GROUP ASSIGNMENT No. 2

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	gadzairrobert@gmail.com	

Objective: to determine which features are helpful in predicting the USD/EUR financial asset

Introduction

In this assignment we continue our analysis on the USD/EUR financial asset. Specifically we do the following:

1. Select at least four explanatory variables and perform the necessary transformations so that they are useful in the model phase. Investigate feature engineering techniques such as PCA and encoding target variables using one-hot encoding.
2. Write a short paragraph about each technique investigated and show its implementation in Python using a Jupyter Notebook. We include appropriate references that indicate where the ideas were sourced.
3. Write a 300 -500 words paragraph explaining each of the below cross-validation techniques: traditional k-fold cross-validation, walk forward analysis, and Purged K-Fold CV.

We analyse the close price data for USD/EUR over the past 1 year for this purpose.

1. Select at least four explanatory variables and perform the necessary transformations so that they are useful in the model phase. Investigate feature engineering techniques such as PCA

We first import the relevant libraries and download our data:

The library **yfinance** (Yahoo! Finance market data downloader) is required to pull stock prices from internet. We install it by applying the below command via terminal.

```
pip install yfinance --upgrade --no-cash-dir
```

Also, to downloadthe Effective Federal Funds Rate from https://fred.stlouisfed.org/series/FEDFUNDS , we install the library.

```
pip install fredapi
```

We then import the below required libraries.

```
import yfinance as yf
from fredapi import Fred
import numpy as np
import pandas as pd
```

```
import matplotlib.pyplot as plt
```

- We pull USD/EUR FX asset data from yahoo finance.

```
data_frame = yf.download('USDEUR=X', start='2019-05-06', end='2020-05-06, progress=False)
data_frame.drop(['Open', 'Volume', 'Adj Close'], axis=1, inplace=True)
```

- Then we will need to download Fed interest rate in the same way.

```
fred = Fred(api_key='267a686a16fe42c7b7defdc69c762c7c')
IRdf = fred.get_series('DFF', '5/6/2019', '5/6/2020') # mm/dd/yyyy
IRdiff=IRdf
```

The variables we use are below:

I. The Difference in US and Euro interest rate

- is now calculated as the first explanatory variable.

$$Fed\ funds\ rate - ECB\ refinancing\ rate$$

- Since ECB rate was found zero during the same period, the equation reduces to fund rate.

II. The Stochastic Oscillator "%K"

- A stochastic oscillator [13] is a momentum indicator comparing a particular closing price of a security to a range of its prices over a certain period of time. It is used to generate overbought and oversold trading signals, utilizing a 0-100 bounded range of values []. It can be calculated as follows: -

$$\%K = \left(\frac{C - L14}{H14 - L14}\right) \times 100$$

$C = The\ most\ recent\ closing\ price$

$L14 = The\ lowest\ price\ traded\ of\ the\ 14\ previous\ trading\ sessions$

$H14 = The\ highest\ price\ traded\ of\ the\ same\ 14\ previous\ trading\ sessions$

- The code is below:

```
def StochasticOsclillator(data):
```

```
    temp_df['LowMin'] = data['Low'].rolling(window=14).min()
    temp_df['HighMax'] = data['High'].rolling(window=14).max()
    return (data['Close']-temp_df['LowMin'])/(temp_df['HighMax']-temp_df['LowMin'])

data_frame['StochasticOsclillator'] = StochasticOsclillator(data_frame)
```

III. The 3-period moving average of %K Stochastic Oscillator =%D

- This moving average is given by

$$\%D_T = \frac{\%K_T + \%K_{T-1} + \%K_{T-2}}{3}$$

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

IV. The 3-period moving average of %D

- This moving average is given by

$$\%Slow\ D_T = \frac{\%D_T + \%D_{T-1} + \%D_{T-2}}{3}$$

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

V. The Senkou Span A

- This indicator is given by [10]

$$Senkou\ A = \frac{Conversion\ line + base\ line}{2}$$

$$Conversion\ line = \frac{9\ period\ high + 9\ period\ low}{2}$$

$$base\ line = \frac{26\ period\ high + 26\ period\ low}{2}$$

- This indicator is coded as below:

```
def Line_trend(data, w):
  return (data['High'].rolling(window=w).max()
```

```
        + data['Low'].rolling(window=w).min())/2

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
```

VI. Historical 4-day trend

- This indicator is given by

$$Trend_4 = \frac{Closing\ price_T}{Closing\ price_{T-4}} - 1$$

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

data_frame['Historical 4-day trend'] = Historical_D_days_trend(data_frame['Close'], 4)
```

VII. Historical 8-day trend

- This indicator is given by

$$Trend_8 = \frac{Closing\ price_T}{Closing\ price_{T-8}} - 1$$

