Analysis of Volatility in the Stock Market

December 22, 2021

Anish Singla 1005801401

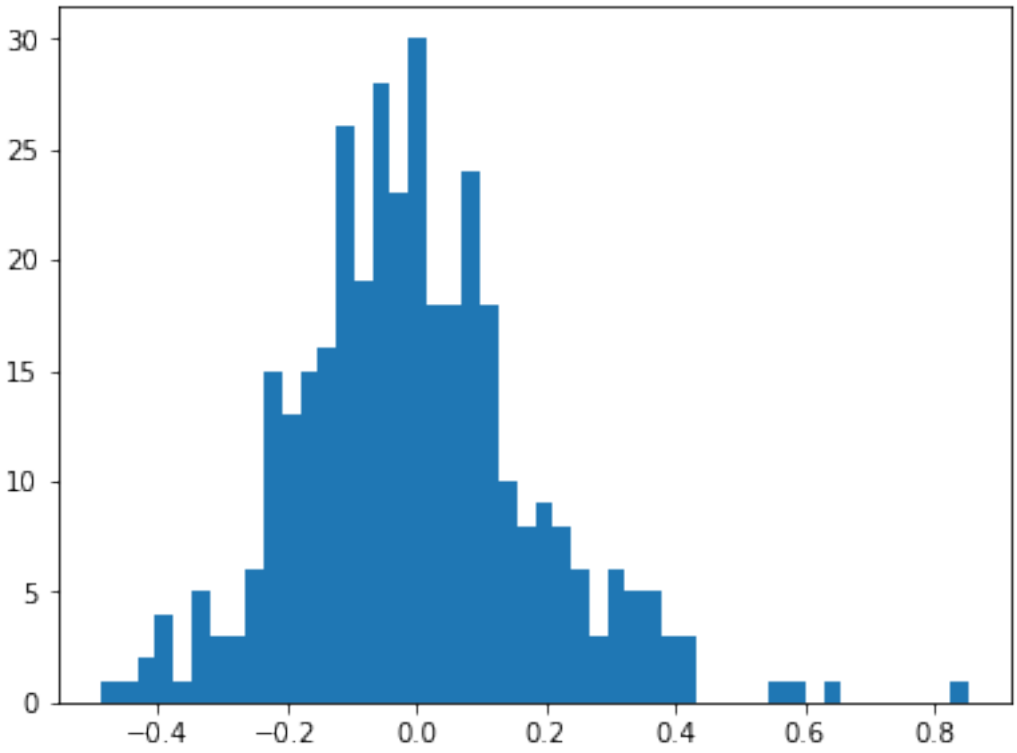
0.1 Introduction

Why do stocks change? What factors influence them? We can look at these market fluctuations and try to predict them. For this casestudy we will be using VIX and EMV which are volatility indexes and with these we will perfom analysis on stocks. In this casestudy we will also look at the other factors that can cause regime switches and market volatility. We will also show that it is possible to forecast the stock market index, however there is a lot of error and it could not be very accurate. The past data is useful and can help us construct a predictive model which can forcast the future stock markets, however with more data it allows our model to be more accurate and can lead to better forecasts. There are various factors that influence the stock markets, however we will show that it is possible to simplify the model and remove factors that are not as important, as a result this will make the model simplier and more practical.

0.2 Stock Market Regimes

From the MVIX dataset we can visualize the indexes on a histogram

[33]:
```
MVIXLog = np.log(MVIX)
count, edges, patches = plt.hist(MVIXLog, M)
```



We can use the Genetic Algorithm that we were given in class to find the best possible mixture for this index. We can see that this index follows a normal distribution and is used to measure for market volatility. So if we find the best possible mixture for this index, we can find three things from it. The first would be the different types of regimes that are present in the stock market, this would be the number of mixtures. The next thing we could find would be how often those regimes occur in our index and this would be the mixture proportions. The last thing we can find would be how those regimes impact the stock market. By using the Genetic Algorithm and setting its parameters based on our observed distribution, we can find optimal mixtures ranging from size 2 to size 5.

