GROUP ASSIGNMENT No. 3

By Group 4-B

Name	Location-Country	E-Mail Address	Non-Contributing Member (X)
Ali Metwaly	Egypt	aiam_2000@hotmail.com	
Dhruv Agrawal	India	dhruva1@stanfordalumni.org	
Gift Mpofu	Botswana	mpofu.gift@yahoo.com	
Liberty Nkonde	South Africa	conteliberty@gmail.com	
Robert Gadzai	South Africa	gadzairobert@gmail.com	

Submission 3: Modeling and Strategy Development

Introduction

In this project we will build a model for the USDEUR FX asset based on classification (forecast if the asset will move up and down, above some threshold such as the 90-day standard deviation). We follow the below steps:

1. Select an algorithm or group of algorithms (for example, neural networks, deep learning, ARMA, ensemble techniques).

2. Fit the model: show that it works out of sample, and use appropriate validation techniques.

3. For the model, provide the following performance metrics:

a. ROC curves

b. Confusion Matrix

c. Precision, recall, F1-Score, Accuracy, and AUC

4. Analyze the metrics and develop a report

5. Create a fund factsheet for our new investment strategy

Initial setup:

We first import the relevant libraries and get the data using the code below:

Import Libraries

In [67]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import yfinance as yf
```

Get Data

In [77]:
```
# Download price history of USDEUR asset
data_frame = yf.download('USDEUR=X', start='2004-04-05', end='2020-06-02',
                         progress=False)

#Calculating the returns
data_frame['Daily_log_Return'] = np.log(data_frame['Adj Close']/
                                        data_frame['Adj Close'].shift(1))
data_frame.drop(['Open','Volume', 'Close'], axis=1, inplace=True)
data_frame.dropna(inplace=True)
data_frame.head()
```

Out[77]:

Date	High	Low	Adj Close	Daily_log_Return
2004-04-06	0.83410	0.82332	0.82433	-0.009778
2004-04-07	0.82926	0.81981	0.82196	-0.002879
2004-04-08	0.82871	0.81853	0.82802	0.007346
2004-04-09	0.82871	0.82556	0.82740	-0.000749
2004-04-12	0.82932	0.82672	0.82829	0.001075

We then compute the explanatory variables and the return labels (+1 indicating we will go long the asset if it is positive and greater than 90 day standard deviation, -1 indicating we will go short the asset if it is negative and less than -1*90 day standard deviation, and 0 otherwise indicating we will not take a position in the asset on that day). All positions are taken at the beginning of the trading day and closed at the end of the trading day. We use the code below:

```
In [69]: def Historical_D_days_trend(data, D):
             return ( data/data.shift(D-1, axis=0) )-1

         def StochasticOsclillator(data):
             temp_df = pd.DataFrame(data=data)
             temp_df['LowMin'] = data['Low'].rolling(window=14).min()
             temp_df['HighMax'] = data['High'].rolling(window=14).max()
             return (data['Adj Close']-temp_df['LowMin'])/(temp_df['HighMax']-temp_df['LowMin'])

         def Line_trend(data, w):
             return (data['High'].rolling(window=w).max() + data['Low'].rolling(window=w).min())/2

         data_frame['StochasticOsclillator'] = StochasticOsclillator(data_frame)

         data_frame['3-period MA of %K %D'] =
         data_frame['StochasticOsclillator'].rolling(window=3).mean()

         data_frame['3-period MA of %D'] =
         data_frame['3-period MA of %K %D'].rolling(window=3).mean()

         data_frame['ConvLine'] = Line_trend(data_frame, 9)
         data_frame['BaseLine'] = Line_trend(data_frame, 26)
         data_frame['SenkouSpanA'] = (data_frame['BaseLine'] + data_frame['ConvLine'])/2

         data_frame['Historical 4-day trend'] =
         Historical_D_days_trend(data_frame['Adj Close'], 4)
         data_frame['Historical 8-day trend'] =
         Historical_D_days_trend(data_frame['Adj Close'], 8)
         data_frame['Historical 16-day trend'] =
         Historical_D_days_trend(data_frame['Adj Close'], 16)
         data_frame['Historical 32-day trend'] =
         Historical_D_days_trend(data_frame['Adj Close'], 32)

         window_size = 90

         data_frame['90 day standard deviation'] =
         data_frame['Daily_log_Return'].rolling(window=window_size).std().dropna()

         data_frame = data_frame.dropna()

