Summary

The purpose of this project is to evaluate the performance for two trading strategies, one using the volatility index VIX value limits for trading decisions over S&P500 index and the other with a multivariate forecasting neural network using the volatility index VIX, S&P500 index and S&P500 index futures.

Some scenarios will be presented in favor of understanding how the return performance will be impact if a variable is changed for the first strategy or if the training dataset size increase or decrease for the second strategy.

What is volatility and implied volatility?

Volatility provide a measure of risk because its value presents the degree of uncertainty surrounding a security future returns. The standard deviation calculated from the returns of a stock withing a historical period of time is considered as a good volatility approximation but there are other methodologies.

implied volatility is a forward-looking expectation of volatility of an underlying security, it is computed from the back out estimation of future standard deviation from a model of live option prices. By knowing this figure is possible to what market believes future volatility will be like on an asset until option's maturity.

VIX index & market sentiment

The CBOE volatility index VIX is also known as the fear because if it has large value usually the market prices go down, this index tells the implied volatility derived from options on S&P500 index (an index for the largest 500 stocks in USA), VIX figures are based on expectation for S&500 performance therefor it has information about the market sentiment.

The calculation of the VIX have been changing across the time in order to obtain more accurate perception of the market, initially in 1993 it was computed from eight S&P100 at the money put and call options, since 2003 the methodology was updated and wider range of options were included.

Data gathering

The data was obtained from yahoo finance by using the API in python, the original dataset contains the daily close value for S&P500, VIX and S&P500 futures from January 2022 to June 2022 for the VIX index rule and from January 2019 to June 2022 for the LTSM multivariate model.

Trading algorithms

1. **Strategy VIX index rule**

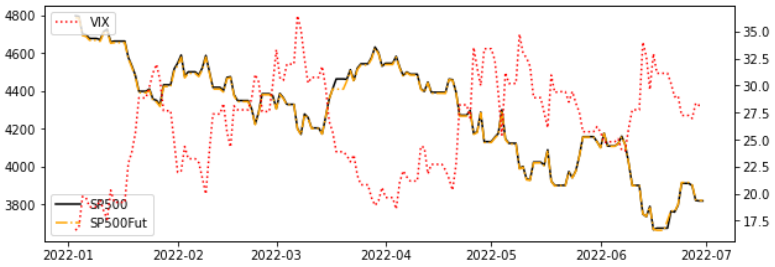
Using a long only trading operations the script takes the VIX index values as an input for decide whether buy or sell on S&P500, buy when VIX values crosses the upper limit of 30 and sell when VIX values turn under 20, A high value in VIX index is characterized to fear in market and a bearish trend in contrast a small value represents a calm market and a bullish trend.

Other condition is the take profit limit of 5% and a stop loss limit of -5%, this rule triggers the scrip decision for close a current position which means sell.

The following libraries were used for developing the strategy:

```
import yfinance as yf
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
```

The strategy was evaluated in a time window from January 1st 2022 to June 30th 2022, this whole period the market was attached to a high volatility contributed by the Russia-Ukraine war and the FED decisions for stopping inflation in USA.



Initially a strategy dataframe is created with the input data (Date, S&P500, VIX, S&P500 futures) plus other columns, the additional columns store the returns and signals when the limits are crossed.

```
Stgy_DF['Ret']=Stgy_DF['Close_SP']/Stgy_DF['Close_SP'].shift(1)-1
Stgy_DF['SigVixg30']=(0+(np.sign(Stgy_DF['Close_VIX']-LimVixUp)==1))
Stgy_DF['SigVixs20']=(0+(np.sign(LimVixDw-Stgy_DF['Close_VIX'])==1))
Stgy_DF['SigRetg+5']=np.nan
Stgy_DF['SigRets-5']=np.nan
Stgy_DF['InvestVal']=np.nan
Stgy_DF['InvestOp']=np.nan
Stgy_DF['StrgRet']=0
Stgy_DF['StrgDecitions']=np.nan
```