[64]:
```
for i in range(2,6):
    optimalData = opitmal.data
    fit = optimalData[i]['fit'][0]
    print('Objective Function Value for ' + str(i) + ' mixtures: '+ str(fit) +␣
↪'\n')
    print(optimalData[i])
    print('\n')
```

```
Objective Function Value for 2 mixtures: 58.09707963075827

       mean  variance  proportion       fit
0  0.002157  0.173463    0.847693  58.09708
1 -0.068098  0.092387    0.153125  58.09708
```

```
Objective Function Value for 3 mixtures: 84.33056638379526

       mean  variance  proportion        fit
0 -0.043668  0.193490    0.266326  84.330566
1  0.017289  0.153496    0.671230  84.330566
2 -0.192444  0.306769    0.061827  84.330566


Objective Function Value for 4 mixtures: 61.83169645799751

       mean  variance  proportion        fit
0  0.041243  0.082099    0.160858  61.831696
1  0.206839  0.151323    0.132169  61.831696
2 -0.062885  0.137320    0.702027  61.831696
3 -0.315756  0.162647    0.005362  61.831696


Objective Function Value for 5 mixtures: 93.26276812141131

       mean  variance  proportion        fit
0  0.209015  0.260236    0.074647  93.262768
1  0.063773  0.220146    0.013344  93.262768
2 -0.188550  0.112890    0.218365  93.262768
3  0.063093  0.087674    0.255275  93.262768
4 -0.001078  0.172855    0.435895  93.262768
```

From the output above we can see that having just having two regimes is enough to explain market volatility compared to the other number of regimes. We can see this by looking at the Objective Function Values that we found above we notice that for only two regimes it has the smallest Objective Function Value therefore it is better than the others. We can also do the Chi Squared test all the various numbers of regimes to confirm our result.

[73]:
```
for i in p_value:
    print('P-value for mixture '+ str(i) + ': ' + str(p_value[i]))
```

```
P-value for mixture 2: 0.09281696910973097
P-value for mixture 3: 0.014659773563586276
P-value for mixture 4: 0.012542887476913052
P-value for mixture 5: 0.03620320680154995
```

The Chi Squared test further backs up what we stated above that two regimes is enough to explain market volatility compared to the other number of regimes, as we can see that none of the other mixture's P-values are significant as they are below 0.05 which is the standard level of significance for the P-value test. Now that we know the number of regimes we can compute a probability transition matrix, where each element will be the probability by which the market could switch

from its current regime to another one, or else just remain in the same regime.

[83]:
```
mean = optimalData[2]['mean']
var = optimalData[2]['variance']
prop = optimalData[2]['proportion']
NormalDist1 = NormalDist(mean[0], var[0])
NormalDist2 = NormalDist(mean[1], var[1])
regime = []

for i in range(N): # Identifying what regime each MVIXLog belongs to
    if prop[0] * NormalDist1.pdf(MVIXLog[i]) > prop[1] * NormalDist2.
 ↪pdf(log_returns[i]):
        regime.append(0)
    else: regime.append(1)

PTMatrix = np.zeros((2, 2))
for i in range(1, len(regime)):
    if regime[i] == 0:
        if regime[i-1] == 0:
            PTMatrix[0][0] += 1
        else:
            PTMatrix[0][1] += 1
    elif regime[i-1] == 0:
        PTMatrix[1][0] += 1
    else: PTMatrix[1][1] += 1
PTMatrix = PTMatrix/PTMatrix.sum(axis=1)[:,None]

print(PTMatrix)
```

```
[[0.62869198 0.37130802]
 [0.76068376 0.23931624]]
```

From the probability transition matrix we can see that the stock market has a higher probability in staying in the first regime and if it is in the second regime it has a high probability of returning back to the first regime. I feel as there are many reasons why this regime-switching phenomenon might occur in stock market some include some private news coming out about a product/service or there might be a launch of a new project/industry which can also impact the stock markets.

0.3 Forces Driving the Market Volatility

Above we investigated the number of regimes that underlie the stock market's volatility and that the rare fluctuation that would occur could come from the market changing from one regime to another. We can take a look at a set of measures to see if they correlated to the fluctuations observed in the stock market. To do this we can perform various types of regression on the CBOE VIX index which can be considered an estimator of the equity market's implied volatility.

0.3.1 Model Specification

We will be using the python library scikit-learn to perform four different types of regression which would be OLS, LASSO, Ridge Regression and Elastic Net Regression. But before we perform these types of regression we need to split the data into train and test data.

[89]:
```
y = data["VIX"]
X = data.drop('VIX', axis=1)
Xtrain, Xtest, ytrain, ytest = train_test_split(X, y, test_size=0.33)
```

OLS Regression is a very common and old regression techinque, however it is very susceptible to multicollinearity and outliers have a greater influence which can result in getting incorrect models. LASSO Regression is when the model performs regularization and simplifies the model by using a sebset of variables, then sets the remaining variables to zero. Ridge Regression also performs regularization. Both Ridge Regression and LASSO Regression can lead to an increase of bias in the model which can again lead to an incorrect model. Elastic Net Regression mixes LASSO Regression and Ridge Regresson. We will perform regression using each of these methods and the one that performs the best on the best and is the simpliest is the one that we will use for our final model.