         Return = data_frame['Daily_log_Return']
         Std = data_frame['90 day standard deviation']

         # making the correct classification for the stock return

         label=np.zeros(np.size(Return))

         for i in range(np.size(Return)):
             if Return[i] > Std[i]:
                 #if it is positive and greater than 90 day standard deviation, label is 1
                 label[i] = 1

             elif Return[i] < (-1*Std[i]):
                 #else if it is negative and less than 90 day standard deviation, label is -1
                 label[i] = -1

             else:    #else label is 0
                 label[i] = 0


         data_frame['Label'] = label

         data_frame.drop(['High','Low','HighMax','LowMin','Adj Close', 'Daily_log_Return',
                          '90 day standard deviation'], axis=1, inplace=True)
         print(data_frame)
```

We then separate the data into explanatory variables and features, binarize the return label y, do a train test split, and apply feature scaling to the explanatory variables using the code below:

Separating explanatory variables and Labels

```
In [ ]: # X has all the explanatory variables in the dataframe without the last column "Label"
X = data_frame.iloc[:,0:-1].values
y = data_frame.iloc[:,-1].values      # y is the vector "Labels"
```

Train-Test split

```
In [ ]: from sklearn.model_selection import train_test_split
from sklearn.preprocessing import label_binarize

y = label_binarize(y, classes=[-1, 0, 1]) #binarize y for one vs rest classification

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20,
                                                    shuffle= False)
```

Feature Scaling

```
In [ ]: from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.transform(X_test)
```

Note: Using binarization [1 0 0] is used as y value when the return label is -1, [0 1 0] is used as y value when the return label is 0, and [0 0 1] is used as y value when the return label is 1.

1. Select an algorithm or group of algorithms (for example, neural networks, deep learning, ARMA, ensemble techniques).

We select the SVM One vs Rest Classifier as the algorithm and use the code below:

SVM classifier

```
In [90]: from sklearn.metrics import roc_curve, auc
from sklearn.metrics import confusion_matrix
from sklearn.multiclass import OneVsRestClassifier
from sklearn import svm

classifier = OneVsRestClassifier(svm.SVC(kernel='rbf', probability=True,
                                 random_state=0))

model_SVC = classifier.fit(X_train, y_train)

#Returns the decision function of the sample for each class in the model.
y_score = model_SVC.decision_function(X_test)

#Perform classification on the samples
y_pred_svm = model_SVC.predict(X_test)
```

2. Fit the model: show that it works out of sample, and use appropriate validation techniques.

We have fitted the model above and we will use ROC curves, Confusion Matrix, Precision – recall curves, F1-Score, Accuracy, and AUC as the validation techniques.

As we will see in the report below (in point number 4), the AUC for both the ROC curve and the Precision – recall curve, the Confusion Matrix, , the F1-Score, and the Accuracy are satisfactory for each return label (-1, 0, or 1). Hence we are satisfied with our model.

3. For the model, provide the following performance metrics:

a. ROC curves

b. Confusion Matrix

c. Precision, recall, F1-Score, Accuracy, and AUC

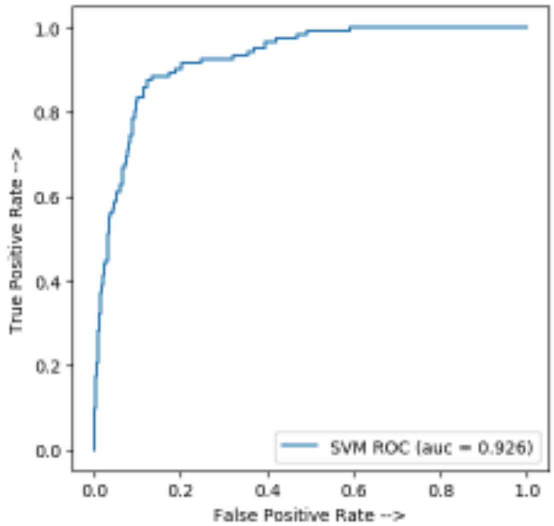
We provide these using the code below:

```
In [91]: from sklearn.metrics import precision_recall_curve
         from sklearn.metrics import f1_score
         from sklearn.metrics import accuracy_score

         fpr = dict()
         tpr = dict()
         roc_auc = dict()
         precision = dict()
         recall = dict()
         precisionrecall_auc = dict()

         for i in range(3):

             fpr[i], tpr[i], threshold1 = roc_curve(y_test[:, i], y_score[:, i])

             roc_auc[i] = auc(fpr[i], tpr[i])

             precision[i], recall[i], threshold2 = precision_recall_curve(y_test[:, i],
                                                                  y_score[:, i])
             precisionrecall_auc[i] = auc(recall[i], precision[i])

             print("For y label",i-1, " : ")

             #ROC curves
             print("ROC curves")
             plt.figure(figsize=(5, 5), dpi=100)
             plt.plot(fpr[i], tpr[i], linestyle='-', label='SVM ROC (auc = %0.3f)' % roc_auc[i])

             plt.xlabel('False Positive Rate -->')
             plt.ylabel('True Positive Rate -->')

             plt.legend()

             plt.show()

             # creating a confusion matrix
             cm = confusion_matrix(y_test[:, i], y_pred_svm[:, i])
             print("Confusion matrix")
             print(cm)


             #Precision Recall curves
             print("Precision Recall curves")
             plt.figure(figsize=(5, 5), dpi=100)
             plt.plot(recall[i], precision[i], linestyle='-',
                      label='SVM Precision Recall (auc = %0.3f)' %  precisionrecall_auc[i])

             plt.xlabel('Recall')
             plt.ylabel('Precision')
             plt.legend()
             plt.show()

             #F1 score
             f1 = f1_score(y_test[:, i], y_pred_svm[:, i])
             print("F1 score: ", f1)

             #Accuracy
             accuracy = accuracy_score(y_test[:, i], y_pred_svm[:, i])
             print("Accuracy: ", accuracy)

             #AUC
             print("AUC for ROC curve: " ,roc_auc[i])
             print("AUC for Precision-Recall curve: " ,precisionrecall_auc[i])

             print()   #new line before showing the results for the next y label
```

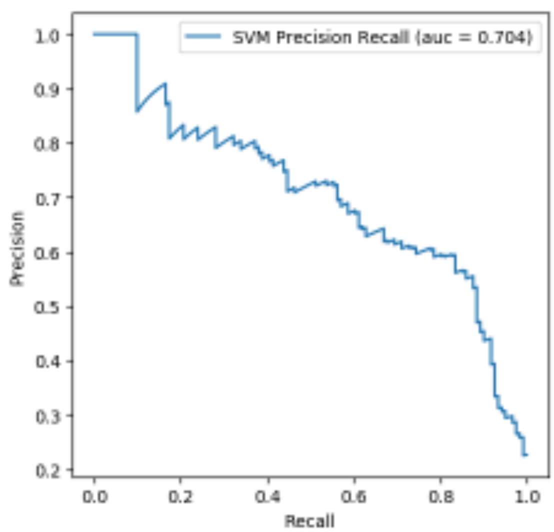
4. Analyze the metrics and develop a report

The report of the above metrics is below:

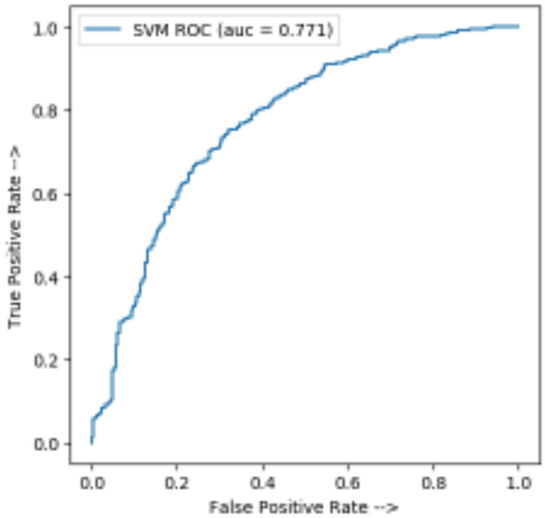
```
For y label -1  :
ROC curves
```



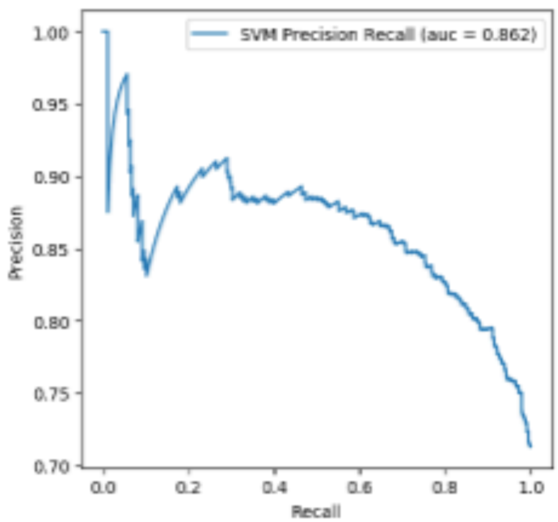

