Clustered RL4T Report

January 3, 2020

[11]:
```
import pandas as pd
import ta
import sklearn.cluster as cluster
import numpy as np
import matplotlib.pyplot as plt
```

[12]:
```
df = pd.read_csv("GOOG.csv", sep=',')
df.head()
```

[12]:

	Date	Open	High	Low	Close	Adj Close	Volume
0	2004-08-19	49.813286	51.835709	47.800831	49.982655	49.982655	44871300
1	2004-08-20	50.316402	54.336334	50.062355	53.952770	53.952770	22942800
2	2004-08-23	55.168217	56.528118	54.321388	54.495735	54.495735	18342800
3	2004-08-24	55.412300	55.591629	51.591621	52.239193	52.239193	15319700
4	2004-08-25	52.284027	53.798351	51.746044	52.802086	52.802086	9232100

[13]:
```
df = ta.add_all_ta_features(df, open="Open", high="High",
                                low="Low", close="Close", volume="Volume")
```

```
C:\Users\John\Anaconda3\envs\ds35\lib\site-
packages\numpy\core\fromnumeric.py:83: RuntimeWarning: invalid value encountered
in reduce
  return ufunc.reduce(obj, axis, dtype, out, **passkwargs)
C:\Users\John\Anaconda3\envs\ds35\lib\site-packages\ta\trend.py:543:
RuntimeWarning: invalid value encountered in double_scalars
  dip[i] = 100 * (self._dip[i]/self._trs[i])
C:\Users\John\Anaconda3\envs\ds35\lib\site-packages\ta\trend.py:547:
RuntimeWarning: invalid value encountered in double_scalars
  din[i] = 100 * (self._din[i]/self._trs[i])
```

[ ]:
```
'''
    Adding the technical indicators.
'''
```

[23]:
```
df.head()
```

[23]:

	Date	Open	High	Low	Close	Adj Close	\
0	2004-08-19	49.813286	51.835709	47.800831	49.982655	49.982655	
1	2004-08-20	50.316402	54.336334	50.062355	53.952770	53.952770	

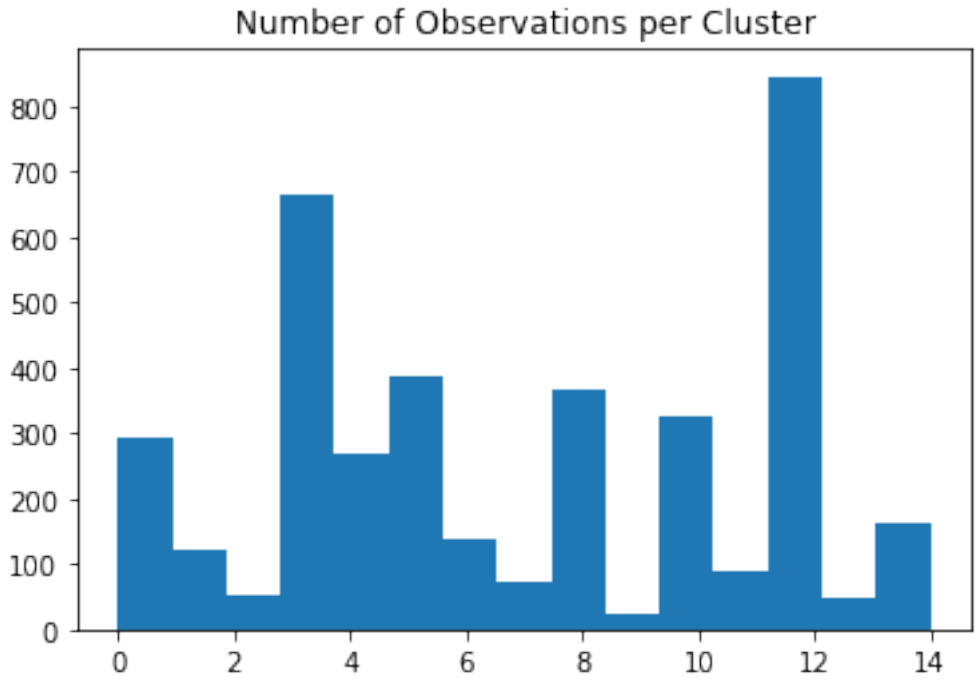
```
2  2004-08-23  55.168217  56.528118  54.321388  54.495735  54.495735
3  2004-08-24  55.412300  55.591629  51.591621  52.239193  52.239193
4  2004-08-25  52.284027  53.798351  51.746044  52.802086  52.802086

     Volume    volume_adi  volume_obv  volume_cmf    ...      momentum_uo  \
0  44871300  3.656204e+06    44871300    0.081482    ...             50.0
1  22942800  2.248105e+07    67814100    0.331510    ...             50.0
2  18342800  7.036664e+06    86156900    0.081673    ...             50.0
3  15319700 -3.322742e+06    70837200   -0.032744    ...             50.0
4   9232100 -3.053841e+06    80069300   -0.027584    ...             50.0

   momentum_stoch  momentum_stoch_signal  momentum_wr  momentum_ao  \
0       54.074101              54.074101   -45.925899          0.0
1       94.131071              74.102586    -5.868929          0.0
2       76.712316              74.972496   -23.287684          0.0
3       50.856148              73.899845   -49.143852          0.0
4       57.305953              61.624806   -42.694047          0.0

   momentum_kama  momentum_roc  others_dr  others_dlr  others_cr
0      49.982655           0.0 -89.788542    0.000000   0.000000
1      53.952770           0.0   7.942985    7.643299   7.942985
2      54.495735           0.0   1.006371    1.001341   9.029292
3      52.239193           0.0  -4.140768   -4.228940   4.514642
4      52.802086           0.0   1.077530    1.071766   5.640819

[5 rows x 74 columns]
```