[119]:
```
print('OLS Regression\n')
ols = linear_model.LinearRegression()
# Train the model using the training sets
ols.fit(Xtrain, ytrain)
# Make predictions using the testing set
ypred = ols.predict(Xtest)
# The coefficients
print("Coefficients: \n", ols.coef_)
# The mean squared error
print("Mean squared error: %.2f" % mean_squared_error(ytest, ypred))
# The coefficient of determination: 1 is perfect prediction
print("Coefficient of determination: %.2f" % r2_score(ytest, ypred))

print("====================================")
print('LASSO Regression\n')

ridge=linear_model.Ridge(alpha=0.5)
# Train the model using the training sets
ridge.fit(Xtrain, ytrain)
# Make predictions using the testing set
ypred = ridge.predict(Xtest)
# The coefficients
print("Coefficients: \n", ridge.coef_)
# The mean squared error
print("Mean squared error: %.2f" % mean_squared_error(ytest, ypred))
# The coefficient of determination: 1 is perfect prediction
print("Coefficient of determination: %.2f" % r2_score(ytest, ypred))

print("====================================")
```

```
print('Ridge Regression\n')

Lasso=linear_model.Lasso(alpha=0.5)
# Train the model using the training sets
Lasso.fit(Xtrain, ytrain)
# Make predictions using the testing set
ypred = Lasso.predict(Xtest)
# The coefficients
print("Coefficients: \n", Lasso.coef_)
# The mean squared error
print("Mean squared error: %.2f" % mean_squared_error(ytest, ypred))
# The coefficient of determination: 1 is perfect prediction
print("Coefficient of determination: %.2f" % r2_score(ytest, ypred))

print("====================================")
print('Elastic Net Regression\n')

Ela=linear_model.ElasticNet(random_state=0)
# Train the model using the training sets
Ela.fit(Xtrain, ytrain)
# Make predictions using the testing set
ypred = Ela.predict(Xtest)
# The coefficients
print("Coefficients: \n", Ela.coef_)
# The mean squared error
print("Mean squared error: %.2f" % mean_squared_error(ytest, ypred))
# The coefficient of determination: 1 is perfect prediction
print("Coefficient of determination: %.2f" % r2_score(ytest, ypred))
```

```
OLS Regression

Coefficients:
 [ 0.03534465  0.66123904  0.11591751  0.76369272  0.43577377  0.16145019
 -1.17164781 -0.31039402  0.22281548 -0.38550491  0.60797052  0.76003348
  0.46988485  0.07075083 -0.05426345 -0.51197856 -3.20040473 -2.32386477
  1.10483581  3.74523875  2.34098204  1.46842226 -1.52452148 -1.41418029
 -0.61711646 -0.72956083 -1.35667669  2.1028198   1.16059775 -6.2065109
  1.27598362  4.77370009  5.73126408  0.67175055  1.35380004 -2.81924692
 -0.36423292  0.47157413 -0.83396466  2.47983114  6.11628167 -1.45331462
  0.0398353  -7.9981978  -0.08881026]
Mean squared error: 28.41
Coefficient of determination: 0.53
====================================
LASSO Regression

Coefficients:
 [ 1.09368631e-01  6.00701155e-01  1.04907006e-01  6.96212567e-01
```

```
  4.71953748e-01  1.42890935e-01 -1.18122124e+00 -2.53389238e-01
  2.03443282e-01 -3.75770716e-01  3.20527881e-01  4.98181650e-01
  4.15763644e-01  8.21733036e-04 -8.72500513e-02 -5.30182092e-01
 -2.67459006e+00 -2.18554268e+00  1.06482069e+00  3.65677449e+00
  2.23760706e+00  1.35592640e+00 -1.40392680e+00 -1.32195770e+00
 -5.85357412e-01 -6.69735391e-01 -1.29615396e+00  2.06430555e+00
  1.17858893e+00 -2.55382480e+00  1.21023601e+00  3.69839510e+00
  5.09580171e+00  6.10518851e-01  1.33325991e+00 -2.49409761e+00
 -3.40110395e-01  4.24809549e-01 -7.26405399e-01  1.95350580e+00
  5.15204400e+00 -1.11189674e+00  4.95638722e-02 -5.07611483e+00
 -5.14830185e-02]
Mean squared error: 27.38
Coefficient of determination: 0.55
====================================
Ridge Regression

Coefficients:
 [ 0.82347251  0.         -0.06975808  0.11191545  0.23583328  0.
 -1.06159702  0.         -0.          0.         -0.         -0.
  0.          0.         -0.19888124 -0.         -0.         -0.
  0.          0.          0.         -0.         -0.         -0.
 -0.         -0.36854367 -0.          0.32167759  0.          0.
  0.         -0.          0.         -0.          0.          0.
 -0.          0.         -0.03557198 -0.          0.          0.
 -0.         -0.          0.        ]
Mean squared error: 26.58
Coefficient of determination: 0.56
====================================
Elastic Net Regression

Coefficients:
 [ 0.74802658  0.         -0.05237398  0.14244318  0.18679562 -0.
 -0.84974367  0.         -0.          0.         -0.         -0.
  0.          0.         -0.10936709 -0.         -0.         -0.
  0.          0.          0.          0.         -0.         -0.
 -0.         -0.38827873  0.          0.34146341  0.          0.
 -0.         -0.          0.         -0.          0.          0.
 -0.          0.         -0.06847581 -0.          0.          0.
 -0.         -0.          0.        ]
Mean squared error: 26.32
Coefficient of determination: 0.56
```

