In [1]:
```python
import numpy as np
import pandas as pd
import yfinance as yf
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns
# Import as Dataframe
root = 'C:/Users/LENOVO/Downloads'
df = pd.read_csv(root + '/crosssection_AAPL.csv')
df.head()
```

Out[1]:

	Date1	Time1	Year	Month	Day	Hour	Minute	SYMBOL	BID	OFR	N_OFFER	N_BID
0	20130102	09:30	2013	1	2	9	30	AAPL	553.45	553.63	6953.0	10359.0
1	20130102	09:31	2013	1	2	9	31	AAPL	530.64	552.42	11562.0	11163.0
2	20130102	09:32	2013	1	2	9	32	AAPL	550.84	551.35	7629.0	6001.0
3	20130102	09:33	2013	1	2	9	33	AAPL	551.07	551.60	5306.0	5562.0
4	20130102	09:34	2013	1	2	9	34	AAPL	552.02	552.92	7767.0	9202.0

In [2]:
```python
df.tail()
```

Out[2]:

	Date1	Time1	Year	Month	Day	Hour	Minute	SYMBOL	BID	OFR	N_OFFER	N_BID
98527	20131231	15:56	2013	12	31	15	56	AAPL	561.14	561.21	1577.0	2588.0
98528	20131231	15:57	2013	12	31	15	57	AAPL	560.70	560.77	2540.0	2867.0
98529	20131231	15:58	2013	12	31	15	58	AAPL	560.55	560.60	2837.0	3185.0
98530	20131231	15:59	2013	12	31	15	59	AAPL	560.81	560.98	5891.0	7227.0
98531	20131231	16:00	2013	12	31	16	0	AAPL	561.11	561.19	9063.0	7702.0

In [3]:
```python
df.describe()
```

Out[3]:

	Date1	Year	Month	Day	Hour	Minute	BID	OFR	N_OFF
count	9.853200e+04	98532.0	98532.000000	98532.000000	98532.000000	98532.000000	98193.000000	98193.000000	9.819300e
mean	2.013067e+07	2013.0	6.543651	15.650794	12.240409	30.575448	471.024906	471.730246	3.481483e
std	3.427930e+02	0.0	3.426409	8.707200	1.894544	17.326909	51.694662	51.776169	1.694026e
min	2.013010e+07	2013.0	1.000000	1.000000	9.000000	0.000000	3.880000	3.950000	1.000000e
25%	2.013040e+07	2013.0	4.000000	8.000000	11.000000	16.000000	435.660000	436.320000	1.231000e
50%	2.013070e+07	2013.0	7.000000	15.500000	12.000000	32.000000	460.670000	461.290000	2.121000e
75%	2.013100e+07	2013.0	10.000000	23.000000	14.000000	46.000000	505.160000	506.120000	3.620000e
max	2.013123e+07	2013.0	12.000000	31.000000	16.000000	59.000000	574.690000	598.000000	1.117690e

In [4]:
```python
#Question 2
df.index = pd.to_datetime(df[['Year', 'Month', 'Day','Hour', 'Minute']])
df.head()
```

Out[4]:

	Date1	Time1	Year	Month	Day	Hour	Minute	SYMBOL	BID	OFR	N_OFFER	N_BID
2013-01-02 09:30:00	20130102	09:30	2013	1	2	9	30	AAPL	553.45	553.63	6953.0	10359.0
2013-01-02 09:31:00	20130102	09:31	2013	1	2	9	31	AAPL	530.64	552.42	11562.0	11163.0
2013-01-02 09:32:00	20130102	09:32	2013	1	2	9	32	AAPL	550.84	551.35	7629.0	6001.0
2013-01-02 09:33:00	20130102	09:33	2013	1	2	9	33	AAPL	551.07	551.60	5306.0	5562.0
2013-01-02 09:34:00	20130102	09:34	2013	1	2	9	34	AAPL	552.02	552.92	7767.0	9202.0

In [5]:
```python
#Question 3
df['AvgPriceAAPL'] = (df['OFR']+df['BID'])/2
```

In [6]:
```python
#Quesion 4
df['RetAAPL0'] = 100*df['AvgPriceAAPL'].pct_change()
```