```
Confusion matrix
[[696   3]
 [101  20]]
Precision Recall curves
```



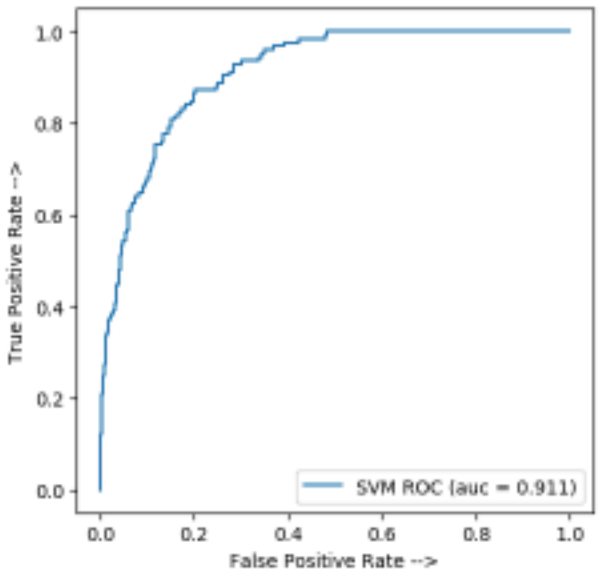

```
F1 score:  0.2777777777777778
Accuracy:  0.8731707317073171
AUC for ROC curve:  0.9261991747360455
AUC for Precision-Recall curve:  0.7038912955484882

For y label 0  :
ROC curves
```



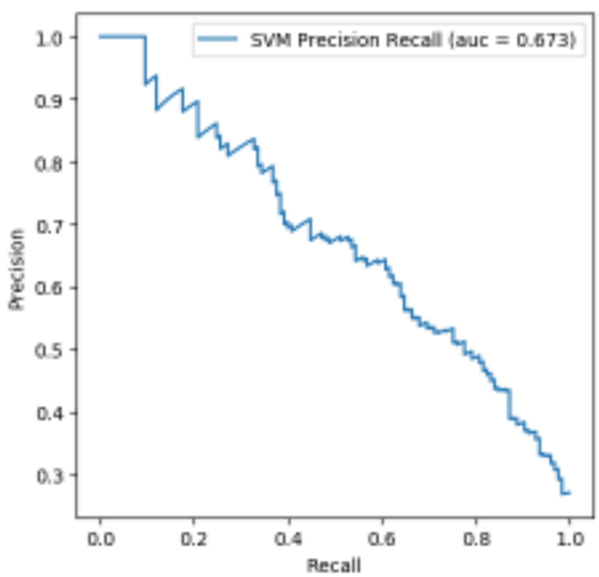

```
Confusion matrix
[[ 28 218]
 [  4 570]]
Precision Recall curves
```




```
F1 score:  0.8370044052863437
Accuracy:  0.7292682926829268
AUC for ROC curve:  0.771090054106116
AUC for Precision-Recall curve:  0.8622353190732959

