

**DEPARTMENT OF COMPUTER TECHNOLOGY,
ANNA UNIVERSITY, MIT CAMPUS CHENNAI 600 044**

**STOCK PRICE PREDICTION USING DECISION TREE
WITH CLUSTERING TECHNIQUE**

A PROJECT REPORT

Submitted by

**R.SASANA (2020503542)
P.RAKSHITHA (2020503024)
F.JOEVITA FAUSTINA DOSS (2020503013)**

Submitted to

MS.R.KEERTHANA

CS6003 - BIG DATA ANALYTICS

DATE: 15.05.2023

Abstract :

This project aims to predict the direction of stock price movement for various stocks in the future months using machine learning techniques. To improve prediction precision, the project clusters time periods based on macroeconomic factors and builds decision trees for each cluster using idiosyncratic factors for each stock. Monthly data from Jan. 01 1990 to Sep. 01 2018 is used for the analysis, and the project focuses on examples of Apple and ATT stocks. By taking the macroeconomic environment into consideration, the project aims to provide a more accurate prediction of stock price movements, as stock prices may behave differently during different periods of business cycles.

Dataset :

The dataset used for the above stock price prediction consists of monthly macroeconomic data such as inflation rate, unemployment rate, rate of change of Dow Jones Industrial Average and rate of change of S&P 500, as well as stock-specific data such as three-month moving average of stock price, two month moving average of stock price, stock price in current month and current fundamental value of the company. The data covers the period from Jan. 01 1990 to Sep. 01 2018 and was scraped from the web.

Modules :

Module 3 - Clustering

K-means is an unsupervised machine learning algorithm that partitions a dataset into K clusters based on similarity between the data points. The algorithm works by first randomly selecting K centroids, then assigning each data point to the nearest centroid, and finally updating the centroids based on the mean of the data points in each cluster. This process is repeated until convergence, meaning the centroids no longer change significantly between iterations.

Module 4 - Classification

A decision tree is a type of supervised machine learning algorithm used for classification and regression analysis. It is a tree-like model where each internal node represents a test on a feature, each branch represents the outcome of the test, and each leaf node represents a class label or a continuous value. The decision tree algorithm works by recursively splitting the data into smaller subsets based on the most significant feature until a stopping criterion is met, such as a maximum depth or a minimum number of samples per leaf node.

Module 10 - Visualisation

Visualization is the process of representing data or information in a graphical or pictorial format that can be easily understood and interpreted by humans. In the context of data analysis and machine learning, visualization is often used to explore and understand the patterns and

relationships in large and complex datasets. Visualization tools can be used to create various types of graphs, charts, maps, and other visual representations of data.

Methodology:

We propose a KMDT (K Means clustering with decision tree) technique for stock price prediction that involves constructing the decision tree for each clusters and compared its correctness.