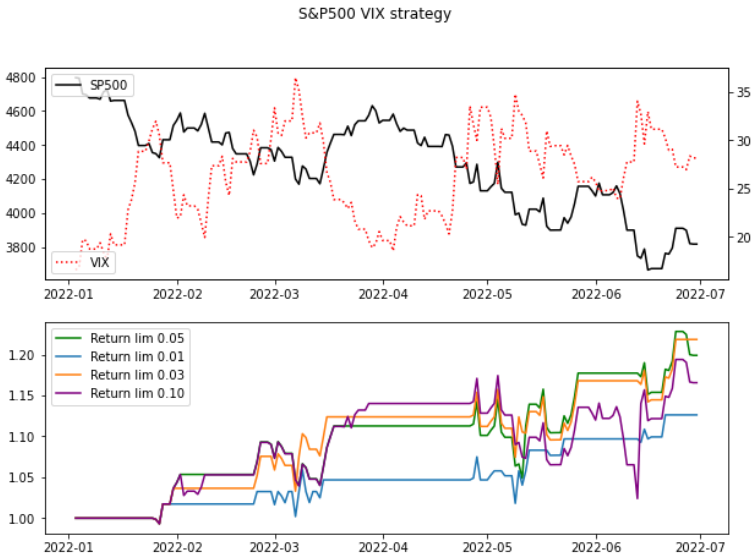
All the strategy is contained in a loop that will go through the strategy dataframe filling the empty cells according to the defined rules. The first investment is 1 unitary money value and this figure will be impacted by the returns only when the buy signal is active, if the signal for selling is active the investment value will keep its amount from the last buy signal, the main parts in the loop are listed below:

- Recurrent actions based on previous signals t-1
- Strategy return calculation
- Action for buy when previous signal is selling or first buy
- Stop loss signals calculations
- Action for buy when previous signal is buying

```
DecCount=0
for i in range(len(Stgy_DF)):
  #Recurrent actions based on previous signals
  if i>0:
    if  np.isnan(Stgy_DF.loc[i,'InvestOp'])!=True and Stgy_DF.loc[i-1,'StrgDecitions']=='sell':
      Stgy_DF.loc[i,'StrgDecitions']='sell'
      Stgy_DF.loc[i,'InvestVal']=Stgy_DF.loc[i-1,'InvestVal']
    elif np.isnan(Stgy_DF.loc[i,'InvestOp'])==True and Stgy_DF.loc[i-1,'StrgDecitions']=='sell':
      Stgy_DF.loc[i,'StrgDecitions']='sell'
      Stgy_DF.loc[i,'InvestOp']=Stgy_DF.loc[i-1,'InvestOp']
      Stgy_DF.loc[i,'InvestVal']=Stgy_DF.loc[i-1,'InvestVal']
    if np.isnan(Stgy_DF.loc[i,'InvestOp'])!=True and Stgy_DF.loc[i-1,'StrgDecitions']=='buy':
      Stgy_DF.loc[i,'StrgDecitions']='buy'
      Stgy_DF.loc[i,'InvestVal']=Stgy_DF.loc[i-1,'InvestVal']*(1+Stgy_DF.loc[i,'Ret'])
    elif np.isnan(Stgy_DF.loc[i,'InvestOp'])==True and Stgy_DF.loc[i-1,'StrgDecitions']=='buy':
      Stgy_DF.loc[i,'StrgDecitions']='buy'
      Stgy_DF.loc[i,'InvestOp']=Stgy_DF.loc[i-1,'InvestOp']
      Stgy_DF.loc[i,'InvestVal']=Stgy_DF.loc[i-1,'InvestVal']*(1+Stgy_DF.loc[i,'Ret'])
  #Strategy returns calculation
  if np.isnan(Stgy_DF.loc[i,'InvestOp'])==True:
    Stgy_DF.loc[i,'StrgRet']=0
  else:
    Stgy_DF.loc[i,'StrgRet']=Stgy_DF.loc[i,'InvestVal']/Stgy_DF.loc[i,'InvestOp']-1
  #Action for buy when previous signal is selling or first buy
  if Stgy_DF.loc[i,'SigVixg30']==1:
    if DecCount==0:
      Stgy_DF.loc[i,'StrgDecitions']='buy'
      Stgy_DF.loc[i,'InvestVal']=1
      Stgy_DF.loc[i,'InvestOp']=Stgy_DF.loc[i,'InvestVal']
      Stgy_DF.loc[i+1,'InvestOp']=Stgy_DF.loc[i,'InvestVal']
      Stgy_DF.loc[i,'StrgRet']=Stgy_DF.loc[i,'InvestVal']/Stgy_DF.loc[i,'InvestOp']-1
      DecCount=+1
    elif DecCount>=0 and Stgy_DF.loc[i-1,'StrgDecitions']=='sell':
      Stgy_DF.loc[i,'StrgDecitions']='buy'
      Stgy_DF.loc[i,'InvestOp']=Stgy_DF.loc[i-1,'InvestOp']
      Stgy_DF.loc[i,'InvestVal']=Stgy_DF.loc[i,'InvestOp']
      Stgy_DF.loc[i+1,'InvestOp']=Stgy_DF.loc[i,'InvestVal']
      Stgy_DF.loc[i,'StrgRet']=Stgy_DF.loc[i,'InvestVal']/Stgy_DF.loc[i,'InvestOp']-1
      DecCount=+1
  # Stop loss signals calculations
  if Stgy_DF.loc[i,'StrgRet']<-0.05:
    Stgy_DF.loc[i,'SigRets-5']=1
  else:
    Stgy_DF.loc[i,'SigRets-5']=0
  if Stgy_DF.loc[i,'StrgRet']>0.05:
    Stgy_DF.loc[i,'SigRetg+5']=1
  else:
    Stgy_DF.loc[i,'SigRetg+5']=0
  # Action for buy when previous signal is buying
  if DecCount>0 and (Stgy_DF.loc[i,'SigVixs20']==1 or Stgy_DF.loc[i,'SigRetg+5']==1 or Stgy_DF.loc[i,'SigRets-5']==1):
    if Stgy_DF.loc[i-1,'StrgDecitions']=='buy':
      Stgy_DF.loc[i,'StrgDecitions']='sell'
      Stgy_DF.loc[i,'InvestOp']=Stgy_DF.loc[i-1,'InvestOp']
      Stgy_DF.loc[i,'InvestVal']=Stgy_DF.loc[i-1,'InvestVal']*(1+Stgy_DF.loc[i,'Ret'])
      Stgy_DF.loc[i+1,'InvestOp']=Stgy_DF.loc[i,'InvestVal']
      Stgy_DF.loc[i,'StrgRet']=Stgy_DF.loc[i,'InvestVal']/Stgy_DF.loc[i,'InvestOp']-1
      DecCount=+1
```