```
data_frame['Historical 8-day trend'] = Historical_D_days_trend(data_frame['Close'], 8)
```

VIII. Historical 16-day trend

- This indicator is given by

$$Trend_{16} = \frac{Closing\ price_T}{Closing\ price_{T-16}} - 1$$

```
data_frame['Historical 16-day trend'] = Historical_D_days_trend(data_frame['Close'], 16)
```

IX. Historical 32-day trend

- This indicator is given by

$$Trend_{32} = \frac{Closing\ price_T}{Closing\ price_{T-32}} - 1$$

```
data_frame['Historical 32-day trend'] = Historical_D_days_trend(data_frame['Close'], 32)
```

The first few rows of our technical indicators are below:

Date	High	Low	Close	StochasticOsclillator	3-period MA of %K %D	3-period MA of %D	ConvLine	BaseLine	SenkouSpanA	Historical 4-day trend	Historical 8-day trend	Historical 16-day trend	Historical 32-day trend
2019-06-19	0.89376	0.89110	0.89300	0.672364	0.590422	0.471410	0.887735	0.890665	0.88920	0.007344	0.010890	-0.002458	-0.001107
2019-06-20	0.89001	0.88377	0.88980	0.585832	0.597055	0.550221	0.887935	0.890665	0.88930	-0.001907	0.007085	-0.008999	-0.003260
2019-06-21	0.88624	0.88235	0.88530	0.313693	0.523963	0.570480	0.887935	0.890665	0.88930	-0.006286	0.003002	-0.014560	-0.009255
2019-06-24	0.87921	0.87710	0.87837	0.073966	0.324497	0.481838	0.885685	0.888615	0.88715	-0.016383	-0.007828	-0.018504	-0.016372
2019-06-25	0.87886	0.87612	0.87717	0.057854	0.148504	0.332322	0.885195	0.888125	0.88666	-0.014194	-0.010513	-0.013507	-0.015632
2019-06-26	0.88099	0.87800	0.87951	0.186777	0.106199	0.193067	0.885195	0.888125	0.88666	-0.006540	-0.013449	-0.010196	-0.011742
2019-06-27	0.88118	0.87850	0.87914	0.166394	0.137008	0.130571	0.885195	0.888125	0.88666	0.000877	-0.013200	-0.012812	-0.012502

2. Write a short paragraph about each technique investigated and show its implementation in Python using a Jupyter Notebook.

Short writeup on PCA:

Principal Component Analyis is defined as a statistical procedure to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables. In other words, it is an unsupervised, non-parametric statistical technique used primarily for dimensionality reduction in machine learning.
Each of the principal components is chosen in such a way so that it would describe the maximum variance left over in the data (after accounting for the variance described by the principal components chosen before it) and all the principal components are chosen to be orthogonal to each other.
PCA is used to filter noisy datasets, such as image compression, interpret and visualize data, find inter-relation between variables in a dataset and to visualize genetic distance and relatedness between populations.
The typical goal of PCA is to reduce the dimensionality of the original feature space by projecting it onto a smaller subspace, where the eigenvectors will form the axes
The use of PCA as a dimension reduction process and a non-dependent procedure aims to reduce attribute space from a large number of variables to a smaller number of factors.

How we applied PCA here:

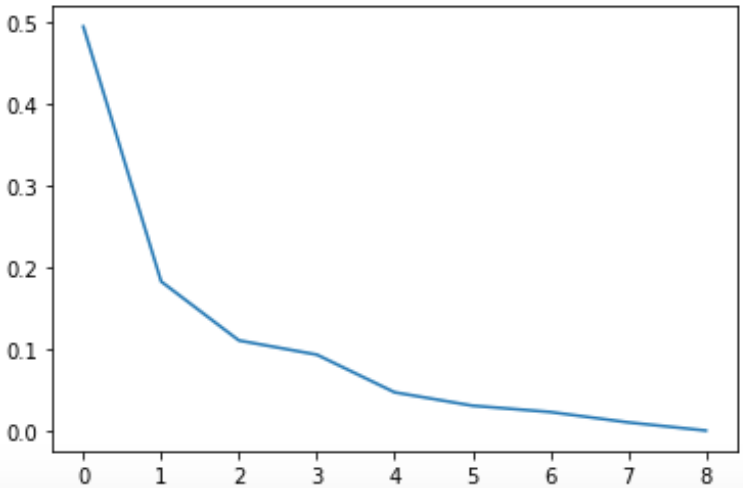
We need to omit the first 32 days to eliminate the Nulls ('NaN') and then run the PCA code by normalizing the explanatory variables by respective means and variance.