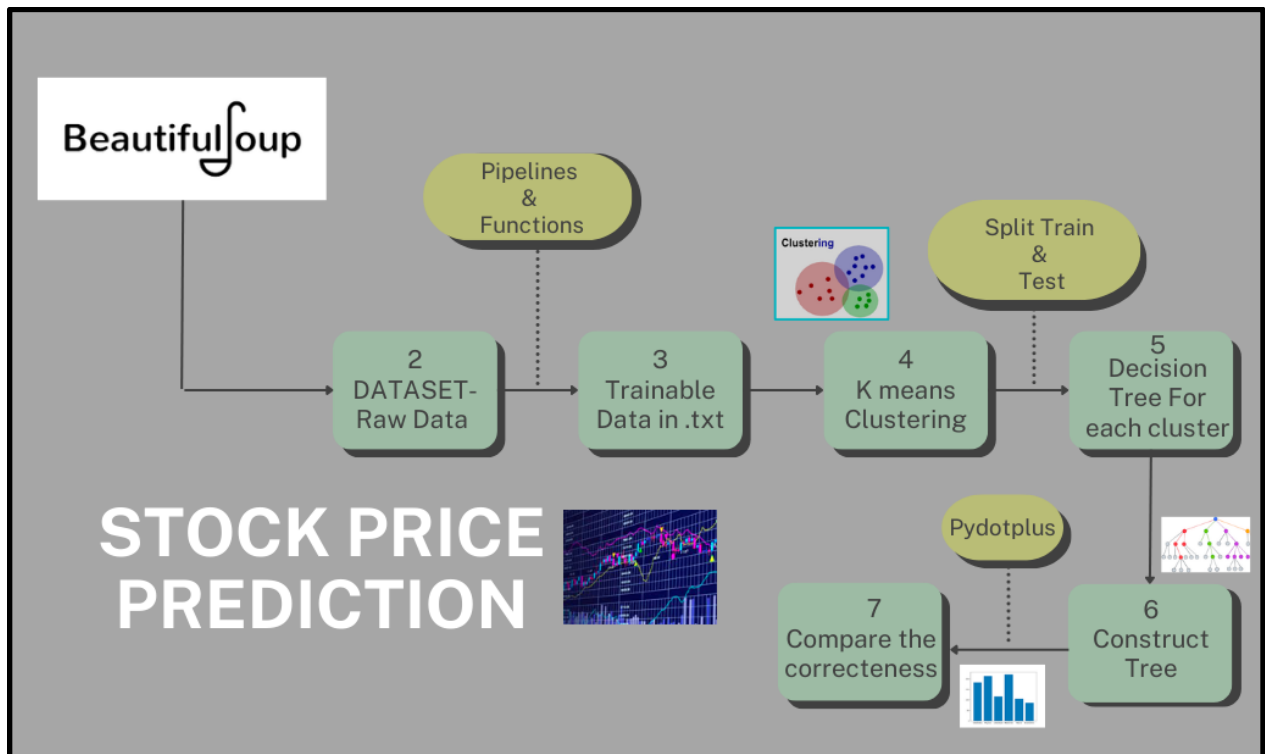


Fig .1 Architecture diagram of Stock price prediction using KMDT Technique

- We utilized Beautiful Soup to scrape the experimental data from the web. After that, we developed pipelines and functions to convert the raw data into trainable data, which were then fed into the model. Apple and ATT were the two stocks used as examples.
- In this project, we utilized K-means clustering to segment the time periods into different clusters based on macroeconomic factors, such as inflation rate, unemployment rate, rate of change of Dow Jones Industrial Average, and rate of change of S&P 500. The reason for using K-means clustering was to identify distinct periods in the business cycle, which can have a significant impact on stock prices.

- For the stock price prediction project, decision trees are utilized as a classification method to predict the direction of stock price movement in the future months. Decision trees are built for each cluster that are identified using the K-means clustering algorithm. The decision trees were constructed using idiosyncratic factors for each stock, which included the three-month moving average of stock price, two-month moving average of stock price, stock price in the current month, and current fundamental value of the company.
- Pydotplus is a Python library that is used for graph visualization. In the above stock prediction project, Pydotplus was used to visualize decision trees. Decision trees are often difficult to understand in their raw form. Pydotplus provides a way to represent the decision tree in a more intuitive and visual way. To use Pydotplus, the decision tree is first created using a machine learning algorithm such as the decision tree classifier. Then, the decision tree is converted into a graph format using Pydotplus.

Code:

getdata.py:

```
import numpy as np
import sys
```

```
def getStockPrice(fileName):
```

```
    price = []
```

```
    size = []
```

```
    with open(fileName) as f:
```

```
        if fileName == 'data/apple':
```

```
            for line in f:
```

```
                item = line.rstrip().split(',')
```

```
                if len(item) >= 2:
```

```
                    price.append(float(item[-2]))
```

```
                    size.append(float(item[-1]))
```

```
        else:
```

```
            for line in f:
```

```
                item = line.rstrip().split(',')
```

```
                if len(item) >= 2:
```

```
                    price.append(float(item[-2]))
```

```
                    size.append(float(item[-1]))
```

```
    return price, size[2:-1]
```

```
def getInputData(fileName):
```

```
    fileName = 'data/' + fileName
```

```
    price, size = getStockPrice(fileName)
```

```
    threeMonthMA = [(i+j+k)/3 for i,j,k in zip(price, price[1: ], price[2: ])]
```

```
    del threeMonthMA[-1]
```

```

twoMonthMA = [(i+j)/2 for i,j in zip(price[1: ], price[2: ])]
del twoMonthMA[-1]
stockReturn = [(j-i)/i for i,j in zip(price[2: ], price[3: ])]
classification = list(map(int, [i>0 for i in stockReturn]))
inputData = np.array([price[2:-1], twoMonthMA, threeMonthMA, size, classification],
dtype=object).T
fileName = fileName + 'TrainData.txt'
np.savetxt(fileName, inputData)
return inputData

if __name__ == '__main__':
    unemployment = []
    with open("data/unemployment", 'r') as f:
        for line in f:
            unemployment.append(list(map(float, line.rstrip().rstrip('\n').split(' '))))
    unemployment = [item for sublist in unemployment for item in sublist]

    inflation = []
    with open("data/inflation", 'r') as f:
        for line in f:
            inflation.append(list(map(float, line.rstrip().rstrip('\n').split(' '))))
    inflation = [item for sublist in inflation for item in sublist]

    djia = []
    with open("data/djia", 'r') as f:
        for line in f:
            djia.append(list(map(float, line.rstrip().rstrip('\n').split(' '))))
    djia = [(j-i)/i for i,j in zip(djia, djia[1:])]

    sp = []
    with open('data/sp', 'r') as f:
        for line in f:
            item = line.rstrip().rstrip('\n').split(' ')
            sp.append(list(map(float, item)))
    sp = [(j-i)/i for i,j in zip(sp, sp[1:])]

    clusterData = np.array([unemployment, inflation, djia, sp]).T
    max_len = max(len(lst) for lst in [unemployment, inflation, djia, sp])
    clusterData = np.array([lst + [0]*(max_len - len(lst)) for lst in [unemployment, inflation, djia,
sp]]).T

    np.savetxt('data/clusterData.txt', clusterData)

    fileName = 'apple'

```

```
getInputData(fileName)
getInputData('att')
```

stockprediction.ipynb

```
%run getData.py
```

```
import numpy as np
from sklearn import preprocessing, cluster, model_selection, svm
import matplotlib.pyplot as plt
```

```
class Classifier(object):
    def __init__(self, microDataLoc, clusterNum=3, macroDataLoc="data/clusterData.txt"):
        self.microDataLoc = microDataLoc
        self.macroDataLoc = macroDataLoc
        self.clusterNum = clusterNum

    def cluster(self):
        data = np.loadtxt(self.macroDataLoc)
        cleanData = preprocessing.scale(data)
        kmeans = cluster.KMeans(n_clusters=self.clusterNum, random_state=11).fit(cleanData)
        groupNum = np.array(kmeans.labels_)
        return groupNum

    def prepareData(self):
        data = np.loadtxt(self.microDataLoc)
        groupNum = self.cluster()
        group, label = [], []
        for i in range(self.clusterNum):
            group.append(data[groupNum==i, :-1])
            label.append(data[groupNum==i, -1])
        return (group, label)

    def trainTestSplit(self):
        train, test, trainLabel, testLabel = [], [], [], []
        group, label = self.prepareData()
        for i in range(self.clusterNum):
            trainData,          testData,          trainLabelData,          testLabelData          =
model_selection.train_test_split(group[i],
                                label[i], test_size=0.3, random_state=11)
            train.append(trainData)
            test.append(testData)
            trainLabel.append(trainLabelData)
            testLabel.append(testLabelData)
        return (train, test, trainLabel, testLabel)
```

```

def train(self):
    pass

def test(self):
    clf, test, testLabel = self.train()
    error = []
    for i in range(self.clusterNum):
        pred = (clf[i].predict(test[i]) == 1)
        caseError = sum([i != j for (i,j) in zip(testLabel[i], pred)])
        error.append(float(caseError)/len(pred))
    return error

def reportResult(self):
    error = self.test()
    for i in range(self.clusterNum):
        print("Group no:" + str(i+1) + "\nincorrect rate")
        print(1-error[i])

    x_labels = [str(i+1) for i in range(self.clusterNum)]
    plt.bar(x_labels, 1-np.array(error))
    plt.xlabel("Group Number")
    plt.ylabel("Correct Rate")
    plt.title("Performance of Decision Tree with Clustering")
    plt.ylim([0, 1])

    plt.savefig("decision_tree_clustering.png")
    plt.show()
    print('\n')
    return error

from sklearn import tree
from sklearn.tree import export_graphviz
import numpy as np
import pydotplus

class DecisionTreeStockPrediction(Classifier):

    def __init__(self, microDataLoc, clusterNum=3, macroDataLoc="data/clusterData.txt"):
        Classifier.__init__(self, microDataLoc, clusterNum, macroDataLoc)

    def visualize(self, clf):
        dot_data = export_graphviz(clf, out_file=None, feature_names=self.features,
        class_names=self.labels, filled=True, rounded=True, special_characters=True)