From looking at the output above we can see that Ridge Regression and Elastic Net Regression both have the highest Coefficient of determination, however Elastic Net Regression has a smaller MSE so it should be the better model for our data. We can also see that Ridge Regression and Elastic Net Regression are considerably sparser than OLS and LASSO regression, which means they have a simplier model as they have less variables.

0.3.2 Fitting and Diagnostics

As we mentioned above in Model Specification, by looking at the output the final model should be Elastic Net Regression as it has the highest Coefficient of determination and the lowest MSE. Addionally it also has a very sparse model making it simple as well. Now we can look at the coeffiencts to see which have the most impact on the volatility of the stock market. Again this is not the best model there is as there is still a very high MSE and the Coefficient of determination is equal to 0.56 which is not the highest it can be, but it is better than the other three and that is why we picked it. If we fine tuned some parameters this model can be better as well, this would be tuning the random_state parameter or tuning the test_size parameter when we split the data.

[132]:
```
ElasticNetCoeff = Ela.coef_
#data = data.drop('VIX', axis=1)
temp = 0
model = {}
for i in data:
    if (temp < 45 and ElasticNetCoeff[temp] != 0):
        model[i] = ElasticNetCoeff[temp]
    temp += 1
model
```

[132]:
```
{'EMV': 0.7480265793327023,
 'Infectious Disease EMV Tracker': -0.05237398110097391,
 'Macroeconomic News and Outlook EMV Tracker': 0.1424431762546036,
 'Macro - Broad Quantity Indicators EMV Tracker': 0.18679562019292745,
 'Macro - Interest Rates EMV Tracker': -0.8497436692650717,
 'Financial Crises EMV Tracker': -0.10936709071022656,
 'Monetary Policy EMV Tracker': -0.38827872505436895,
 'Financial Regulation EMV Tracker': 0.3414634149248973,
 'Trade Policy EMV Tracker': -0.06847581038242066}
```

0.4 Discussion/Conclusion

By looking at the model that we have generated from Elastic Net Regression we can see that the biggest influencers of VIX are EMV, Interest Rates EMV Tracker, Monetary Policy EMV Tracker, and Financial Regulation. So by looking at these variables we notice they are all economic factors, therefore we can infer that economic factors impact the volatility of the stock market the most. Like we said in Fitting and Diagnostics our model is not perfect and can get tuned further to recieve better results. One way to fix this would be to invest more time and carefully test and tune the parameters. Also the data that we recieved is not perfect for what we are trying to predict, as we are trying to predict the number of sudden fluctuations but the data only has a small portion of these sudden fluctuations. Having so few of these can lead to the model being skewed which can give us biased results. One way to conteract this would be to either have more data that includes the sudden fluctuations that we are trying to predict or we can extraploate the data and try to test with the addional data that we have created, however that can also lead to a worse result.

0.5 Bibliography

1. Kuepper, J. (2021, December 7). Cboe volatility index (VIX). Investopedia. Retrieved December 22, 2021, from https://www.investopedia.com/terms/v/vix.asp
2. developers, scikit-learn. (2007). Sklearn.linear_model.linearregression. scikit. Retrieved December 22, 2021, from https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html
3. Hunter, J. (2012). Matplotlib.pyplot.hist¶. matplotlib.pyplot.hist - Matplotlib 3.5.1 documentation. Retrieved December 22, 2021, from https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.hist.html