In [7]:
```python
#Question 5
df['RetAAPL1'] = df['RetAAPL0'].shift(-1)
df['RetAAPL2'] = df['RetAAPL1'].shift(-1)
df['RetAAPL3'] = df['RetAAPL2'].shift(-1)
df['RetAAPL4'] = df['RetAAPL3'].shift(-1)
df['RetAAPL5'] = df['RetAAPL4'].shift(-1)
df.head()
```

Out[7]:

	Date1	Time1	Year	Month	Day	Hour	Minute	SYMBOL	BID	OFR	N_OFFER	N_BID	AvgPriceAAPL	Re
2013-01-02 09:30:00	20130102	09:30	2013	1	2	9	30	AAPL	553.45	553.63	6953.0	10359.0	553.540	
2013-01-02 09:31:00	20130102	09:31	2013	1	2	9	31	AAPL	530.64	552.42	11562.0	11163.0	541.530	-2
2013-01-02 09:32:00	20130102	09:32	2013	1	2	9	32	AAPL	550.84	551.35	7629.0	6001.0	551.095	1
2013-01-02 09:33:00	20130102	09:33	2013	1	2	9	33	AAPL	551.07	551.60	5306.0	5562.0	551.335	0
2013-01-02 09:34:00	20130102	09:34	2013	1	2	9	34	AAPL	552.02	552.92	7767.0	9202.0	552.470	0

In [8]:
```python
#Question 6
df['BASprd'] = 100*(df['OFR']-df['BID'])/(df['OFR']+df['BID'])
df['BAImb'] = 100*(df['N_BID']-df['N_OFFER'])/(df['N_BID']+df['N_OFFER'])
```

In [9]:
```python
#Question 7
df.dropna(how='any', inplace=True)
```

In [10]:
```python
df.head()
```

Out[10]:

	Date1	Time1	Year	Month	Day	Hour	Minute	SYMBOL	BID	OFR	...	N_BID	AvgPriceAAPL	RetAAPL0
2013-01-02 09:31:00	20130102	09:31	2013	1	2	9	31	AAPL	530.64	552.42	...	11163.0	541.530	-2.169672
2013-01-02 09:32:00	20130102	09:32	2013	1	2	9	32	AAPL	550.84	551.35	...	6001.0	551.095	1.766292
2013-01-02 09:33:00	20130102	09:33	2013	1	2	9	33	AAPL	551.07	551.60	...	5562.0	551.335	0.043550
2013-01-02 09:34:00	20130102	09:34	2013	1	2	9	34	AAPL	552.02	552.92	...	9202.0	552.470	0.205864
2013-01-02 09:35:00	20130102	09:35	2013	1	2	9	35	AAPL	553.76	554.89	...	14888.0	554.325	0.335765

5 rows × 21 columns

In [11]:
```python
df.tail()
```

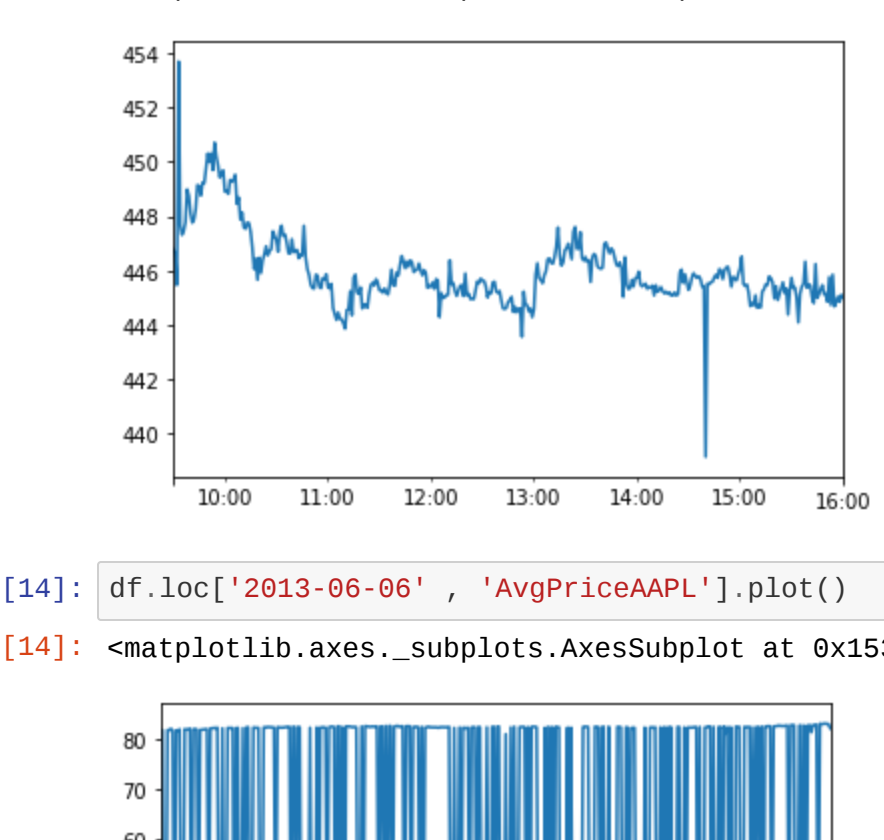
Out[11]:

	Date1	Time1	Year	Month	Day	Hour	Minute	SYMBOL	BID	OFR	...	N_BID	AvgPriceAAPL	RetAAPL0
2013-12-31 15:51:00	20131231	15:51	2013	12	31	15	51	AAPL	560.35	561.33	...	3838.0	560.840	0.000892
2013-12-31 15:52:00	20131231	15:52	2013	12	31	15	52	AAPL	560.85	560.89	...	1130.0	560.870	0.005349
2013-12-31 15:53:00	20131231	15:53	2013	12	31	15	53	AAPL	560.98	561.06	...	3224.0	561.020	0.026744
2013-12-31 15:54:00	20131231	15:54	2013	12	31	15	54	AAPL	561.09	561.20	...	3305.0	561.145	0.022281
2013-12-31 15:55:00	20131231	15:55	2013	12	31	15	55	AAPL	561.16	561.66	...	2079.0	561.410	0.047225

5 rows × 21 columns

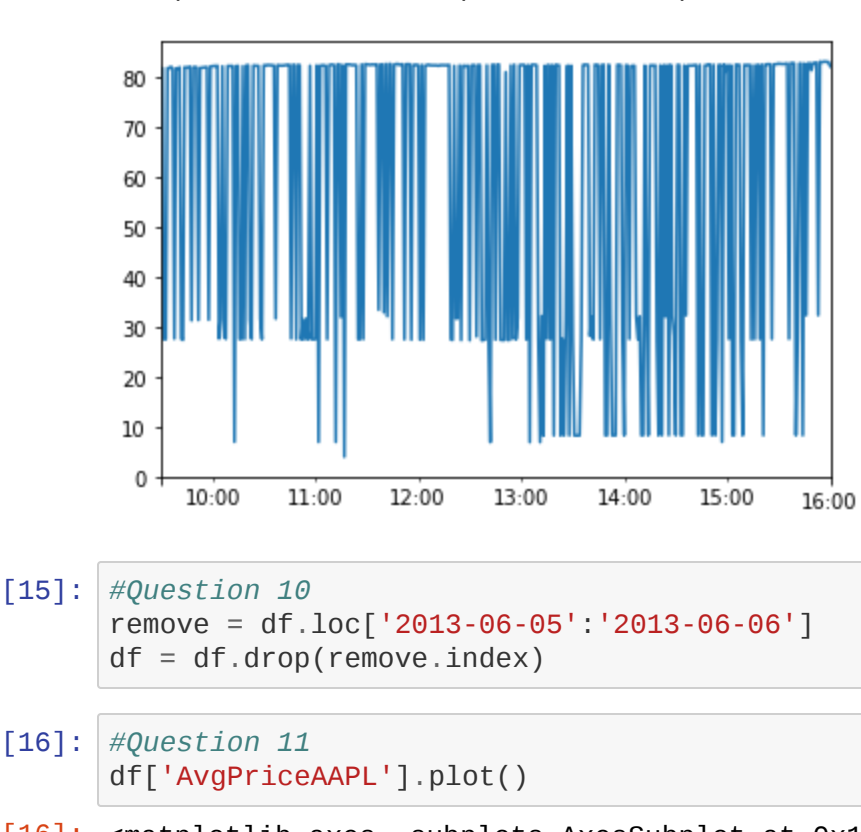
In [12]:
```python
#Question 8
df['AvgPriceAAPL'].plot()
```