```
df = data_frame [31:] # remove NaN entries
from sklearn.decomposition import PCA
```

```
from sklearn.preprocessing import StandardScaler

pca = PCA(n_components=9)

PCAData = pd.DataFrame(list(zip(IRdiff,df['StochasticOsclillator'],
                    df['3-period MA of %K %D'],
                    df['3-period MA of %D'],
                    df['SenkouSpanA'],
                    df['Historical 4-day trend'],
                    df['Historical 8-day trend'],
                    df['Historical 16-day trend'],
                    df['Historical 32-day trend'])))

PCAData = StandardScaler().fit_transform(PCAData)

pca.fit_transform(PCAData)

print(pca.explained_variance_ratio_)
plt.plot(range(0, 9), pca.explained_variance_ratio_)
```

- As we can see the first 6 principal components are responsible for the majority of the variation (>95%) in the explanatory variables.

```
[0.49441181 0.18333965 0.11133997 0.09407832 0.04808811 0.03182073
 0.02407789 0.01143838 0.00140516]

Out[15]: [<matplotlib.lines.Line2D at 0x1a20b7e5f8>]
```



- We get the main 6 principal components using the code below:

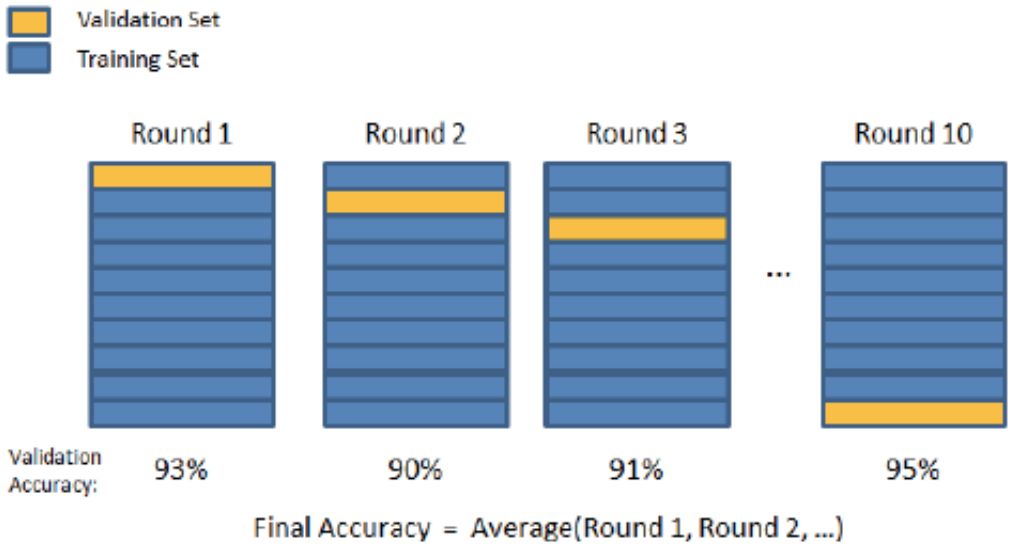
```
pca = PCA(n_components=6)

pca.fit_transform(PCAData)

