Financial Market Performance Forecasting

INFO 523 - Fall 2025 Final Project Write-up
Team 5 Members - Nicholas Tyler, John Moran, Zuleima Cota
University of Arizona, College of Information Science

Introduction/Motivation

Analyzing financial market data can provide insights that help inform sound investment strategies and mitigate risk for businesses and individual investors. With that in mind, this project assessed historical stock market performance to better understand how long-term trends and macroeconomic factors like unemployment rates relate to market behavior. By combining market-based and macroeconomic indicators, the goal of this analysis was to explore long-term volatility patterns and conduct stock market price forecasting. The two main research questions that were followed to shape the scope of this project included:

1. Can we predict the Volatility Index within the financial markets based on various economic factors (i.e. unemployment rates, U.S. Treasury 3-Month Bond Yield, etc.)?
2. What insights can historical stock market data offer to forecast future price movements and assess market volatility?

Together, these questions offered a framework for evaluating what data mining models can be applied to identify underlying patterns and forecasting. By using various regression models to assess question 1 and time series analyses for question 2, the results help clarify not only how well these models perform, but also what they reveal about the long-term dynamics of the stock market.

Data Overview

The Daily Stocks Data dataset (stocks_data.csv) is a dataset from Kaggle which captures data on financial market performance from 1990 to 2024. This data has relevant information related to volume, macroeconomic indicators, volatility index, and uncertainty metrics. This dataset is compiled from a variety of historical records, including the Chicago Board Option Exchange, Yahoo Finance and historical finance datasets, Bureau of Economic Analysis, Federal Reserve, Economic Policy Uncertainty Index, and the Global Policy Uncertainty Database.

While the Daily Stocks Data dataset had insightful features for our analysis, this dataset lacked a clear understanding of the unemployment rate, as this information was only presented in quartiles. Therefore, we obtained a dataset from the Bureau of Labor Statistics (SeriesReport.csv), which provides the unemployment rate per month for the United States from 1990 - 2024. By joining this data with our preexisting dataset, we obtained a dataset with a complete picture of the financial market along with the current economic conditions per day from 1990-2024.

Exploratory Data Analysis

Our dataset has 1 date feature (dt) and 14 total numeric features. Multiple numeric features were positively skewed, including VIX, S&P 500 Index (and its trading volume), DJIA (and its trading volume), the Economic Policy Uncertainty Index, Geopolitical Risk Index, the Previous Day Volatility Index, and Unemployment Percent. Furthermore, there were a few features that exhibited slight multinomial distributions, including the S&P 500 Trading Volume, the Hang Seng Index, U.S. Treasure 3-Month Bond Yield, and Unemployment (in Quartiles).

Methods, Analysis and Results

To conduct our analysis, a total of seven models were trained. For question 1 we used Ridge, Random Forest Regressor, Lasso, and Gradient Boost Regression. For question 2 we used LTSM, Prophet, and ARIMA.

Ridge

The batch_ridge() function was created in utils/model_training, in order to train and get predictions from a batch of Ridge models. We used sklearn.linear_model's Ridge module for this model implementation. We created 3 models for our Ridge examination, one used unemployment numbers, the second used bond yield rates, and our third used both unemployment and bond yield rate as features. All 3 models used vix (volatility index) as the target, and aimed to determine how well these features can predict the volatility index of financial markets. Our first model ended training with a coefficient of ~2.158. This coefficient intuitively makes sense in this context, as unemployment goes up, we expect market volatility to follow. Although this result is intuitive, the results from this model, and the other two, suggest none of the models were able to accurately predict vix with these features. Of the three models, the third produced the best mean squared error (MSE) score of ~53.178, which is quite high given the normal range of VIX ~(12-25). Similarly, the first model had the highest R-Squared score of ~0.097, meaning only 9.7% of the target's variance could be explained by the model. These results suggest that unemployment and bond yield rates are not strong indicators of VIX, or Ridge lacks the complexity to accurately learn the underlying patterns here.