Out[12]: <matplotlib.axes._subplots.AxesSubplot at 0x153255bf1c8>



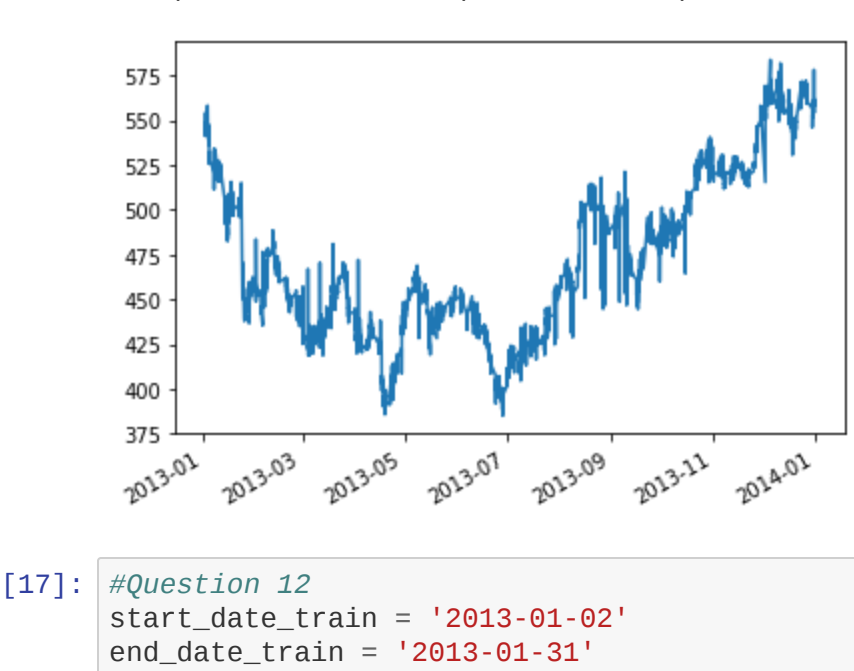
In [13]:
```python
#Question 9
df.loc['2013-06-05' , 'AvgPriceAAPL'].plot()
```

Out[13]: <matplotlib.axes._subplots.AxesSubplot at 0x15326ff44c8>



In [14]:
```python
df.loc['2013-06-06' , 'AvgPriceAAPL'].plot()
```

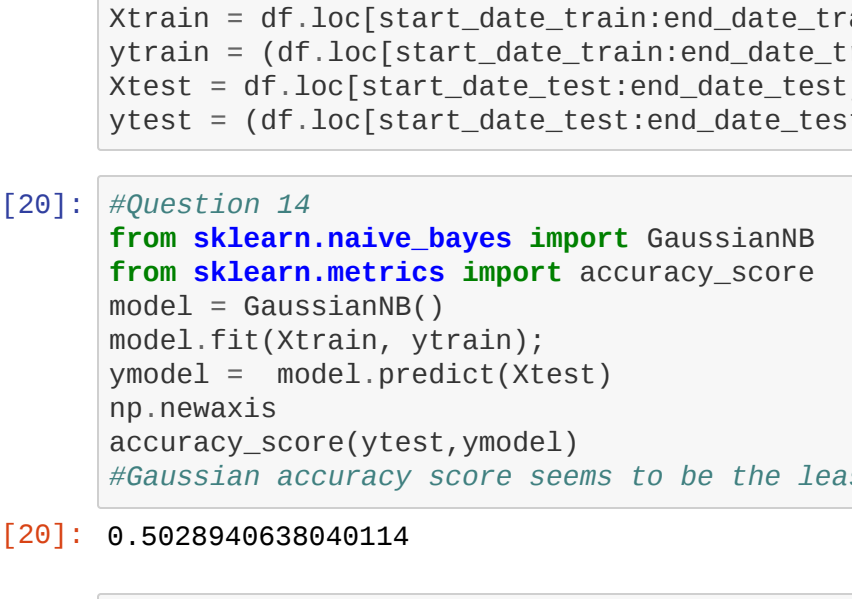
Out[14]: <matplotlib.axes._subplots.AxesSubplot at 0x15326fcadc8>



In [15]:
```python
#Question 10
remove = df.loc['2013-06-05':'2013-06-06']
df = df.drop(remove.index)
```

In [16]:
```python
#Question 11
df['AvgPriceAAPL'].plot()
```

Out[16]: <matplotlib.axes._subplots.AxesSubplot at 0x153277d9788>



In [17]:
```python
#Question 12
start_date_train = '2013-01-02'
end_date_train = '2013-01-31'
```