```

```

graph = pydotplus.graph_from_dot_data(dot_data)
graph.write_pdf("decision_tree.pdf")
return graph

def train(self):
    train, test, trainLabel, testLabel = self.trainTestSplit()
    clf = [tree.DecisionTreeClassifier() for i in range(self.clusterNum)]
    for i in range(self.clusterNum):
        clf[i].fit(train[i], trainLabel[i])
        self.features = [str(j+1) for j in range(train[i].shape[1])]
        self.labels = [str(k) for k in np.unique(trainLabel[i])]
        self.visualize(clf[i])
    return (clf, test, testLabel)

if __name__ == "__main__":
    print("Cluster no:1\n")
    # Case when 1 cluster
    apple = DecisionTreeStockPrediction("data/appleTrainData.txt", clusterNum=1)
    apple.reportResult()


    # Case when 2 clusters
    print("Cluster no:2\n")
    apple = DecisionTreeStockPrediction("data/appleTrainData.txt", clusterNum=2)
    apple.reportResult()

    # for the case when cluster = 3
    print("Cluster no:3\n")
    apple = DecisionTreeStockPrediction("data/appleTrainData.txt", clusterNum=3)
    apple.reportResult()



    print("Cluster no:4\n")
    # Case when 4 clusters
    apple = DecisionTreeStockPrediction("data/appleTrainData.txt", clusterNum=4)
    apple.reportResult()
    clf, _, _ = apple.train()
    apple.visualize(clf[0])

```


Dataset:



1989-11-01,1.651786,1.687500,1.517857,1.580357,0.135998,631234800
1989-12-01,1.589286,1.633929,1.160714,1.258929,0.116345,1485296400
1990-01-01,1.258929,1.383929,1.147321,1.214286,0.112219,936874400
1990-02-01,1.232143,1.258929,1.151786,1.214286,0.112219,620972800
1990-03-01,1.196429,1.549107,1.187500,1.437500,0.145975,1139118400
1990-04-01,1.428571,1.580357,1.361607,1.406250,0.142802,910618800
1990-05-01,1.419643,1.526786,1.383929,1.473214,0.149602,1028907600
1990-06-01,1.477679,1.602679,1.339286,1.598214,0.175927,988013600
1990-07-01,1.589286,1.705357,1.428571,1.500000,0.165116,1040953200
1990-08-01,1.500000,1.562500,1.196429,1.321429,0.145459,782084800
1990-09-01,1.303571,1.339286,0.973214,1.035714,0.124516,551737200
1990-10-01,1.053571,1.142857,0.866071,1.098214,0.132029,1264029200
1990-11-01,1.089286,1.375000,1.062500,1.312500,0.157791,813652000
1990-12-01,1.330357,1.616071,1.321429,1.535714,0.203632,1073523200
1991-01-01,1.526786,2.000000,1.500000,1.982143,0.262828,1759326800
1991-02-01,1.982143,2.223214,1.955357,2.044643,0.271115,1386445200
1991-03-01,2.035714,2.508929,2.035714,2.428571,0.342147,1302649600
1991-04-01,2.428571,2.616071,1.946429,1.964286,0.276737,1810762800
1991-05-01,1.714286,1.919643,1.571429,1.678571,0.236484,1993210800
1991-06-01,1.678571,1.767857,1.437500,1.482143,0.224888,1022915600
1991-07-01,1.508929,1.723214,1.491071,1.651786,0.250628,945873600
1991-08-01,1.642857,1.982143,1.633929,1.892857,0.287206,1043162400
1991-09-01,1.883929,1.910714,1.660714,1.767857,0.286305,598267600
1991-10-01,1.758929,2.008929,1.660714,1.839286,0.297873,898497600
1991-11-01,1.830357,1.973214,1.696429,1.812500,0.293535,924008400
1991-12-01,1.812500,2.071429,1.732143,2.103393,0.349560,651792400
1992-01-01,1.991071,2.464286,1.982143,2.312500,0.401490,1175790000
1992-02-01,2.312500,2.500000,2.205357,2.410714,0.418542,633749200
1992-03-01,2.419643,2.446429,2.062500,2.080357,0.381117,746415600
1992-04-01,2.044643,2.187500,1.892857,2.147321,0.393385,982847600

