In [1]:
```
from sklearn.svm import SVC

from sklearn.metrics import accuracy_score

import pandas as pd

import numpy as np
import matplotlib.pyplot as plt
```

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

Out[2]: 4139

In [3]:
```
df.rename(columns={"Date":"date","Open":"open","High":"high","Low":"low","Clos
df.head()
```

Out[3]:

	date	Symbol	Series	Prev Close	open	high	low	Last	close	VWAP	Volume	
0	2004-08-25	TCS	EQ	850.00	1198.7	1198.7	979.00	985.00	987.95	1008.32	17116372	1.7
1	2004-08-26	TCS	EQ	987.95	992.0	997.0	975.30	976.85	979.00	985.65	5055400	4.9
2	2004-08-27	TCS	EQ	979.00	982.4	982.4	958.55	961.20	962.65	969.94	3830750	3.7
3	2004-08-30	TCS	EQ	962.65	969.9	990.0	965.00	986.40	986.75	982.65	3058151	3.0
4	2004-08-31	TCS	EQ	986.75	986.5	990.0	976.00	987.80	988.10	982.18	2649332	2.6

In [4]:
```
df.isnull().sum()
```

Out[4]:
```
date                    0
Symbol                  0
Series                  0
Prev Close              0
open                    0
high                    0
low                     0
Last                    0
close                   0
VWAP                    0
Volume                  0
Turnover                0
Trades               1683
Deliverable Volume      0
%Deliverble             0
dtype: int64
```

In [5]:
```
df.dropna(inplace=True)
df.isna().any()
```

Out[5]:
```
date                 False
Symbol               False
Series               False
Prev Close           False
open                 False
high                 False
low                  False
Last                 False
close                False
VWAP                 False
Volume               False
Turnover             False
Trades               False
Deliverable Volume   False
%Deliverble          False
dtype: bool
```

In [6]:
```
df.shape
```

Out[6]:
```
(2456, 15)
```

In [7]:
```
closedf = df['close']
print("Shape of close dataframe:", closedf.shape)
```

```
Shape of close dataframe: (2456,)
```

In [8]:
```
from sklearn.preprocessing import MinMaxScaler
scaler=MinMaxScaler(feature_range=(0,1))
closedf=scaler.fit_transform(np.array(closedf).reshape(-1,1))
print(closedf.shape)
```

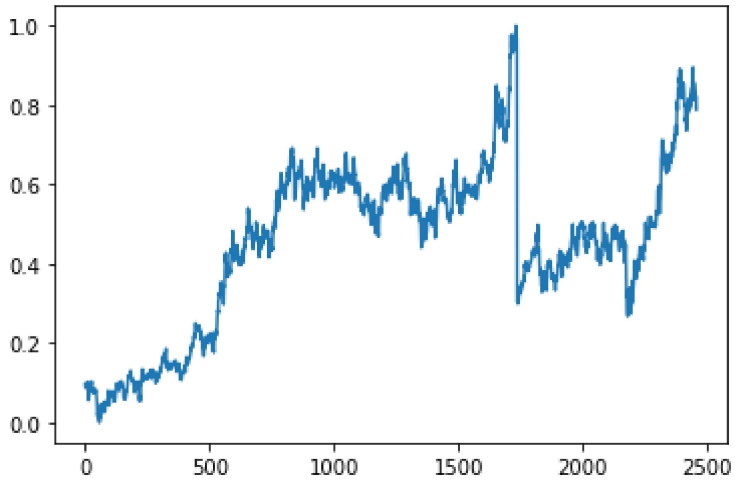
```
(2456, 1)
```

In [26]:
```
print(closedf)
```

```
[[0.0958241 ]
 [0.09297563]
 [0.08714836]
 ...
 [0.82142126]
 [0.81812597]
 [0.78848696]]
```

