DAYANANDA SAGAR UNIVERSITY

KUDLU GATE, BANGALORE – 560068

Bachelor of Technology

in

COMPUTER SCIENCE AND ENGINEERING

Major Project Phase-II

Report

"VOLATILITY PREDICTION"

Submitted By:

LYSETTI LAKSHMI POOJITHA – ENG18CS0150

MOHAMMED KASHIF-ENG18CS0169

MUSKAAN GOEL- ENG18CS0177

MUSKAAN SINHA- ENG18CS0178

SAYEEDA UMAIMA SADAFUNNISA-ENG18CS0251

Under the supervision of

Prof. Geetha

Professor, Department of Computer Science and Engineering

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING,
SCHOOL OF ENGINEERING
DAYANANDA SAGAR UNIVERSITY,

(2021-2022)

DAYANANDA SAGAR UNIVERSITY

School of Engineering, Kudlu Gate, Bangalore-560068

CERTIFICATE

This is to certify that the Phase-II project work titled **"VOLATILITY PREDICTION"** is carried out by **MOHAMMAD KASHIF (ENG18CS0169), LYSETTI POOJITHA LAKSHMI (ENG18CS0150), MUSKAAN GOEL (ENG18CS0177), MUSKAAN SINHA (ENG18CS0178), SAYEEDA UMAIMA SADAFUNNISA (ENG18CS0251),** bonafide students of Bachelor of Technology in Computer Science and Engineering at the School of Engineering, Dayananda Sagar University, Bangalore in partial fulfillment for the award of degree in Bachelor of Technology in Computer Science and Engineering, during the year 2021-2022.

Prof Geetha	**Dr Girisha G S**	**Dr. A Srinivas**
Professor	Chairman CSE	Dean
Dept. of CSE,	School of Engineering	School of Engineering
School of Engineering	Dayananda Sagar University	Dayananda Sagar University
Dayananda Sagar University		
Date:	Date:	Date:

Name of Examiner **Signature Of Examiner**

1.

2.

DECLARATION

We, **MOHAMMAD KASHIF (ENG18CS0169), LYSETTI POOJITHA LAKSHMI (ENG18CS0150), MUSKAAN GOEL (ENG18CS0177), MUSKAAN SINHA (ENG18CS0178), SAYEEDA UMAIMA SADAFUNNISA (ENG18CS0251)** are students of seventh semester B.Tech in Computer Science and Engineering, at School of Engineering, **Dayananda Sagar University**, hereby declare that the phase-II project titled **"Volatility Prediction"** has beencarried out by us and submitted in partial fulfilment for the award of degree in **Bachelor of Technology in Computer Science and Engineering** during the academic year **2021-2022.**

MOHAMMAD KASHIF (ENG18CS0169)

LYSETTI POOJITHA LAKSHMI (ENG18CS0150)

MUSKAAN GOEL (ENG18CS0177)

MUSKAAN SINHA (ENG18CS0178)

SAYEEDA UMAIMA SADAFUNNISA (ENG18CS0251)

ACKNOWLEDGEMENT

It is a great pleasure for us to acknowledge the assistance and support of many individuals who have been responsible for the successful completion of this project work.

First, we take this opportunity to express our sincere gratitude to School of Engineering & Technology, Dayananda Sagar University for providing us with a great opportunity to pursue our Bachelor's degree in this institution.

We would like to thank ***Dr. A Srinivas. Dean, School of Engineering&Technology, Dayananda Sagar University*** *for his constant encouragement and expert advice.*

It is a matter of immense pleasure to express our sincere thanks to ***Dr.Girisha G S, Department Chairman, Computer Science and Engineering, Dayananda Sagar University****, for providing right academic guidance that made our task possible.*

We would like to thank our guide ***Prof.Geetha Assistant Professor, Dept. of Computer Science and Engineering, Dayananda Sagar University****, for sparing her valuable time to extend help in every step of our project work, which paved the way for smooth progress and fruitful culmination of the project.*

We would like to thank our Project Coordinators ***Dr. Meenakshi Malhotra ,Dr.Bharanidharan*** *and all thestaff members of Computer Science and Engineering for their support.*

We are also grateful to our family and friends who provided us with every requirement throughout the course.

We would like to thank one and all who directly or indirectly helped us in the Project work.