In [18]:
```python
start_date_test = '2013-02-01'
end_date_test = '2013-02-28'
```

In [19]:
```python
#Question 13
Xtrain = df.loc[start_date_train:end_date_train,'RetAAPL0':'RetAAPL4'].values
ytrain = (df.loc[start_date_train:end_date_train,'RetAAPL5'].values >0 ).astype(int)
Xtest = df.loc[start_date_test:end_date_test,'RetAAPL0':'RetAAPL4'].values
ytest = (df.loc[start_date_test:end_date_test,'RetAAPL5'].values >0 ).astype(int)
```

In [20]:
```python
#Question 14
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score
model = GaussianNB()
model.fit(Xtrain, ytrain);
ymodel =  model.predict(Xtest)
np.newaxis
accuracy_score(ytest,ymodel)
#Gaussian accuracy score seems to be the least.
```

Out[20]: 0.5028940638040114

In [21]:
```python
#Question 15
from sklearn.svm import SVC
model_SVC = SVC(kernel='rbf', C=10, gamma='auto')
model_SVC.fit(Xtrain, ytrain)
```

Out[21]: SVC(C=10, gamma='auto')

In [22]:
```python
y_pred_SVC = model_SVC.predict(Xtest)
accuracy_score(ytest , y_pred_SVC)
#The accuracy is 58% which is higher than Gaussian
```

Out[22]: 0.580966482702921

In [23]:
```python
#Question 16
from sklearn.model_selection import GridSearchCV
param_grid = {'C': [1, 5, 10, 50]}
grid = GridSearchCV(model_SVC, param_grid,cv = 5)
grid.fit(Xtrain, ytrain)
print(grid.best_params_)
#10 is the best parameter.
```

{'C': 10}

In [24]:
```python
#Quesion 17
from sklearn.ensemble import RandomForestClassifier
model_RF = RandomForestClassifier(n_estimators=100, random_state=0)
model_RF.fit(Xtrain, ytrain)
```

Out[24]: RandomForestClassifier(random_state=0)

In [25]:
```python
y_pred_RF = model_RF.predict(Xtest)
accuracy_score(ytest, y_pred_RF)
#With 54.7% accurage this is better than Gaussian but Lesser than SVC
```

Out[25]: 0.54785300848028

In [26]:
```python
#Question 18
y_model_NB = ymodel
y_model_SVC = y_pred_SVC
y_model_RF = y_pred_RF
y_model_NB2 = ymodel
y_model_SVC2 = y_pred_SVC
y_model_RF2 = y_pred_RF
```

In [27]:
```python
RetNB = (y_model_NB*df.loc[start_date_test:end_date_test,'RetAAPL5'].values
-(1-y_model_NB)*df.loc[start_date_test:end_date_test,'RetAAPL5'].values)
CsRetNB = (1+0.01*RetNB).cumprod()
CsRetNB
```

Out[27]:
```
array([0.99992338, 1.00031741, 1.0016209 , ..., 1.02914025, 1.02885695,
       1.02931731])
```

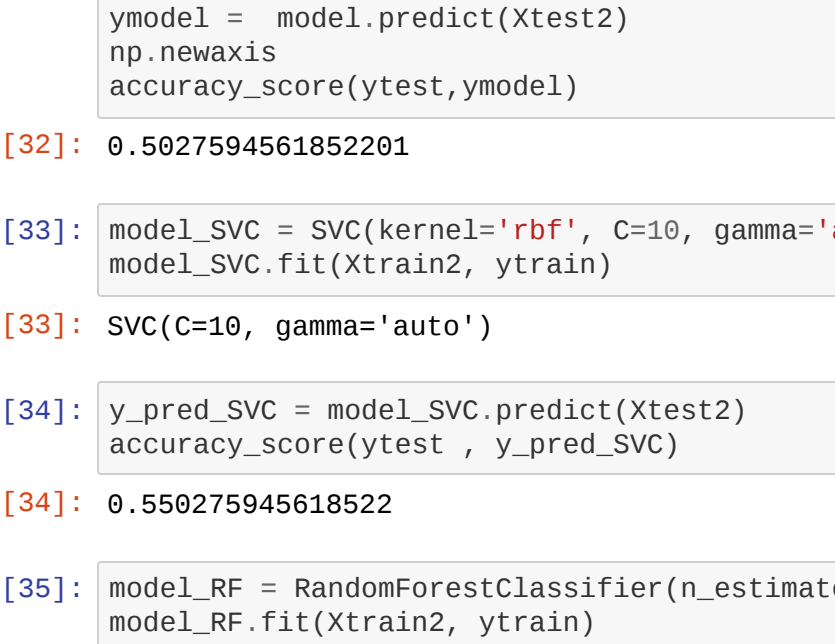
In [28]:
```python
RetSVC = (y_model_SVC*df.loc[start_date_test:end_date_test,'RetAAPL5'].values
-(1-y_model_SVC)*df.loc[start_date_test:end_date_test,'RetAAPL5'].values)
CsRetSVC = (1+0.01*RetSVC).cumprod()
CsRetSVC
```