In [9]:
```
plt.plot(closedf)
```

Out[9]:
```
[<matplotlib.lines.Line2D at 0x115e34be3a0>]
```



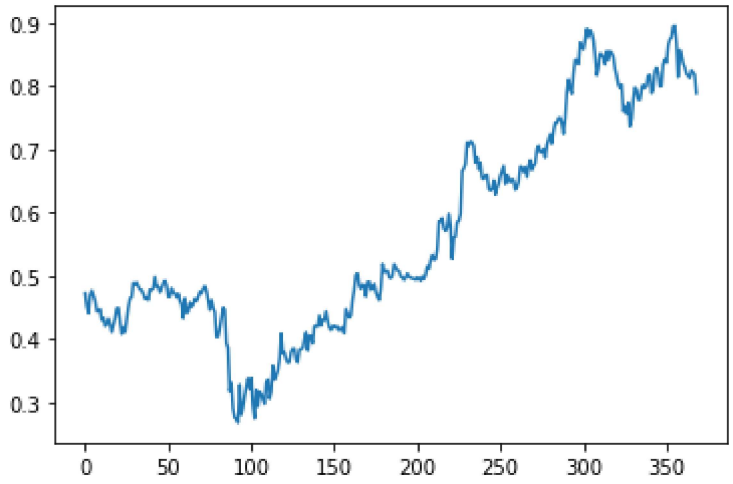
In [30]:
```
training_size=int(len(closedf)*0.85)
test_size=len(closedf)-training_size
train_data,test_data=closedf[0:training_size,:],closedf[training_size:len(clos
print("train_data: ", train_data.shape)
print("test_data: ", test_data.shape)
```

```
train_data:  (2087, 1)
test_data:  (369, 1)
```

In [11]:
```
scaler = MinMaxScaler(feature_range=(0, 1))
scaled_data = scaler.fit_transform(df['close'].values.reshape(-1, 1))
```

In [12]:
```
plt.plot(test_data)
```

Out[12]:
```
[<matplotlib.lines.Line2D at 0x115e364ba60>]
```



In [13]:
```
def create_dataset(dataset, time_step):
    dataX, dataY = [], []
    for i in range(len(dataset)-time_step-1):
        a = dataset[i:(i+time_step), 0]   ###i=0, 0,1,2,3-----99   100
        dataX.append(a)
        dataY.append(dataset[i + time_step, 0])
    return np.array(dataX), np.array(dataY)
```

In [14]:
```
# reshape into X=t,t+1,t+2,t+3 and Y=t+4
time_step = 30
X_train, y_train = create_dataset(train_data, time_step)
X_test, y_test = create_dataset(test_data, time_step)

print("X_train: ", X_train.shape)
print("y_train: ", y_train.shape)
print("X_test: ", X_test.shape)
print("y_test", y_test.shape)
```

```
X_train:  (2056, 30)
y_train:  (2056,)
X_test:  (338, 30)
y_test (338,)
```

In [15]:
```
from sklearn import svm
clf=svm.SVR()
clf.fit(X_train,y_train)
accuracy=clf.score(X_test,y_test)
print (accuracy)
```

```
0.943620075365538
```

In [16]:
```
from sklearn.svm import SVR

svr_rbf = SVR(kernel= 'rbf', C= 1e2, gamma= 0.1)
svr_rbf.fit(X_train, y_train)
```

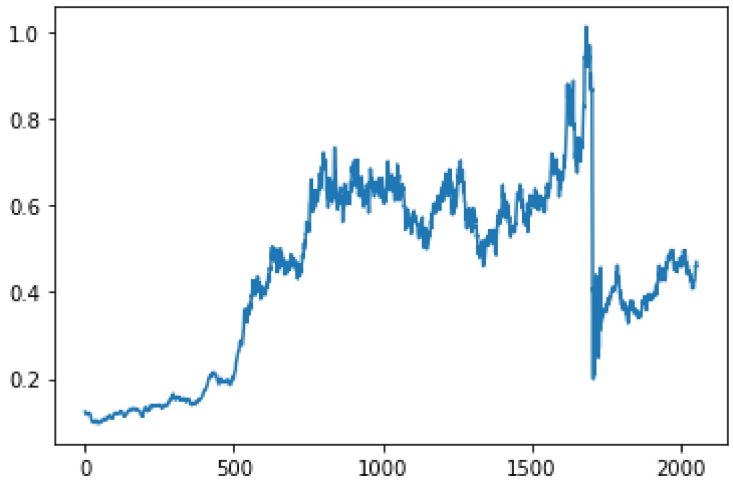
Out[16]:
```
SVR(C=100.0, gamma=0.1)
```