  appleTrainData.txt

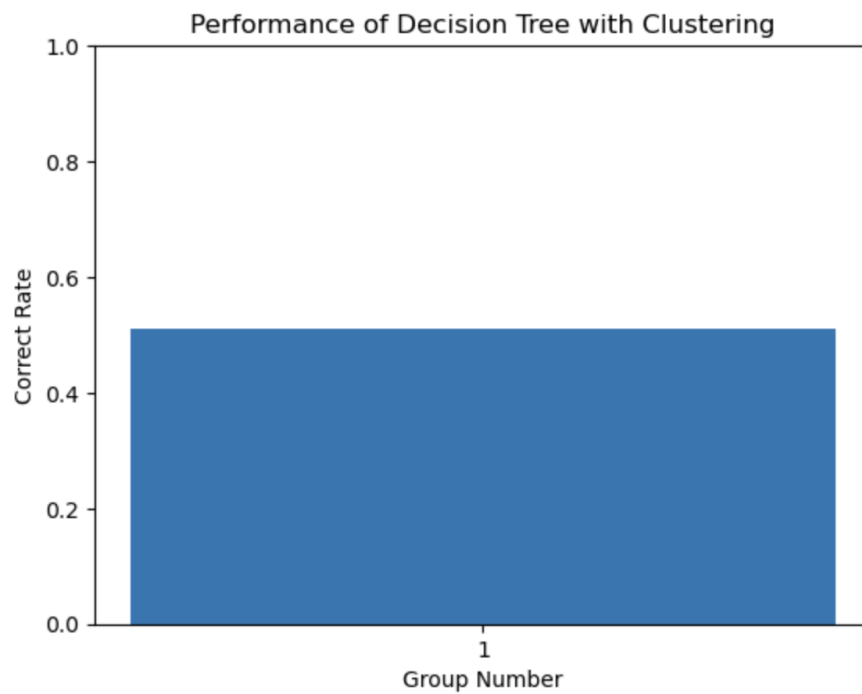
```
1.12218999999999993e-01 1.14281999999999947e-01 1.21520666666666657e-01 9.36874400000000000e+08
0.00000000000000000e+00
1.12218999999999993e-01 1.12218999999999993e-01 1.13594333333333338e-01 6.20972800000000000e+08
1.00000000000000000e+00
1.459749999999999936e-01 1.290969999999999895e-01 1.23470999999999974e-01 1.13911840000000000e+09
0.00000000000000000e+00
1.428020000000000123e-01 1.443885000000000030e-01 1.336653333333333304e-01 9.10618800000000000e+08
1.00000000000000000e+00
1.496020000000000127e-01 1.46201999999999986e-01 1.461263333333333303e-01 1.02890760000000000e+09
1.00000000000000000e+00
1.759270000000000000e-01 1.627645000000000064e-01 1.561103333333333232e-01 9.88013600000000000e+08
0.00000000000000000e+00
1.651160000000000128e-01 1.70521499999999925e-01 1.635483333333333233e-01 1.04095320000000000e+09
0.00000000000000000e+00
1.454590000000000050e-01 1.5528750000000000227e-01 1.621673333333333300e-01 7.82084800000000000e+08
0.00000000000000000e+00
1.245160000000000017e-01 1.3498750000000000103e-01 1.450303333333333344e-01 5.51737200000000000e+08
1.00000000000000000e+00
1.320290000000000075e-01 1.2827250000000000115e-01 1.340013333333333334e-01 1.26402920000000000e+09
1.00000000000000000e+00
1.577909999999999868e-01 1.449099999999999833e-01 1.3811200000000000125e-01 8.13652000000000000e+08
1.00000000000000000e+00
2.036320000000000074e-01 1.80711499999999971e-01 1.64483999999999913e-01 1.02352320000000000e+09
1.00000000000000000e+00
2.628280000000000061e-01 2.33229999999999929e-01 2.080836666666666668e-01 1.75932680000000000e+09
1.00000000000000000e+00
2.711149999999999949e-01 2.6697150000000000282e-01 2.458583333333333176e-01 1.38644520000000000e+09
1.00000000000000000e+00
3.421469999999999789e-01 3.066309999999999869e-01 2.9203000000000000118e-01 1.30264960000000000e+09
0.00000000000000000e+00
```

