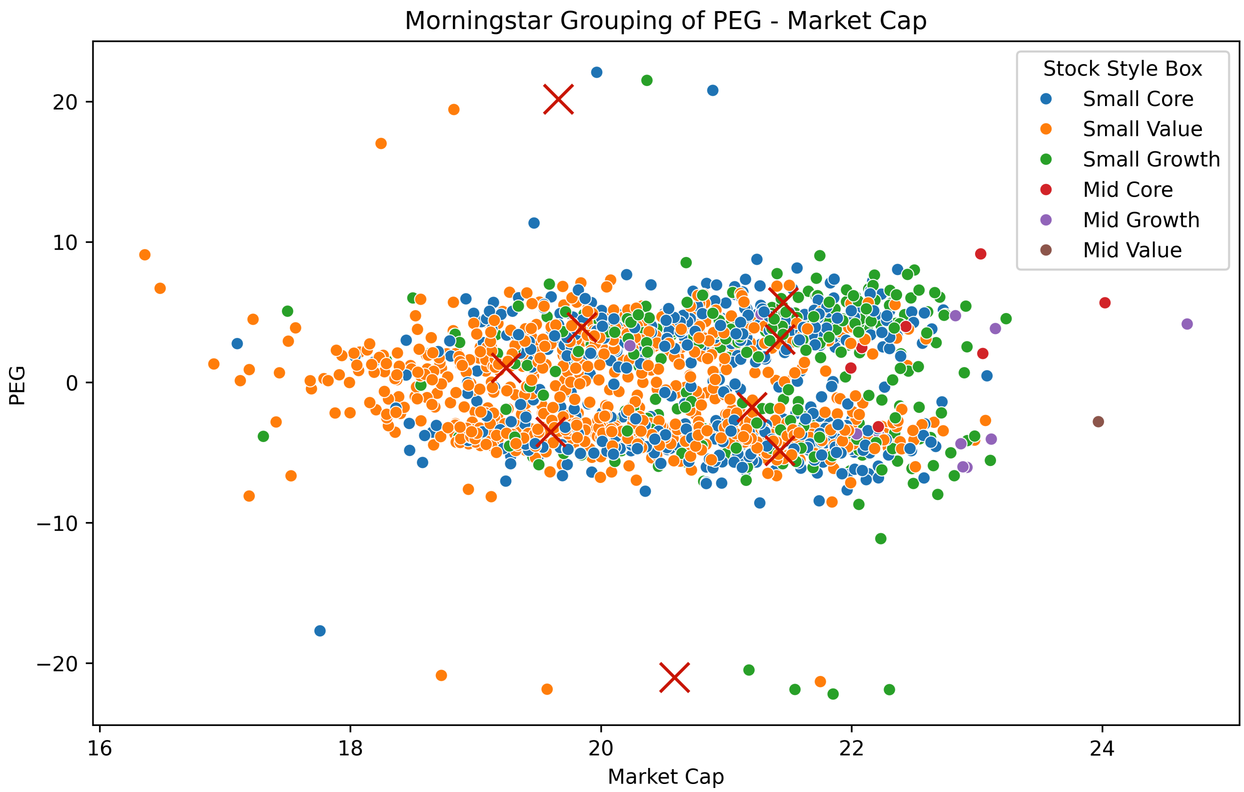
In [75]:
```
plt.figure(dpi=600, figsize=(10, 6))
plt.scatter(df['Market Cap'], df['PEG'], c=df['Cluster'], cmap='viridis', marker='o', s=50, label
plt.scatter(centers[:, 1], centers[:, 0], c='#CC1400', marker='x', s=200, label='Cluster Centers'
plt.xlabel('Market Cap')
plt.ylabel('PEG')
plt.title('K-means Clustering of PEG - Market Cap')
plt.legend()
# plt.savefig('K-means Clustering of PEG - Market Cap')
plt.show()
```



This graph shows the relationship reflected through the K-Means algorithm. The X-axis is the PEG and the Y-axis is the Market Cap. The colors represent each cluster, and the red X represents the centroid of each cluster. It is clear that the data in a same cluster are around a same near centroid.

Morningstar Stock Style Box

In [76]:
```
plt.figure(dpi=600, figsize=(10, 6))
sns.scatterplot(data=df, x='Market Cap', y='PEG', hue='Stock Style Box')
plt.scatter(centers[:, 1], centers[:, 0], c='#CC1400', marker='x', s=200, label='Cluster Centers'
plt.xlabel('Market Cap')
plt.ylabel('PEG')
plt.title('Morningstar Grouping of PEG - Market Cap')
# plt.savefig('Morningstar Grouping of PEG - Market Cap.png')
plt.show()
```



In this graph, the colored dots represent the specific MorningStar Rating of each company, and X represests the centroid of each cluster from the K-Means alrigothm. Observe that clearly, the centroids do not align with the groups of the data classified by the MorningStar Rating, meaning that the clusters from the K-means method does not show significant relationship with the MorningStar Rating classification.

K-Nearest Neighbors (KNN)

Purpose

KNN is a supervised learning algorithm used for classification and regression tasks. Here we used PEG and Market Cap as two features and used labels generated from K-means clustering to train the KNN model, then used this model to predict the label of a new stock.

How it Works

1. Training Phase: KNN does not have a traditional training phase. Instead, it stores the entire training dataset.
2. Prediction: For a new data point, KNN calculates the distance (commonly Euclidean) to all points in the training dataset.
3. Neighbor Selection: It selects the KK nearest neighbors based on the calculated distances.
4. Voting (for classification): The algorithm assigns the class label that is most common among the KK neighbors.
5. Averaging (for regression): For regression tasks, it averages the values of the KK nearest neighbors.

Model

In [77]:
```
X = df[['Market Cap','PEG']]
y = df['Cluster']
```

```
knn = KNeighborsClassifier(n_neighbors=9)
knn.fit(X,y)
```

Out[77]:

```
KNeighborsClassifier
KNeighborsClassifier(n_neighbors=9)
```

New Stock Analysis

We found a new stock TURN with Market Capitailization 33.7 Mil and PEG 2.08

In [78]:

```
stock = {'Market Cap': 17.333,
 'PEG': 0.736
 }

knn.predict([[17.333, 0.736]])
```

Out[78]:

```
array([0], dtype=int32)
```

The new stock belongs to cluster 0