In [17]:
```
# Lets Do the prediction

train_predict=svr_rbf.predict(X_train)
test_predict=svr_rbf.predict(X_test)

train_predict = train_predict.reshape(-1,1)
test_predict = test_predict.reshape(-1,1)

print("Train data prediction:", train_predict.shape)
print("Test data prediction:", test_predict.shape)
```

```
Train data prediction: (2056, 1)
Test data prediction: (338, 1)
```

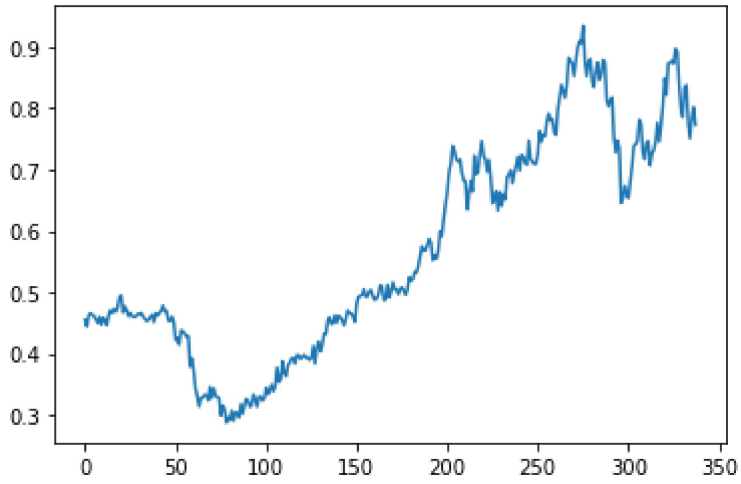
In [18]:
```
plt.plot(train_predict)
```

Out[18]:
```
[<matplotlib.lines.Line2D at 0x115e36cbf40>]
```



In [19]:
```
plt.plot(test_predict)
```

Out[19]:
```
[<matplotlib.lines.Line2D at 0x115e37337c0>]
```



In [20]:
```
train_predict = scaler.inverse_transform(train_predict)
test_predict = scaler.inverse_transform(test_predict)
original_ytrain = scaler.inverse_transform(y_train.reshape(-1,1))
original_ytest = scaler.inverse_transform(y_test.reshape(-1,1))
```

In [21]:
```
import math
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
# Evaluation metrices RMSE and MAE
print("Train data RMSE: ", math.sqrt(mean_squared_error(original_ytrain,train_
print("Train data MSE: ", mean_squared_error(original_ytrain,train_predict))
print("Test data MAE: ", mean_absolute_error(original_ytrain,train_predict))
print("-------------------------------------------------------------------------
print("Test data RMSE: ", math.sqrt(mean_squared_error(original_ytest,test_pre
print("Test data MSE: ", mean_squared_error(original_ytest,test_predict))
print("Test data MAE: ", mean_absolute_error(original_ytest,test_predict))
```

```
Train data RMSE:  93.00967965381454
Train data MSE:  8650.800509305203
Test data MAE:  72.49675842176669
-------------------------------------------------------------------------------
--------
Test data RMSE:  104.80747453012401
Test data MSE:  10984.606717382594
Test data MAE:  80.1787916181858
```

In [22]:
```
from sklearn.metrics import r2_score
print("Train data R2 score:", r2_score(original_ytrain, train_predict))
print("Test data R2 score:", r2_score(original_ytest, test_predict))
```

```
Train data R2 score: 0.9717946586579922
Test data R2 score: 0.9497401656707489
```

In [23]:
```
df6 = pd.DataFrame({'Actual': original_ytrain.flatten(), 'Predicted' : train_p
print(df6)
```

```
       Actual    Predicted
0     1149.05  1249.177259
1     1123.70  1245.052216
2     1146.05  1242.422110
3     1125.05  1239.325840
4     1140.05  1235.099704
...       ...          ...
2051  2252.80  2095.874860
2052  2269.65  2116.976633
2053  2200.90  2157.130954
2054  2193.95  2146.633011
2055  2201.85  2177.631512

[2056 rows x 2 columns]
```

In [24]:
```
plt.plot(df6)
plt.legend(['Targeted','Predicted'],loc='lower right')
```

Out[24]:
```
<matplotlib.legend.Legend at 0x115e376fdc0>
```