Cluster no:1

Group no:1

correct rate

0.5096153846153846



Cluster no:2

Group no:1

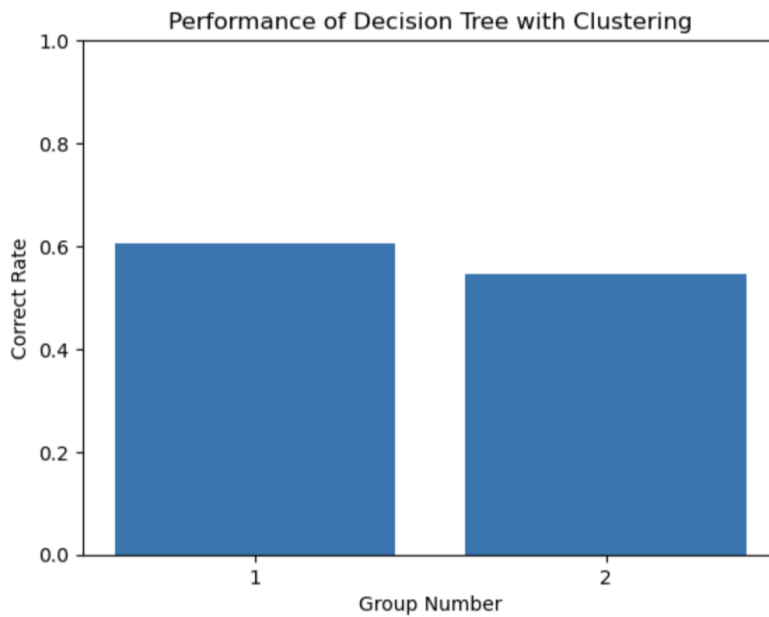
correct rate

0.6056338028169015

Group no:2

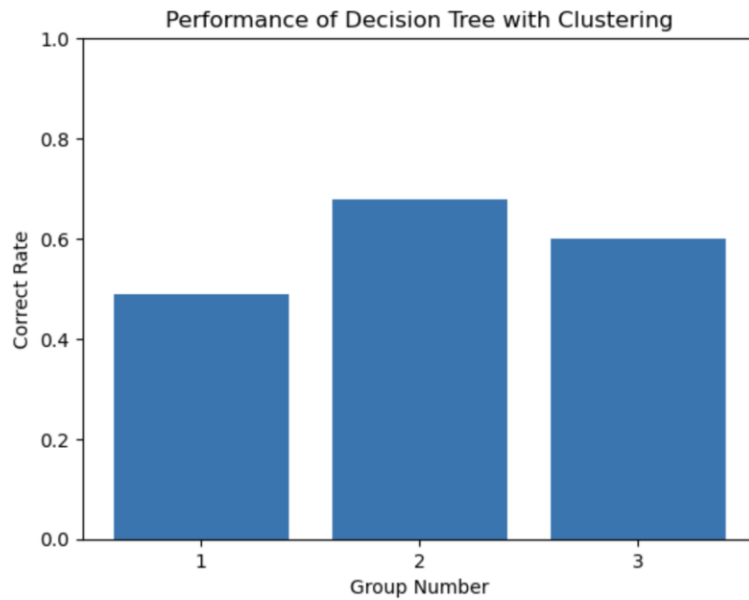
correct rate

0.5454545454545454



Cluster no:3

Group no:1
correct rate
0.4901960784313726
Group no:2
correct rate
0.6785714285714286
Group no:3
correct rate
0.6



Cluster no:4

Group no:1
correct rate
0.6363636363636364
Group no:2
correct rate
0.5652173913043479
Group no:3
correct rate
0.39473684210526316
Group no:4
correct rate
0.5909090909090908





Conclusion :

In conclusion, this project aims to predict the direction of stock price movement in the future months using machine learning techniques and big data analytics. The project first clusters time periods into 3-4 clusters based on macroeconomic factors, then builds decision trees with idiosyncratic factors for each stock to catch the trend of stock price movement. The chosen machine learning technique takes the macroeconomic environment into consideration when predicting stock prices, which can improve prediction precision since stock prices may behave differently during different periods of business cycles.