Back testing from January 2022 to June 2022

By evaluating the original strategy, the final return was 19.9% in a time window from January 1st 2022 to June 30th 2022, other scenarios where evaluating by setting the stop lost and the take profit limit from (0.05;-0.05), (0.01; -0.01), (0.03; -0.03), (0.10;-0.10), the strategy that perform with the greatest return of 21.8% is the one with a profit limit of (0.03; -0.03).



Range	+-0.01	+-0.03	+-0.05	+-0.10
Return	0.1262	0.2184	0.199	0.1656

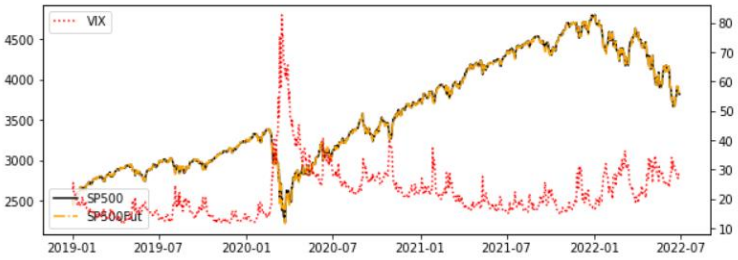
2. Strategy LTSM S&P500 multivariate prediction

The trading procedure follows a buy or sell signal that depends only in the difference between the S&P500 real value and the next day forecast, a multivariate LTISM (long short-term memory) neural network with a single output will provide the next day vale.

For defining and executing the LTSM neural network the algorithm requires some additional libraries for machine learning like Tensor flow keras to define the model and layers and Sklearn which is really useful for scaling the input data to the model, the libraries used are below.

```
import numpy as np
from keras.models import Sequential
from keras.layers import LSTM
from keras.layers import Dense, Dropout
import pandas as pd
from matplotlib import pyplot as plt
from sklearn.preprocessing import StandardScaler
import seaborn as sns
import yfinance as yf
from datetime import datetime, timedelta
```

This model will use past data for training purposes, so the data base is from January 1st 2019 to July 30th 2022 but the strategy execution will always take place between January 1st 2022 to July 30th 2022.