For y label 1  :
ROC curves
```




```
Confusion matrix
[[690   5]
 [ 98  27]]
Precision Recall curves
```




```
F1 score:  0.34394904458598724
Accuracy:  0.874390243902439
AUC for ROC curve:  0.9111827338129496
AUC for Precision-Recall curve:  0.6733474327764624
```

From the above we can see that the AUC for both the ROC curve and the Precision – recall curve, the Confusion Matrix, ,the F1-Score, and the Accuracy are satisfactory for each return label (-1, 0, or 1).

5. Create a fund factsheet for our new investment strategy

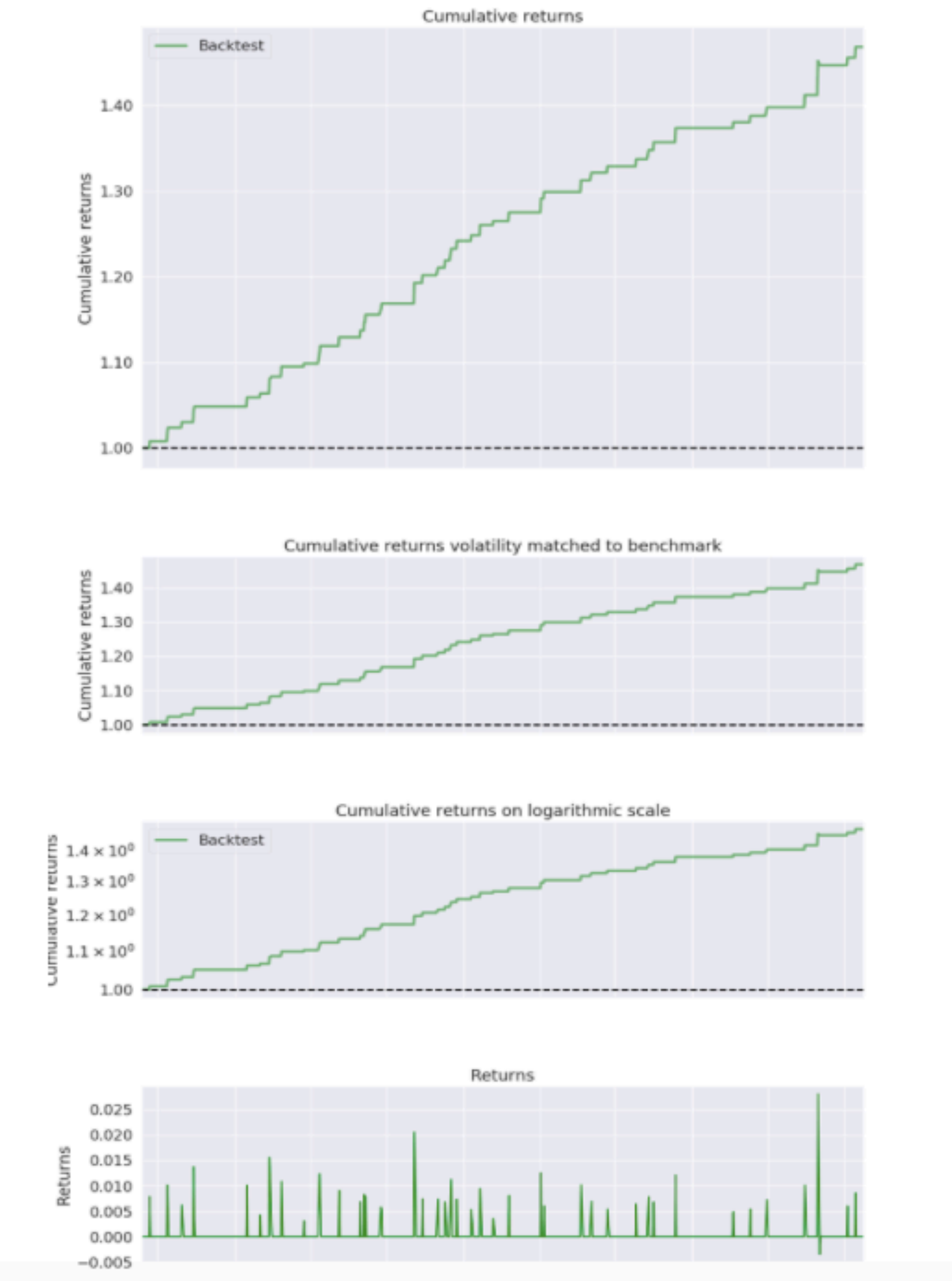
We develop the fund factsheet using the code below:

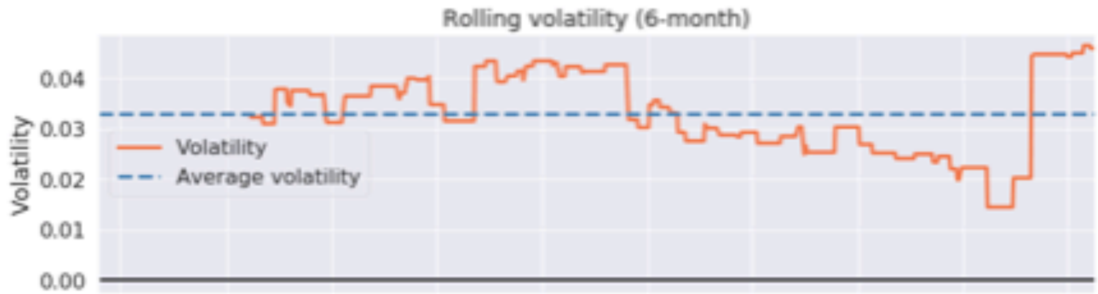
In [98]:
```
import pyfolio as pf
import warnings
from sklearn import preprocessing