Out[28]:
```
array([0.99992338, 1.00031741, 0.99901391, ..., 6.16836922, 6.16667121,
       6.16391195])
```

In [29]:
```python
RetRF = (y_model_RF*df.loc[start_date_test:end_date_test,'RetAAPL5'].values
-(1-y_model_RF)*df.loc[start_date_test:end_date_test,'RetAAPL5'].values)
CsRetRF = (1+0.01*RetRF).cumprod()
CsRetRF
```

Out[29]:
```
array([0.99992338, 1.00031741, 0.99901391, ..., 3.1224414 , 3.12158186,
       3.12018512])
```

In [30]:
```python
#Question 19
plt.plot(CsRetNB, label ='Cumilative Return Gaussian')
plt.plot(CsRetSVC, label ='Cumilative Return SVC')
plt.plot(CsRetRF, label ='Cumilative Return RF')
plt.ylabel('Cum Returns')
plt.legend()
plt.show()
```



In [31]:
```python
#Question 20
Xtrain2 = df.loc[start_date_train:end_date_train,'RetAAPL0':'BAImb'].drop(['RetAAPL5'], axis
= 1).values
Xtest2 = df.loc[start_date_test:end_date_test,'RetAAPL0':'BAImb'].drop(['RetAAPL5'], axis =
1).values
```

In [32]:
```python
model = GaussianNB()
model.fit(Xtrain2, ytrain);
ymodel =  model.predict(Xtest2)
np.newaxis
accuracy_score(ytest,ymodel)
```

Out[32]: 0.5027594561852201

In [33]:
```python
model_SVC = SVC(kernel='rbf', C=10, gamma='auto')
model_SVC.fit(Xtrain2, ytrain)
```

Out[33]: SVC(C=10, gamma='auto')

In [34]:
```python
y_pred_SVC = model_SVC.predict(Xtest2)
accuracy_score(ytest , y_pred_SVC)
```

Out[34]: 0.550275945618522

In [35]:
```python
model_RF = RandomForestClassifier(n_estimators=100, random_state=0)
model_RF.fit(Xtrain2, ytrain)
```

Out[35]: RandomForestClassifier(random_state=0)

In [36]:
```python
y_pred_RF = model_RF.predict(Xtest2)
accuracy_score(ytest, y_pred_RF)
```

Out[36]: 0.552698882756764

In [37]:
```python
y_model_NB = ymodel
y_model_SVC = y_pred_SVC
y_model_RF = y_pred_RF
```

In [38]:
```python
RetNB = (y_model_NB*df.loc[start_date_test:end_date_test,'RetAAPL5'].values
-(1-y_model_NB)*df.loc[start_date_test:end_date_test,'RetAAPL5'].values)
CsRetNB = (1+0.01*RetNB).cumprod()
CsRetNB
```

Out[38]:
```
array([0.99992338, 1.00031741, 1.0016209 , ..., 1.03869202, 1.03840609,
       1.03887072])
```