LTSM multivariate model

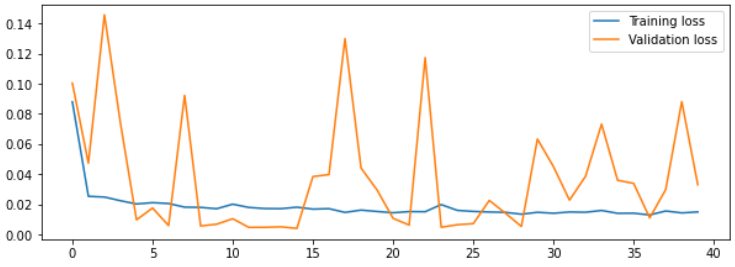
This model has 4 layers in total, the input layer is a LTSM that receives information from 9 past days that corresponds to the 3 different variables (S&P500, VIX, S&P500 futures), the next layer is a LTSM, the next layer is a dropout with a rate of 0.2 followed by a dense layer with one single output. The general parameters are the optimizer as adam and the loss measure as mse.

```
model = Sequential()
model.add(LSTM(64, activation='relu',
input_shape=(trainX.shape[1], trainX.shape[2]),
return_sequences=True))
model.add(LSTM(32, activation='relu',
return_sequences=False))
model.add(Dropout(0.2))
model.add(Dense(trainY.shape[1]))
model.compile(optimizer='adam', loss='mse')
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_18 (LSTM)               (None, 9, 64)             17408

lstm_19 (LSTM)               (None, 32)                12416

dropout_9 (Dropout)          (None, 32)                0

dense_9 (Dense)              (None, 1)                 33

=================================================================
```

In training we have 0.2 for validation split, 40 epochs and a batch size of 7, the dataset training window is from January 1st 2019 to December 31st 2021.

```
history = model.fit(trainX, trainY, epochs=40,
batch_size=7, validation_split=0.2, verbose=1)
```



The returns a prediction for S&P 500 close values, the forecast is more accurate when it is closer to the start date.

Strategy dataframe is created for storing the information in each iteration of the loop.

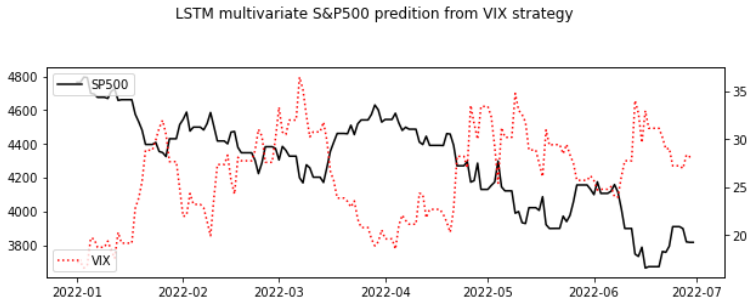
```
StgyDf=StgyPredDf.dropna().copy()
StgyDf.reset_index(inplace=True)
StgyDf.drop(['index'], axis=1,inplace=True)
StgyDf['Ret']=StgyDf['Close_SP']/StgyDf['Close_SP'].
shift(1)-1
StgyDf['Signal']=np.sign(StgyDf['Pred']-
StgyDf['Close_SP'])
StgyDf['InvestVal']=np.nan
StgyDf['InvestOp']=np.nan
StgyDf['StrgRet']=0
StgyDf['StrgDecitions']=np.nan
```

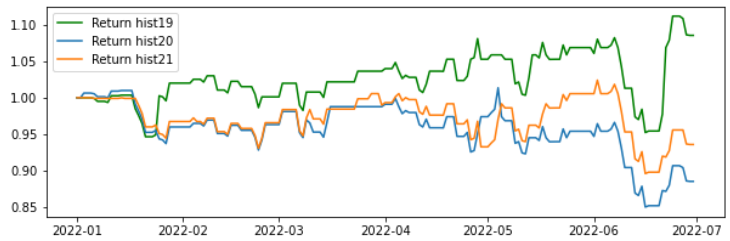
The script uses the S&P 500 next day prediction for a long only trading strategy, the main parts in the loop are listed below:

- Recurrent actions based on previous signals t-1
- Strategy return calculation
- Action for buy when previous signal is selling or first buy
- Action for buy when previous signal is buying

Back testing from January 2022 to June 2022

During the model execution it was necessary to change the size of the dataset for training by taking the data from January 1st 2019, January 1st 2020 and January 1st 2021, when the prediction is not very accurate the model decisions will negatively impact the returns as presented in the chart below.





Range	>2019	>2020	>2021
Return	0.0855	-0.1149	-0.0643

Conclusions

The first strategy which takes VIX input is very efficient in the back-testing time window but the correlations between S&P 500 and VIX might change over the years so is always a good option to find a quantitative procedure for updating the rules limits.

The LTSM forecast in the second strategy did not perform well when the training dataset was only from 1 year back information, its accuracy improves when more data was added, this model have a lot of features for balance and update for future opportunities would be an attractive idea to train a new neural network every 1 or 2 weeks while our strategy is being executed.

References

- [Walter de Gruyter Inc (2019)] Patrick Boyle and Jesse McDougall - Trading and Pricing Financial Derivatives.
- [Cengage Learning (2013)] Jeff Madura - Financial Markets and Institutions