TABLE OF CONTENTS

TABLE OF FIGURES

LIST OF ABBREVIATIONS

- **LSTM:**LONG SHORT TERM MEMORY
- **ARCH:**AUTO REGRESSIVE HETEROSKEDASTICITY
- **GARCH:**GENERALIZED AUTO REGRESSIVE HETEROSKEDASTICITY
- **GB :**GIGABYTES
- **EDA :** EXPLORATORY DATA ANALYSIS
- **BTC-USD:**BITCOIN US DOLLAR

ABSTRACT

In finance, volatility refers to the variation degree of asset price, and it measures the uncertainty of the price. It plays an important role in both academic research and the financial industry. In risk management and performance measurement, volatility is a risk indicator itself and can be a part of some other indicators, like the Sharpe ratio. In portfolio theory, Markowitz used volatility to measure the risks of assets and the overall risk of the portfolio. Volatility is both the input and the optimization target of the portfolio construction model. In derivative pricing, prices of derivatives can be determined by the volatility of the underlying assets . Since volatility can be useful in different financial areas, and in some situations, the information about future volatility is needed to make financial decisions, it can be valuable to forecast volatility. Econometric methods are widely used to forecast the volatility of financial assets. Engle introduced the autoregressive conditional heteroscedasticity (ARCH) model to describe volatility. In this model, the conditional variance is given as a function of the previous variances. The volatility dynamics can be gained by maximizing the likelihood of the model, and then the estimated model can be used to forecast future volatility. Bollerslev extended the ARCH model to be a generalized autoregressive conditional heteroscedasticity (GARCH) model. This model makes conditional variance to be a function of previous errors and previous variances.

CHAPTER 1

INTRODUCTION

The unprecedented volatility in financial markets during the last decade has led to increased awareness about the importance of explaining key drivers behind such movements. On the one hand, the structure of financial markets is complex, highly nonlinear and has a low signal-to-noise ratio, which makes it nearly impossible to predict short-term asset returns On the other hand, various stylized facts (e.g., the slow decay of autocorrelation in absolute returns), implies that volatility is to a large extent predictable. With its crucial role in asset pricing, portfolio allocation, and risk management, volatility forecasting has therefore attracted a large amount of attention. The origins of this literature go back to Engle (1982); who developed discrete-time GARCH and stochastic volatility processes for modeling autoregressive conditional heteroskedasticity (see Hansen and Lunde, 2005, for a broader review). As argued by Corsi (2009), however, standard GARCH and stochastic volatility models are unable to reproduce all of the salient features of financial markets. Adding even more complexity to the problem of portfolio management is the dynamic of stock volatility—that is, the degree of variability of a stock's price over some finite period. Predicting volatility is important for many reasons. An accurate picture of a portfolio's volatility is necessary for an investment manager to manage portfolio risk and meet investment criteria according to the fund's investment mandate. It is likewise important for options traders and financial product structurers, whose financial instruments are priced in part on expectations of volatility. There are many traders whose entire strategy relies on their ability to predict a more accurate picture of volatility than the rest of the market, thereby allowing them to exploit a form of arbitrage. As such, in this study, we aim to extend existing work on the use of neural networks to predict stock price volatility. The goal of this work is to advance the understanding of novel methods of volatility prediction so as to positively impact the stability of the overall financial markets—with participants better informed about

their portfolio risk, those participants can do a better job of managing risk and avoid contributing to future stock market meltdowns

1.1. PURPOSE AND PROCESS

In finance, volatility refers to the variation degree of asset price, and it measures the uncertainty of the price. It plays an important role in both academic research and the financial industry. In risk management and performance measurement, volatility is a risk indicator itself and can be a part of some other indicators, like the Sharpe ratio. In portfolio theory, Markowitz used volatility to measure the risks of assets and the overall risk of the portfolio. Volatility is both the input and the optimization target of the portfolio construction model. In derivative pricing, prices of derivatives can be determined by the volatility of the underlying assets..

1.2 SCOPE

Our Project will use Recurrent Neural Networks LSTM , and Traditional Methods like GARCH and ARCH .We will be comparing the results from Both Neural Networks and Traditional methods and give the best result of those both. The scope of this project would be to build a model using Deep learning and to give desired results.

1.3 INTENDED AUDIENCE

This project is intended to benefit people in finance sector and business professionals in share market.