warnings.filterwarnings('ignore')

# Binarize labels in a one-vs-all fashion
lb = preprocessing.LabelBinarizer()

lb.fit_transform([-1, 0, +1])

#choosing y_hat value to be the one which has the highest corresponding y_score
#This is to uncover the predicted return label (-1,0, or 1) from the decision function
#score that we uncover from the SVM

y_hat = lb.inverse_transform(y_score)


# Getting last 20% of USDEUR Return data to backtest performance of our trading strategy

Return20 = Return.tail(len(y_hat))

#Calculating the return of our strategy which goes long (+1), short (-1), or stays
#flat on the USDEUR asset (Long and short are always on equal units)

StrategyReturn = y_hat*Return20

import pyfolio as pf
import warnings
warnings.filterwarnings('ignore')

#Creating the Fund Factsheet
pf.create_returns_tear_sheet(StrategyReturn)
```

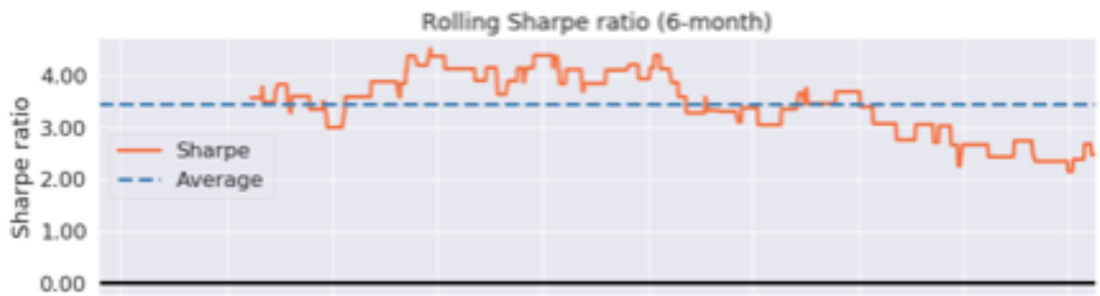
The fund factsheet generated is below:

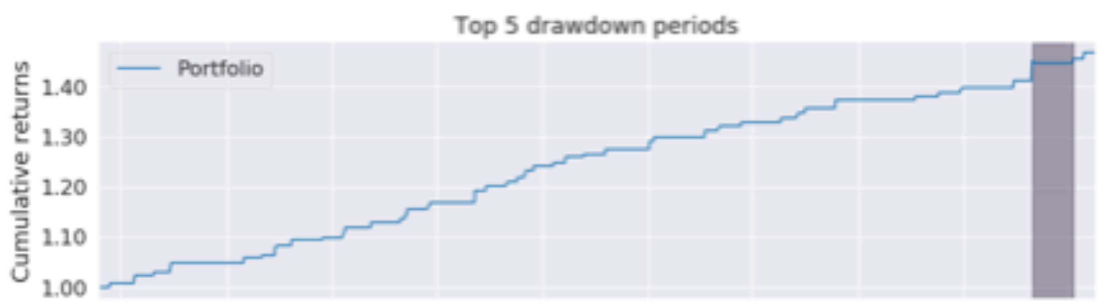
Start date	2017-04-06
End date	2020-06-01
Total months	39
	Backtest
Annual return	12.5%
Cumulative returns	46.8%
Annual volatility	3.5%
Sharpe ratio	3.36
Calmar ratio	36.07
Stability	0.98
Max drawdown	-0.3%
Omega ratio	112.20
Sortino ratio	61.64
Skew	6.09
Kurtosis	48.00
Tail ratio	inf
Daily value at risk	-0.4%

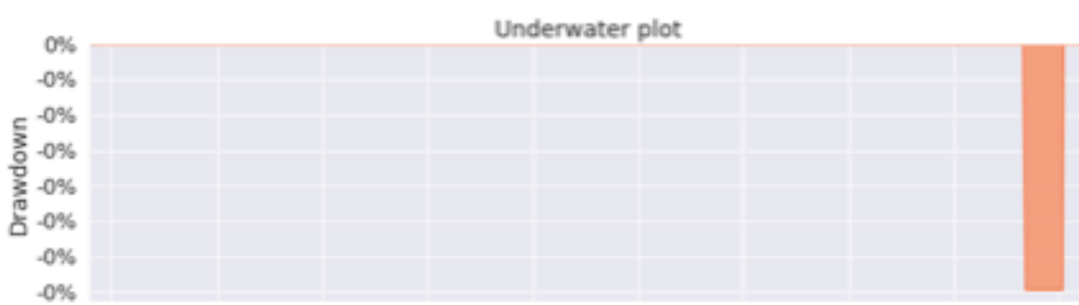
Worst drawdown periods	Net drawdown in %	Peak date	Valley date	Recovery date	Duration
0	0.35	2020-03-20	2020-03-23	2020-05-06	34
1	0.00	2017-04-06	2017-04-06	2017-04-06	1
2	0.00	2017-04-06	2017-04-06	2017-04-06	1
3	0.00	2017-04-06	2017-04-06	2017-04-06	1
4	0.00	2017-04-06	2017-04-06	2017-04-06	1


Cumulative returns
Backtest
Cumulative returns
1.40
1.30
1.20
1.10
1.00
Cumulative returns volatility matched to benchmark
Cumulative returns
1.40
1.30
1.20
1.10
1.00
Cumulative returns on logarithmic scale
Backtest
Cumulative returns
1.4×10^0
1.3×10^0
1.2×10^0
1.1×10^0
1.00
Returns
Returns
0.025
0.020
0.015
0.010
0.005
0.000
−0.005


Rolling volatility (6-month)
Volatility
0.04
0.03
0.02
0.01
0.00
Volatility
Average volatility


Rolling Sharpe ratio (6-month)
Sharpe ratio
4.00
3.00
2.00
1.00
0.00
Sharpe
Average


Top 5 drawdown periods
Cumulative returns
1.40
1.30
1.20
1.10
1.00
Portfolio


Underwater plot
Drawdown
0%
-0%
-0%
-0%
-0%
-0%
-0%
-0%

The results above show an Annual return of 12.5% and Cumulative Return of 46.8% for the trading strategy over the backtest time period.

Annual Volatility – measures the degree of variation of a trading price series over a period of time and is measured by the standard deviation of logarithmic returns. Here the value is 3.5%.

Sharpe Ratio – gauges investment performance by adjusting for its risk. Measuring the risk-adjusted performance of our trading strategy, the model above has produced a Sharpe Ratio of 3.36. The grading thresholds for Sharpe ratio are: less than 1 → bad, 1 – 1.99 → adequate/good, 2 – 2.99 → very good and greater than 3 → excellent. The results above show a very impressive Sharpe Ratio of 3.36.