print(np.round(pca.components_,2))
```

```
[[-0.02  0.43  0.44  0.4   0.09  0.27  0.39  0.37  0.31]
 [-0.57 -0.13 -0.05  0.05  0.71 -0.24 -0.16  0.08  0.25]
 [-0.15 -0.05 -0.31 -0.49  0.11  0.71  0.26 -0.05  0.22]
 [-0.68  0.22  0.16  0.07 -0.21  0.15  0.09 -0.25 -0.56]
 [-0.01 -0.06 -0.12 -0.1   0.04  0.14 -0.22  0.83 -0.45]
 [-0.04 -0.39 -0.16  0.02  0.01 -0.32  0.82  0.13 -0.16]]
```

3. Write a 300 -500 words paragraph explaining each of the below cross-validation techniques: traditional k-fold cross-validation, walk forward analysis, and Purged K-Fold CV.

K-fold cross-validation

Removing a part of the training data for validation poses a problem of underfitting. This is because when we reduce the size of the training data, we risk losing important patterns/ trends in data set. This in turn increases error induced by bias.
This method provides the solution to this problem through repeated holdout - we average the scores after K different holdouts. Every data point gets to be in a validation set exactly once and gets to be in a training set K-1 times. This significantly reduces both underfitting and overfitting as each data point is being used both in training and validation set.



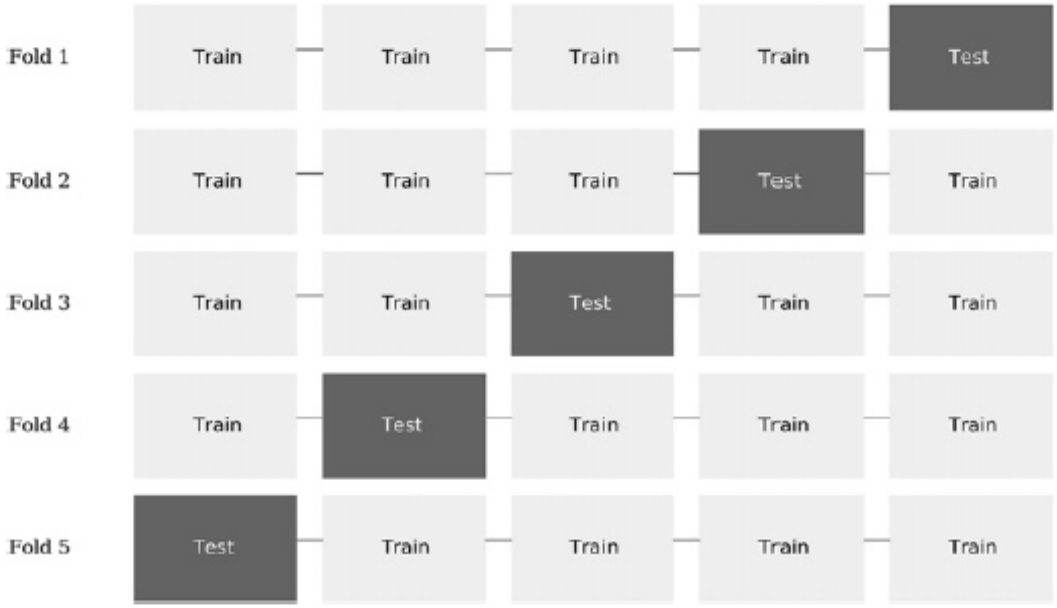
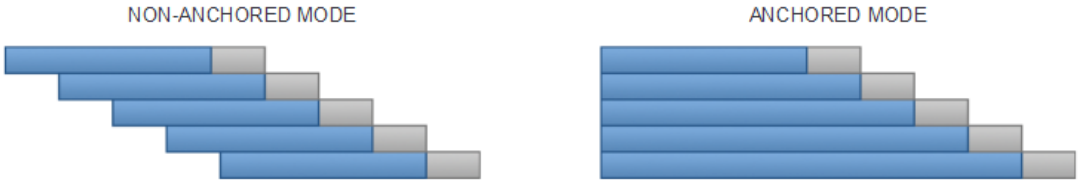


FIGURE 7.1 Train/test splits in a 5-fold CV scheme

Walk forward analysis

Walk forward analysis optimizes a set of variables on an initial period (in-sample data). It then tests the best parameters on the following period (out of sample data), and repeats the process by shifting forward the time windows. The in-sample optimization periods may have a single starting point (in which case it is called Anchored mode) or different starting points (in which case it is called Non-anchored mode).



By comparing the results of these 2 periods, the method allows us to:

- Check that the performance of the strategy on the optimization periods (in-sample data shown in blue) are consistent with the test periods (out-of-sample data shown in grey). This is to avoid the risk of "overfitting"
- Check strategy behavior as market conditions change
- Use past data to test the strength of the strategy

Purged K-Fold CV

One of the reasons why k-fold CV fails in finance is that observations cannot be assumed to be drawn from an IID process.

For instance :Because of serial correlation, $X_t \approx X_{t+1}$ and because labels are derived from data points overlapping in time, $Y_t \approx Y_{t+1}$.

By placing t and t + 1 in different sets (training and testing), information is leaked.
When a classifier is first trained on (X_t , Y_t), and then it is asked to predict $E[Y_{t+1} \mid X_{t+1}]$ based on an observed X_{t+1}, this classifier is more likely to achieve $Y_{t+1} = E[Y_{t+1} \mid X_{t+1}]$ even if X is an irrelevant feature.

Purged k-fold CV reduces leakage by purging from the training set all observations whose labels overlapped in time with those labels included in the testing set.
A label $Y_i = f [[t_{i,0} , t_{i,1}]]$ overlaps with Y_j if any of the three sufficient conditions is met:
1. $t_{j,0} \leq t_{i,0} \leq t_{j,1}$
2. $t_{j,0} \leq t_{i,1} \leq t_{j,1}$
3. $t_{i,0} \leq t_{j,0} \leq t_{j,1} \leq t_{i,1}$

Through this implementation Purged k-fold CV reduces leakage caused by k-fold CV.

Conclusion

In our submission we investigated feature selection and engineering techniques useful in predicting a target variable. We explored several explanatory variables and
using PCA figured out the 5 components that explained close to 95% of the variation in those explanatory variables. Finally, we gave a write up on three cross validation techniques: traditional k-fold cross-validation, walk forward analysis, and Purged K-Fold CV.

We plan to implement a machine learning algorithmic trading strategy on this asset in Submission 3 (Modelling and Strategy Development) of this course.

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