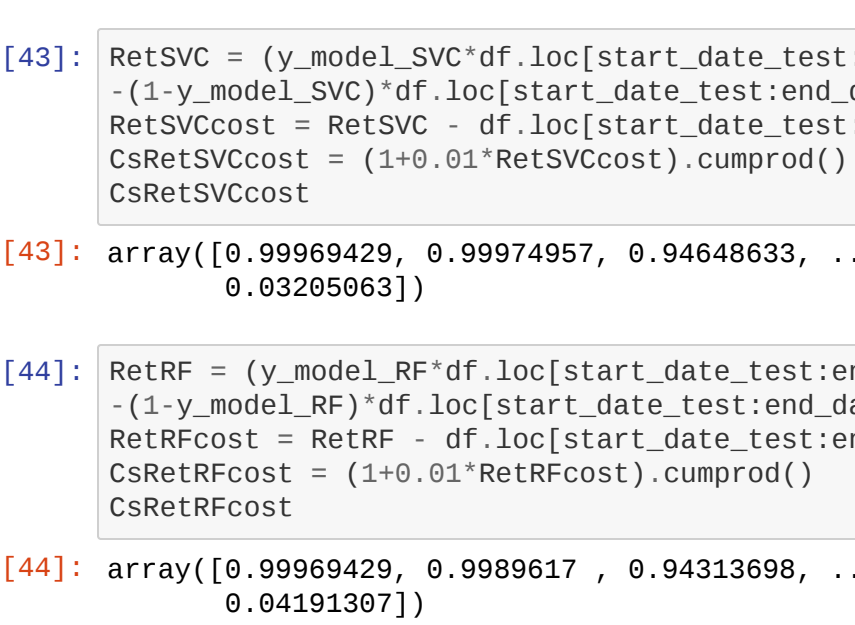
In [39]:
```python
RetSVC = (y_model_SVC*df.loc[start_date_test:end_date_test,'RetAAPL5'].values
-(1-y_model_SVC)*df.loc[start_date_test:end_date_test,'RetAAPL5'].values)
CsRetSVC = (1+0.01*RetSVC).cumprod()
CsRetSVC
```

Out[39]:
```
array([0.99992338, 1.00031741, 1.0016209 , ..., 3.34263558, 3.34171543,
       3.34321068])
```

In [40]:
```python
RetRF = (y_model_RF*df.loc[start_date_test:end_date_test,'RetAAPL5'].values
-(1-y_model_RF)*df.loc[start_date_test:end_date_test,'RetAAPL5'].values)
CsRetRF = (1+0.01*RetRF).cumprod()
CsRetRF
```

Out[40]:
```
array([0.99992338, 0.99952936, 0.99822689, ..., 4.37601168, 4.37480706,
       4.37284956])
```

In [41]:
```python
plt.plot(CsRetNB, label ='Cumilative Return Gaussian')
plt.plot(CsRetSVC, label ='Cumilative Return SVC')
plt.plot(CsRetRF, label ='Cumilative Return RF')
plt.ylabel('Cum Returns')
plt.legend()
plt.show()
```



In [42]:
```python
#Question 21
RetNB = (y_model_NB*df.loc[start_date_test:end_date_test,'RetAAPL5'].values
-(1-y_model_NB)*df.loc[start_date_test:end_date_test,'RetAAPL5'].values)
RetNBcost = RetNB - df.loc[start_date_test:end_date_test,'BASprd'].values
CsRetNBcost = (1+0.01*RetNBcost).cumprod()
CsRetNBcost
```

Out[42]:
```
array([0.99969429, 0.99974957, 0.94648633, ..., 0.00997628, 0.00995238,
       0.00995312])
```

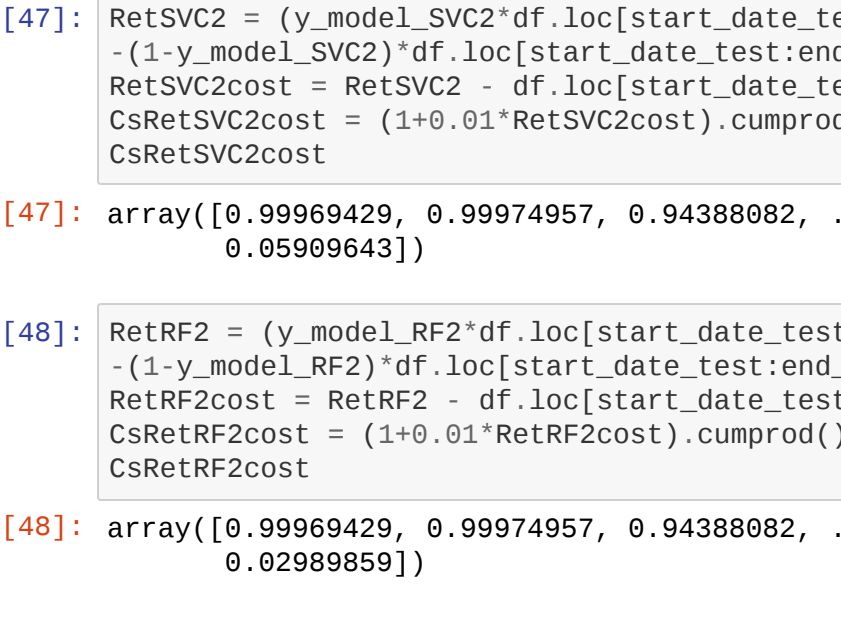
In [43]:
```python
RetSVC = (y_model_SVC*df.loc[start_date_test:end_date_test,'RetAAPL5'].values
-(1-y_model_SVC)*df.loc[start_date_test:end_date_test,'RetAAPL5'].values)
RetSVCcost = RetSVC - df.loc[start_date_test:end_date_test,'BASprd'].values
CsRetSVCcost = (1+0.01*RetSVCcost).cumprod()
CsRetSVCcost
```