Calmar ratio - is a comparison of the average annual compounded rate of return and the maximum drawdown risk of commodity trading advisors and hedge funds. The lower the Calmar ratio, the worse the investment performed on a risk-adjusted basis over the specified time period; the higher the Calmar ratio, the better it performed. Here the value is an impressive 36.07.

Max drawdown – a drawdown is a reduction of one's capital after a series of losing trades. The max drawdown of -0.3% reflects the maximum equity loss experienced in this portfolio. It can be calculated as

$$\frac{\text{EquityPeakHigh} - \text{EquityTroughLow}}{\text{EquityPeakHigh}}$$

The max drawdown results help give potential investors a better idea of how a worst case scenario may be like in terms of the greatest loss over a specific time period.

Omega ratio - is defined as the probability weighted ratio of gains versus losses. The higher the Omega ratio for the given threshold, the more desirable the strategy. Here the omega ratio is 112.2 which makes it very desirable.

Sortino ratio - The Sortino ratio differentiates harmful volatility from total volatility by using the asset's standard deviation of negative portfolio returns instead of the total standard deviation of portfolio return. It is defined as

$$\text{Sortino Ratio} = \frac{R_p - r_f}{\sigma_d}$$

R_p=Actual or expected portfolio return, r_f = Risk-free rate σ_d =Standard deviation of the downside
For this strategy, the value is an impressive 61.64.

Skewness – refers to level of asymmetry in an otherwise symmetrical distribution. A normal distribution should have a skewness = 0 where the mean = the median. However, if the majority of the returns are positive rather than negative, a large right tail and a positive skewness will result, with the mean > median. The results above show a skew of 6.09. The return distribution has a positive skewness which is desirable as it posits a greater chance of realizing a large positive return than a negative return for a risk-averse investor.

Kurtosis – measures the peakedness (or flatness) of the distribution. A normal distribution has a kurtosis of 3. Here the kurtosis is 48.

Tail ratio - is the ratio between the 95th and the absolute value of the 5th percentile of the daily returns distribution. Here instead of the 5th percentile being negative it is 0, which is why the tail ratio comes out as infinite.

Daily Value at risk – calculates the maximum loss expected on an investment on a daily basis at a 95% confidence level. This is only -0.4% in this case.

Conclusion

We have used various technical indicators over the time frame 2004-04-05 to 2017-04-06 to come up with a trading strategy for the USDEUR FX asset which we backtest over the time frame from 2017-04-06 to 2020-06-01.