CHAPTER 2

PROBLEM DEFINITION

Volatility is the range of price change a security experiences over a given period of time. If the price stays relatively stable, the security has low volatility. A highly volatile security hits new highs and lows quickly, moves erratically, and has rapid increases and dramatic falls, Because people tend to experience the pain of loss more acutely than the joy of gain, a volatile stock that moves up as often as it does down may still seem like an unnecessarily risky proposition. Since Volatility is important for Finance industry our project would aim to forecast the behavior the Stock Market Volatility of Bitcoin Data for next set of days. Volatility forecasting has important implications for all investors focused on risk-adjusted returns, especially those that employ asset allocation, risk-parity, and volatility-targeting strategies. An understanding of the different approaches used to forecast volatility and the implications of their assumptions and dependencies provides a robust framework for the process of risk budgeting.

CHAPTER 3

LITERATURE REVIEW

Sl.no	Paper Title	Year of Publish	Technology	Inference
1.	Prediction Approach for Stock Market Volatility Based on Time Series Data **Ieee journal (vol 7)**	25 January 2019	ARIMA Model	Introduces the concept of time series analysis and forecasting in the perspective of Indian economy .The predicted time series has been compared with the actual time series, which shows roughly a deviation of 5% mean percentage error for both Nifty and Sensex on average
2.	Machine Learning for Multi-step Ahead Forecasting of Volatility Proxies **Midas conference**	September 2017	Machine Learning	By using a Nonlinear Autoregressive Exogenous Model (NARX) model combining two proxies to predict one of them, shows that it is possible to improve the prediction of the future value of some proxies by using the

				information provided by the others.
3.	Volatility Forecasting Techniques using Neural Networks **International journal of engineering research & technology (ijert)**	June-2021	Deep Learning	It is observed that all neural network models perform much better than traditional models. But since most of these models depend only on historical data, more research is needed in considering market sentiment as a variable as it plays an important role in market fluctuations.
4.	Machine Learning for Realised Volatility Forecasting **SSRN Electronic Journal**	November 2021	Deep Learning	This paper compares machine learning and HAR models for forecasting realised volatility of 23 NASDAQ stocks using 146 variables extracted from limit order book (LOBSTER) and stock-specific news (Dow Jones Newswires) from 27 July 2007 to 18 November 2016. We

				find simpler ML to outperform HARs on normal volatility days.
5.	Forecasting Volatility in Indian Stock Market using Artificial Neural Network with Multiple Inputs and Outputs **International journal of computer applications**	June 2015	Deep Learning	The objective is to capture the effects of both external and internal shocks on spot market volatility. The advantage of using the ANN framework is that it does not presume any linearity in the relationship between the inputs and outputs. Further, it allows for interaction and feedback between the inputs.