[ ]:
```
'''
    Determining how many observations there are per cluster.
    The more clusters the better as long as there are enough observations per␣
 ↪cluster.
    Deciding on n > 30 so we can assume normality.
'''
```

[22]:
```
# Only accepting indicators as inputs
X = df[df.columns[7:]].values

n_clusters = 15
clustered = cluster.KMeans(n_clusters=n_clusters).fit(X)

predictions = []
rewards = []
for i in range(len(X)-1):
    predict = clustered.predict([X[i]])[0]
    predictions.append(predict)

# All but one bin has > 30 observations
max_count = max(set(predictions), key=predictions.count)
```

```
for i in range(n_clusters):
    print("Cluster {} has {} observations".format(i, predictions.count(i)))

plt.title("Number of Observations per Cluster")
plt.hist(predictions, n_clusters)
plt.show()
```

```
Cluster 0 has 293 observations
Cluster 1 has 123 observations
Cluster 2 has 54 observations
Cluster 3 has 666 observations
Cluster 4 has 270 observations
Cluster 5 has 389 observations
Cluster 6 has 140 observations
Cluster 7 has 73 observations
Cluster 8 has 365 observations
Cluster 9 has 24 observations
Cluster 10 has 324 observations
Cluster 11 has 90 observations
Cluster 12 has 846 observations
Cluster 13 has 47 observations
Cluster 14 has 163 observations
```



[ ]:
```
'''
    This defines the reward structure for the Q Learning model.
```

```
    If the next 5 day high is greater than 1.01 then the reward is 0.01.
    If it is not, meaning the price never went above 1.01%, then I'm assuming I will sell it at negative 5%.

    The reward for not buying is the opposite.
'''
```

[24]:
```
def reward_structure(j, action):

    reward = max([x[1] for x in X[:6+j]]) / X[j+1][0]

    if action == 0:  # Buy and then sell on next 5 day high
        if reward > 1.01:
            return .01
        else:
            return -.05
    else:  # Don't Buy
        if reward > 1.01:
            return -.01
        else:
            return .05
```

[ ]:
```
'''
    Settings for Q
'''
```

[26]:
```
n_actions = 2 # Buy / Don't Buy
n_states = n_clusters
n_episodes = 10
alpha = .01 # Learning Rate

# Gamma not needed since there isn't a need for a transitional discount from one state to the next

epsilon = 1.0 / (n_actions * 1.0) # Exploring / Exploting at a 50/50 chance

Q = np.zeros([n_states, n_actions]) # Initializing Q table with zeros
```

