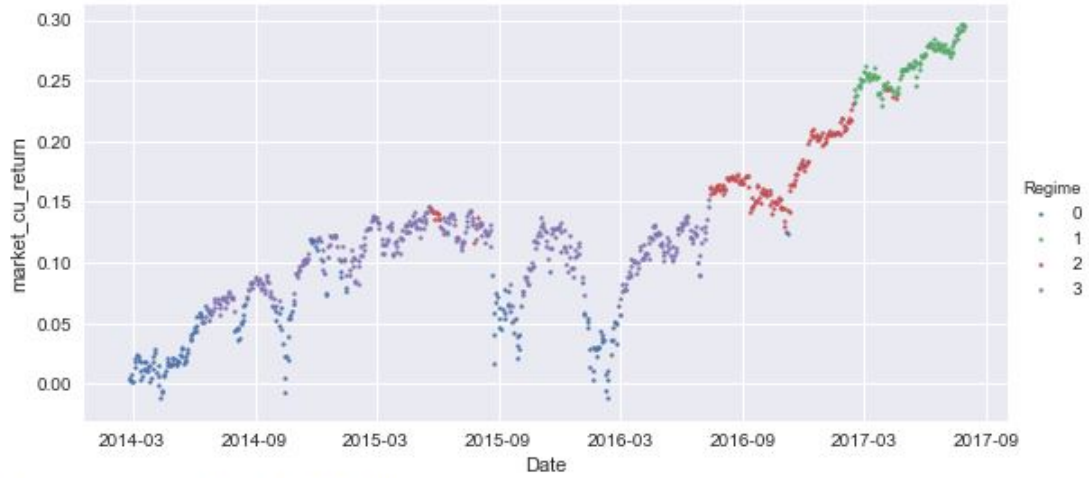
Towards the end of the last blog, I printed the Mean and Covariance values for all the regimes and plotted the regimes. The new output with indicators as feature set would look like this:




```
Mean for regime 0:  -1.2311519005
Co-Variance for regime 0:  0.848747281837
Mean for regime 1:  2.1245975683
Co-Variance for regime 1:  0.350080406727
Mean for regime 2:  0.690047980703
Co-Variance for regime 2:  0.425844469889
Mean for regime 3:  -0.331060686123
Co-Variance for regime 3:  0.324614165759
```

Next, I scaled the Regimes data frame, excluding the Date and Regimes columns, created in the earlier piece of code and saved it back in the same columns. By doing so, I will not be losing any features but the data will be scaled and ready for training the support vector classifier algorithm. Next, I created a signal column which would act as the prediction values. The algorithm would train on the features' set to predict this signal.

```
ss1 =StandardScaler()
columns =Regimes.columns.drop(['Regime','Date'])
Regimes[columns]= ss1.fit_transform(Regimes[columns])
Regimes['Signal']=0
Regimes.loc[Regimes['Return']>0,'Signal']=1
Regimes.loc[Regimes['Return']<0,'Signal']=-1
```

Next, I instantiated a support vector classifier. For this, I used the same SVC model used in the example by sklearn. I have not optimized this support vector classifier for best hyper parameters. In the machine learning course on Quantra™, we have extensively discussed how to use hyper parameters and optimize the algorithm to predict the daily Highs and Lows, in turn the volatility of the day.

Coming back to the blog, the code for support vector classifier is as below:

```
cls= SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape=None, degree=3, gamma='auto', kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

Next, I split the test data of the unsupervised regime algorithm into train and test data. We use this new train data to train our support vector classifier algorithm. To create the train data I dropped the columns that are not a part of the feature set:

'Signal','Return','market_cu_return','Date'

Then I fit the X and y data sets to the algorithm to train it on.

```
split2= int(.8*len(Regimes))

X = Regimes.drop(['Signal','Return','market_cu_return','Date'], axis=1)
y= Regimes['Signal']

cls.fit(X[:split2],y[:split2])
```


Next, I calculated the test set size and indexed the predictions accordingly to the data frame df.

The reason for doing this is that the original return values of 'SPY' are stored in df, while those in Regimes is scaled hence, won't be useful for taking a cumulative sum to check for the performance.

```
p_data=len(X)-split2
```

Next, I saved the predictions made by the SVC in a column named Pred_Signal.

Then, based on these signals I calculated the returns of the strategy by multiplying signal at the beginning of the day with the return at the opening (because our returns are from open to open) of the next day.

```
df['Pred_Signal']=0
df.iloc[-p_data:,df.columns.get_loc('Pred_Signal')]=cls.predict(X[split2:])

print(df['Pred_Signal'][-p_data:])

df['str_ret'] =df['Pred_Signal']*df['Return'].shift(-1)
```

Finally, I calculated the cumulative strategy returns and the cumulative market returns and saved them in df. Then, I calculated the sharpe ratio to measure the performance. To get a clear understanding of this metric I plotted the performance to measure it.