We denoted the strategy using return labels (+1 indicating we will go long the asset if it is positive and greater than 90 day standard deviation, -1 indicating we will go short the asset if it is negative and less than -1*90 day standard deviation, and 0 otherwise indicating we will not take a position in the asset on that day). All positions are taken at the beginning of the trading day and closed at the end of the trading day.

We binarized the return label y, did a train test split, and applied feature scaling to the explanatory variables. We then used the SVM One vs Rest Classifier and found in our cross validation that the AUC for both the ROC curve and the Precision – recall curve, the Confusion Matrix, , the F1-Score, and the Accuracy are satisfactory for each return label (-1, 0, and 1).

Our backtest results showed that our trading strategy has minimal drawdown and an annual return of 12.5% with impressive Sharpe, Calmar, Omega, Tail, and Sortino ratios.
We created a fund factsheet with the details of our strategy backtest results using Pyfolio.

References

1) Bengio, Y., Courville, A. and Vincent, P., 2013. 'Representation learning: A review and new perspectives'. IEEE Transactions on Pattern Analysis and Machine Intelligence, 35(8), p. 1798.
2) Brownlee, J. (2016). Supervised and Unsupervised Machine Learning Algorithms. [online] Available at: https://machinelearningmastery.com/supervised-and-unsupervised-machine-learning-algorithms/
3) Lopéz de Prado, M. (2018). Advances in Financial Machine Learning. Wiley Publishers.
4) McCarthy, J., Minsky, M.L., Rochester, N. and Shannon, C.E. (1955). A Proposal for the Dartmouth Summer Research Project on Artificial Intelligence, Dartmouth proposal.
5) Scikit-learn. (2017). Scikit-learn User Guide. [online], Available at: http://scikit-learn.org/stable/user_guide.html
6) Bishop, C.M. (2007). Pattern Recognition and Machine Learning (Information Science and Statistics). Springer.
7) Federal Reserve Bank of St Louis. Available at https://fred.stlouisfed.org/
8) Key ECB interest rates. Available at https://www.ecb.europa.eu/stats/policy_and_exchange_rates/key_ecb_interest_rates/html/index.en.html
9) Principal Component Analysis with Python. Available at: https://www.geeksforgeeks.org/principal-component-analysis-with-python/
10) Brownlee, J (2018). Basics of Linear Algebra for Machine Learning (Discover the Mathematical Language of Data in Python), Machine Learning Mastery
11) Strategy Optimisation with Walk Forward Analysis. Available at: https://www.prorealcode.com/blog/learning/strategy-optimisation-walk-analysis
12) Sanjay, M(2018). Why and how to Cross Validate a Model? Available at: https://towardsdatascience.com/why-and-how-to-cross-validate-a-model-d6424b45261f
13) Hayes, Adam (2020). Stochastic Oscillator Definition. Available at: https://www.investopedia.com/terms/s/stochasticoscillator.asp
14) Rambo, James (2017). An Omega Ratio Analysis Of Global Hedge Fund Returns