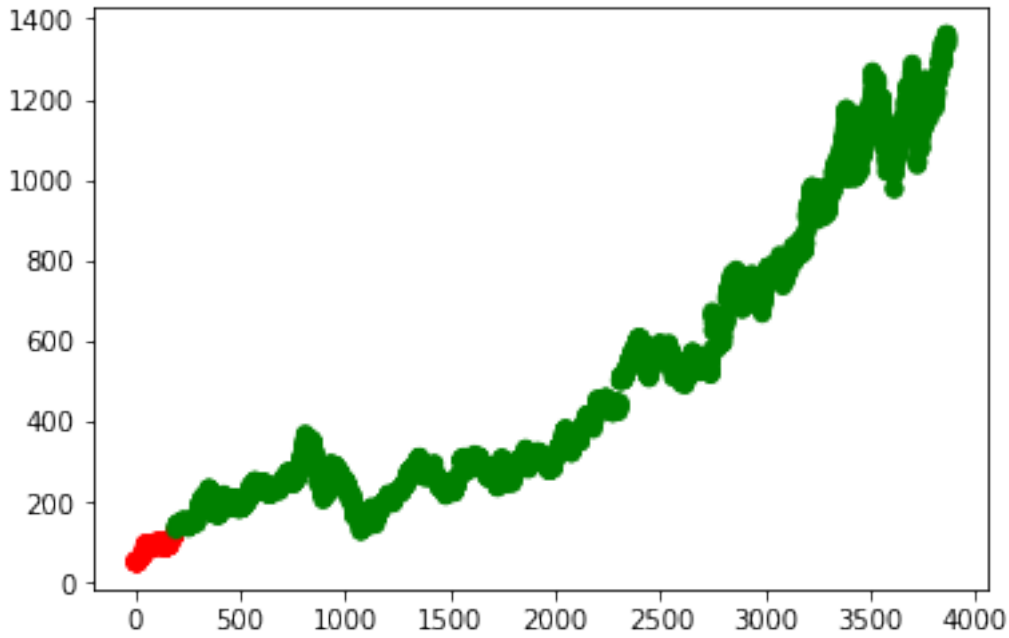
[28]:
```
for i in range(n_episodes):
    for j in range(len(X)-6):
        state = clustered.predict([X[j]])

        action = np.argmax(Q[state])
        if np.random.random() < epsilon:
            action = np.random.randint(n_actions)

        reward = reward_structure(j, action)

        Q[state, action] += alpha * reward
```

```
print(Q)
```

```
[[ 0.434  -0.152 ]
 [-0.2814  0.8646]
 [ 0.0796 -0.0284]
 [ 0.9913 -0.3307]
 [ 0.4053 -0.1347]
 [ 0.5826 -0.1954]
 [ 0.1618 -0.0582]
 [ 0.1064 -0.0396]
 [ 0.5502 -0.1798]
 [-0.0595  0.1805]
 [ 0.4876 -0.1604]
 [ 0.1317 -0.0483]
 [ 1.2614 -0.4306]
 [-0.011   0.051 ]
 [ 0.2441 -0.0819]]
```

[ ]:
```
'''

    With this reward structure and Q Learning model, we should buy as long as␣
 ↪we are not in state [1], [9], or [13].

    As highlighted below via Daily Close. It is interesting that all of them␣
 ↪are located at the start.

'''
```

[49]:
```
dont_buy = []
for x in range(len(df["Close"])):
    cluster = clustered.predict([X[x]])

    if cluster == 1 or cluster == 9 or cluster == 13:
        dont_buy.append('r')
    else:
        dont_buy.append('g')

plt.scatter([x for x in range(len(df["Close"]))], df["Close"], color=dont_buy)
plt.show()
```



[ ]:
```
''' Looking at the average gain per cluster. '''
```

[47]:
```
clusters = [[] for i in range(n_clusters)]

for x in range(1, len(df["Close"])):
    cluster = clustered.predict([X[x]])[0]

    change = df["Close"][x] / df["Close"][x-1]
    clusters[cluster].append(change)

for i in range(n_clusters):
    print("For Cluster {} - AVG CHANGE: {}".format(i, round(np.
 ↪average(clusters[i]), 5)))
```

```
For Cluster 0 - AVG CHANGE: 1.00012
For Cluster 1 - AVG CHANGE: 1.00342
For Cluster 2 - AVG CHANGE: 0.98386
For Cluster 3 - AVG CHANGE: 1.0007
For Cluster 4 - AVG CHANGE: 1.0006
For Cluster 5 - AVG CHANGE: 1.00075
For Cluster 6 - AVG CHANGE: 1.00009
For Cluster 7 - AVG CHANGE: 1.00015
For Cluster 8 - AVG CHANGE: 1.00256
For Cluster 9 - AVG CHANGE: 1.00937
For Cluster 10 - AVG CHANGE: 1.00056
For Cluster 11 - AVG CHANGE: 1.00293
```

```
For Cluster 12 - AVG CHANGE: 1.00073
For Cluster 13 - AVG CHANGE: 1.00784
For Cluster 14 - AVG CHANGE: 1.00529
```

[ ]:
```
''' More to analyze!!! '''
```