Random Forest Regressor

The random_forest() function was created in utils/model_training, in order to train and get predictions for sklearns.ensemble's RandomForestRegressor. Within this function, RandomizedSearchCV was used to tune the model's hyperparameters. This allowed the model to explore a wide range of hyperparameter permutations, and find the combination that best generalizes the unseen data. The Random Forest model was trained with the following features: sp500, sp500_volume, djia, hsi, ads, us3m, joblessness, epu, GPRD, and Unemployment Percent, with VIX as the target. After training, the model was able to achieve a MSE of ~2.096 and a R-Square score of ~0.965. These results indicate that the Random Forest Regressor captured the nonlinear relationships among the features and explained a substantial portion of VIX's variance.

Lasso

We also trained a lasso regression model due to its built-in techniques to avoid the overfitting of models. The functions for this model (lasso_regression and lasso_hyperparameter_tuning) can be found in utils/lasso_regression.py. For this model, we incorporated all features (excluding date) of our dataset, set Volatility Index (VIX) as our response variable, and split the data into 80% training data and 20% testing data. Following training the model with our training dataset, we assessed performance on this model by utilizing mean square error and R-squared scores. We initially received a very poor R-squared score of -0.00017 and an MSE score of 0.1081. However, following these results, we performed hyperparameter tuning on the alpha value to ensure that we had the best value selected. Therefore, after performing lasso cross-validation, we determined that an alpha value of 0.0001 would produce the best model, which gave us a much better R-squared value of 0.5605 and MSE score of 0.0475. While these values were improved, this model still does not effectively predict volatility in the financial markets, so we cannot effectively utilize this model with certainty to predict volatility within the financial markets.

Gradient Boost Regression

Following our lasso regression, we trained a gradient boost regression model due to its high performance and proven results with financial market predictions. The functions for this model (gradient_boost and gradient_boost_hyperparameter_tuning) can be found in utils/gradient_boost_regression.py. For our gradient boost regression model, we trained two models one with Principal Component Analysis (PCA) applied and the other without PCA applied. For both models, we also incorporated hyperparameter tuning to ensure that we had the best model selected. As with our lasso regression model, we used all features (excluding date) into our model, used Volatility Index (VIX) as our response variable, and split our dataset into 80% for training data and 20% for testing data. On our non-PCA model, we determined through grid search cross-validation that our best parameters for our non-PCA model were a learning rate of 0.1, a max depth of 7, and 300 estimators. Our PCA model had the same parameters except for having 100 estimators rather than 300. Following hyperparameter tuning, we discovered that the model without PCA applied outperformed the PCA model. Our non-PCA model returned an R-squared value of 0.9626 and a MSE of 0.004, whereas our PCA model returned an R-squared value of 0.6844 and a mean squared error of 0.0341. Therefore, while both gradient boost regression models performed very well, our model utilizing our normal features rather than PCA features displayed a very high predictive power on our model.

LSTM

To assess future market volatility, we trained a Long Short-Term Memory (LSTM) model. A LSTM model is a type of Recurrent Neural Network (RNN) used to solve sequential learning tasks. The function for this model (lstm_function) can be found in utils/LSTM.py. For this model, we included all predictor variables, set Volatility Index (VIX) as the response variable, and split the dataset into 80% training and 20% testing. We utilized mean squared error and r-squared for our performance scores. This model performed very poorly, receiving an R-squared value of -9.7147 and MSE of 1.2614. Therefore, we decided to disregard this model any further in our analysis.

Prophet

To predict future market volatility, we trained a prophet model. A prophet model is a model developed by Meta used for forecasting time series data, which made it perfect for our analysis. The function for this model (prophet_model) can be found in utils/prophet.py. For this model, we only used the date column and the predictor variable (Volatility Index). The overall model was evaluated by returning the MSE and R-squared values, which were 0.0478 and 0.5618, respectively.