CHAPTER 4

PROJECT DESCRIPTION

4.1 PROPOSED DESIGN

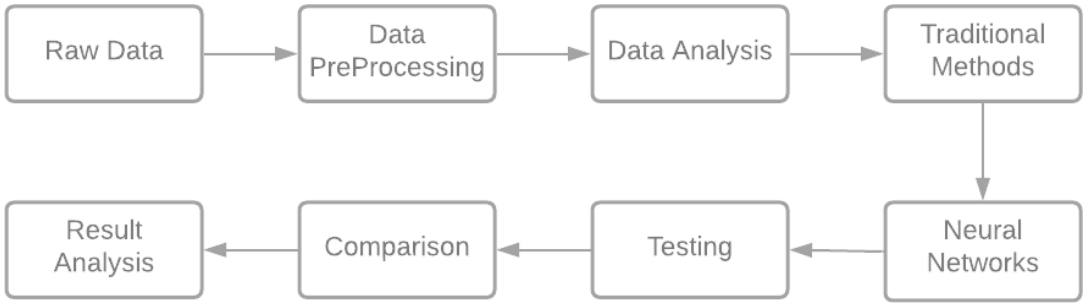


Fig 4.1.1 Flow chart

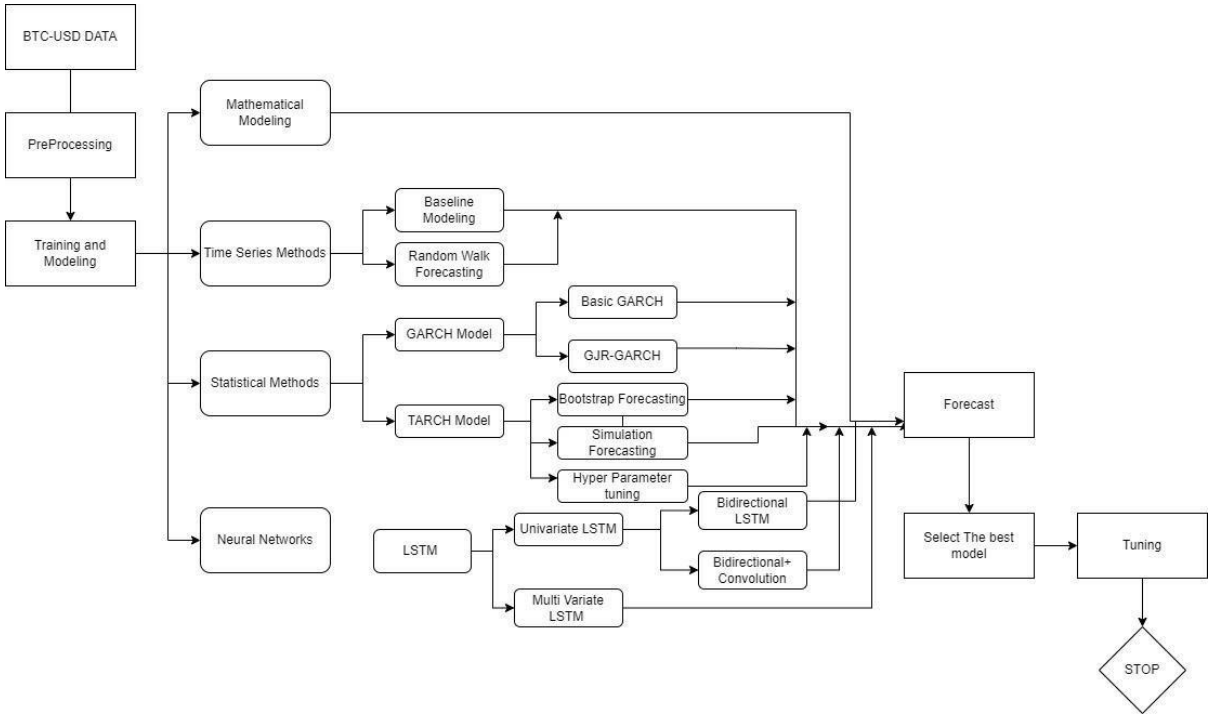


Fig 4.1.2 Detailed Design

1)Mathematical Model : In this model we do performance metrics using RMPSE(Root Mean Squared percentage error) and RMSE(Root Mean Squared Error) with RMSPE prioritised. Usually with financial time series, if we just shift through the historic data trying different methods, parameters and timescales, it's almost certain to find some strategy with in-sample profitability at some point. However the whole purpose of "forecasting" is to predict the future based on currently available information, and a model that performs best on training data might not be the best when it comes to out-of-sample generalisation (or overfitting). Avoiding/Minimising overfitting is even more important in the constantly evolving financial markets where the stake is high.

2)Time Series Model: Time series analysis includes many categories or variations of data, analysts sometimes must make complex models. However, analysts can't account for all variances, and they can't generalise a specific model to every sample. Models that are too complex or that try to do too many things can lead to a lack of fit. Lack of fit or overfitting models lead to those models not distinguishing between random error and true relationships, Baseline Models: A baseline model is essentially a simple model that acts as a reference in a machine learning project. Its main function is to contextualise the results of trained models. Baseline models usually lack complexity and may have little predictive power. Regardless, their inclusion is a necessity for many reasons.

a) Mean Baseline Model- One of the essential characteristics of Volatility is its mean-revert over the long term. Therefore my first baseline model would be a very simple one that only outputs the average current realised volatility of the whole training set as it predicts everything. We calculate the mean of the scaled training data. We then create a series of predictions for the baseline model based on the validation data set.

b) Random Walk Naive Forecasting- A commonly known fact about volatility is that it tends to be autocorrelated, and clusters in the short-term. This property can be used to implement a naive model that just "predicts" future volatility by using whatever the daily volatility was at the immediate previous time step. We use previous n_future days volatility and plot predictions vs target values on validation set. We then append the metrics output to dataframe.