Out[43]:
```
array([0.99969429, 0.99974957, 0.94648633, ..., 0.03212522, 0.03204827,
       0.03205063])
```

In [44]:
```python
RetRF = (y_model_RF*df.loc[start_date_test:end_date_test,'RetAAPL5'].values
-(1-y_model_RF)*df.loc[start_date_test:end_date_test,'RetAAPL5'].values)
RetRFcost = RetRF - df.loc[start_date_test:end_date_test,'BASprd'].values
CsRetRFcost = (1+0.01*RetRFcost).cumprod()
CsRetRFcost
```

Out[44]:
```
array([0.99969429, 0.9989617 , 0.94313698, ..., 0.04204824, 0.04194752,
       0.04191307])
```

In [45]:
```python
plt.plot(CsRetNBcost, label ='Cumilative Return Gaussian')
plt.plot(CsRetSVCcost, label ='Cumilative Return SVC')
plt.plot(CsRetRFcost, label ='Cumilative Return RF')
plt.ylabel('Cum Returns')
plt.legend()
plt.show()
```



In [46]:
```python
RetNB2 = (y_model_NB2*df.loc[start_date_test:end_date_test,'RetAAPL5'].values
-(1-y_model_NB2)*df.loc[start_date_test:end_date_test,'RetAAPL5'].values)
RetNB2cost = RetNB2 - df.loc[start_date_test:end_date_test,'BASprd'].values
CsRetNB2cost = (1+0.01*RetNB2cost).cumprod()
CsRetNB2cost
```

Out[46]:
```
array([0.99969429, 0.99974957, 0.94648633, ..., 0.00988299, 0.00985932,
       0.00986004])
```

In [47]:
```python
RetSVC2 = (y_model_SVC2*df.loc[start_date_test:end_date_test,'RetAAPL5'].values
-(1-y_model_SVC2)*df.loc[start_date_test:end_date_test,'RetAAPL5'].values)
RetSVC2cost = RetSVC2 - df.loc[start_date_test:end_date_test,'BASprd'].values
CsRetSVC2cost = (1+0.01*RetSVC2cost).cumprod()
CsRetSVC2cost
```

Out[47]:
```
array([0.99969429, 0.99974957, 0.94388082, ..., 0.05928701, 0.059145  ,
       0.05909643])
```

In [48]:
```python
RetRF2 = (y_model_RF2*df.loc[start_date_test:end_date_test,'RetAAPL5'].values
-(1-y_model_RF2)*df.loc[start_date_test:end_date_test,'RetAAPL5'].values)
RetRF2cost = RetRF2 - df.loc[start_date_test:end_date_test,'BASprd'].values
CsRetRF2cost = (1+0.01*RetRF2cost).cumprod()
CsRetRF2cost
```

Out[48]:
```
array([0.99969429, 0.99974957, 0.94388082, ..., 0.02999501, 0.02992316,
       0.02989859])
```

In [49]:
```python
plt.plot(CsRetNB2cost, label ='Cumilative Return Gaussian')
plt.plot(CsRetSVC2cost, label ='Cumilative Return SVC')
plt.plot(CsRetRF2cost, label ='Cumilative Return RF')
plt.ylabel('Cum Returns')
plt.legend()
plt.show()
```


In [ ]:
```python
#Second Last graph X includes BA columns. And last graph doesn't.
```