Two plots were also created from this model, one returning all historical and forecasted results and one returning results from the latest historical year and first forecasted year. Total forecast results show an overall slight increase in VIX over time, following the ascent initiated around 2015. The confidence interval increases drastically as the forecasted dates get farther from the historical results, which is to be expected. The first forecasted year (as seen on the zoomed-in plot) displays an overall shape that is like the prior year's results with a relatively small confidence interval (less than 1%). Therefore, although this model appears to do a decent job forecasting Volatility within the financial markets, we must be cautious when making predictions using this model due to its average R-squared value.

ARIMA

Another model used in the time-series analysis included the Autoregressive Integrated Moving Average (ARIMA) model. This model was selected because it can predict future outcomes based on an analysis of past values in a time series. ARIMA stands for autoregressive (AR) with parameter p, moving average (MA) with parameter q to learn patterns from previous values, and it uses differencing (I) with parameter d to handle non-stationary data. The parts to consider include starting with decomposition of the data to get a sense of the patterns and running an Augmented Dickey-Fuller (ADF) to test for stationarity which is evaluated based on the p-value. The p-value returned at a value below 0.05 which indicates that the series is stationary showing that the mean and variance stay constant over time. Due to this, the d parameter can be set to 0. To evaluate ARIMA we used MAE and RMSE scores receiving an MAE value of 5.666 and RMSE of 7.952.

Two methods to complete this section were used including ARIMA and Auto-ARIMA, a more automatic version using the pmdarina package to find the best hyperparameters. The findings indicate that while both are suitable for assessing long-term linear behaviour, there is a limitation when uncertainties occur such as the instability of the markets in 2020 during the COVID pandemic. The predicted values in our charts remained linear throughout all assessed years despite the spikes seen in 2020 which is inaccurate. These forecasts provide adequate estimates when relationships are linear over time but do not respond well to volatility surges.

Conclusion

Overall, Gradient Boost Regression outperformed the other models in question 1 (R-squared = 0.9626, MSE = 0.004). Random Forest also did well with a close R-squared value (R-square = 0.965, MSE of 2.096). This reveals that Gradient Boost Regression can handle non-linear relationships and when considering volatility, this is a suitable model that can capture a better picture of market changes.

For Question 2, Prophet achieved the best forecasting performance (MSE = 0.0478, R-squared = 0.5618), whereas the LSTM model performed the worst (R-squared = −9.7147, MSE = 1.2614). This indicates that the Prophet model can handle changes in trends within a given time period while also considering aspects like seasonality (yearly, weekly, daily data). These findings suggest that, although financial markets are inherently unpredictable, pairing regression and time-series models offers the most comprehensive approach for interpreting market behavior.

References:

"ARIMA." Darts Documentation. https://unit8co.github.io/darts/generated_api/darts.models.forecasting.arima.html#darts.models.forecasting.arima.ARIMA.

"Ex-09: Time Series Analysis." *Data Mine Arizona*, University of Arizona INFO 523 Course Website. https://datamineaz.org/exercises/ex-09.html#step-1-setup-and-data-preprocessing.

Lewinson, Eryk. Python for Finance Cookbook - Second Edition : Over 80 Powerful Recipes for Effective Financial Data Analysis, Packt Publishing, Limited, 2022. pp. 176-190. ProQuest Ebook Central, https://ebookcentral.proquest.com/lib/uaz/detail.action?docID=30301147.

TechwithJulles. "Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) — Creating an LSTM Model in Python Using TensorFlow and Keras." *Medium*, 13 Oct. 2024, https://medium.com/@techwithjulles/recurrent-neural-networks-rnns-and-long-short-term-memory-lstm-creating-an-lstm-model-in-13c88b7736e2.

Zevallos-Aquije, A., et al. "Prediction of Lead Concentration in the Rímac River
Using ARIMA, SARIMA and Python-Based Time Series
Analysis" *E3S Web of Conferences*, 2025, https://doi.org/10.1051/e3sconf/202564301001.