3)Statistical Model: A statistical model is a mathematical model that embodies a set of statistical assumptions concerning the generation of sample data. A statistical model represents, often in considerably idealised form, the data-generating process.

c) Garch Model- It stands for "Generalised Autoregressive Conditional Heteroskedasticity". It is an extension of the ARCH model. It includes lag variance terms with lag residual errors from the mean process calculated in the above models. Mathematically , GARCH can be represented as:

i) Basic Garch: We visualize autocorrelation of squared returns and we also visualize partial autocorrelation of squared returns. The plot seems to indicate that there is only significant correlations upto the 7th lags, the ones other than that doesn't seem significant.

ii) GARCH model with Asymmetric Shock Response- The basic garch model assumes that positive or negative news have similar impact on volatility. In reality the market tends to change in an asymmetric way and negative news impacts the market more than the positive ones. GJR-Garch accounts for all the asymmetry of shock responses.

d) Tarch Model- It is another member in the GARCH family called TARCH(threshold autoregressive conditional heteroskedasticity). TARCH models the volatility using absolute

we first set the seed for reproducibility. Later we get volatility scaler and scaled conditional volatility from the model result. There are three parts of TARCH Modeling:

i) Bootstrap-based Forecasting for TARCH

ii)Simulation Based Forecasting for TARCH

iii)Hyperparameter Tuning for TARCH

2) Neural Networks: GARCH model is the gold standard for volatility prediction without traditional financial institutions, there has been an increasing number of professionals and researchers turning to Machine Learning, especially Neural Networks, to gain insights into the market in recent years. a) LSTM- It is translated for long short term memory and its an artificial neural network used in fields of artificial intelligence and deep learning. LSTM networks are also suitable for classification, processing and making predictions based on time series data.

i) univariate LSTM: A part of neural networks which can predict "the future" is RNN. This works best for time series data, like stock prices. - Bidirectional LSTM

ii) multivariate LSTM: The first multivariate model would be relatively simple within 2 layers of bidirectional LSTM. However, having more features would mean the model would be prone to overfitting. We set the seed for reproducibility after which a multivariate bidirectional LSTM neural network is constructed. The further process is followed by layering, fitting, visualising , forecasting and plotting predictions vs target values on the

validation set.

iii) We also try to forecast the predictability using 3 layers and 4 layers multivariate LSTM.

4.2ASSUMPTIONS AND DEPENDENCIES

- The data available online is not tampered.
- It would be used on OS that have enough performance

CHAPTER 5
REQUIREMENTS

5.1 FUNCTIONAL REQUIREMENTS

- Web Scrapping/Data Mining:Collecting the data from Yahoo Finanace
- Perform EDA
- Train over Traditional methods
- Training over Recurrent Neural Networks
- Comparing the Accuracies
- Getting the insights of the results

5.2 NON-FUNCTIONAL REQUIREMENTS

- Availability- The probability that system will be in an operable and committable state at the start of a mission when the mission is called for at arandom time.

- Correctness - Accuracy during Prediction is very important. The system shouldprovide accurate results based on the data

- Maintainability- During Prediction some errors might occur but the systemshould be capable of being retained, restored to a serviceable operation.

- Usability-The system should be used effectively to achieve objectives witheffectiveness and efficiency for proper use.

5.3 SOFTWARE REQUIREMENTS

- Jupyter notebook / Google Colab
- Python3.x.x
- NumPy
- Scikit Learn
- TensorFlow
- Keras
- Matplotlib
- Seaborn
- And other Python3 PyPi modules
- Yahoo ticker

5.4 HARDWARE REQUIREMENTS

- 4 GB RAM and above recommended.
- 10 GB internal storage and above recommended.
- Intel core i3 processor and above.

CHAPTER 6
METHODOLOGY

6.1 Analysis

The closing price is the raw price or cash value of the last transacted price in a security before the market officially closes for normal trading. It is often the reference point used by investors to compare a stock's performance since the previous day—and closing prices are frequently used to construct line graphs depicting historical price changes over time. The adjusted closing price factors in anything that might affect the stock price after the market closes, such as dividends or splits. Most stocks and other financial instruments are traded after-hours, although in far smaller volumes. Therefore, the closing price of any security is often different from its after-hours trading price. Closing prices are useful markers for investors to use to assess changes in stock prices over time. Even in the era of 24-hour trading, there is a closing price for any stock or other security, and it is the final price at which it trades during regular market hours on any given day. The closing price is considered the most accurate valuation of a stock or other security until trading resumes on the next trading day. The closing price on one day can be compared to the closing price on the previous day, 30 days earlier or a year earlier, to measure the changes in market sentiment toward that stock.

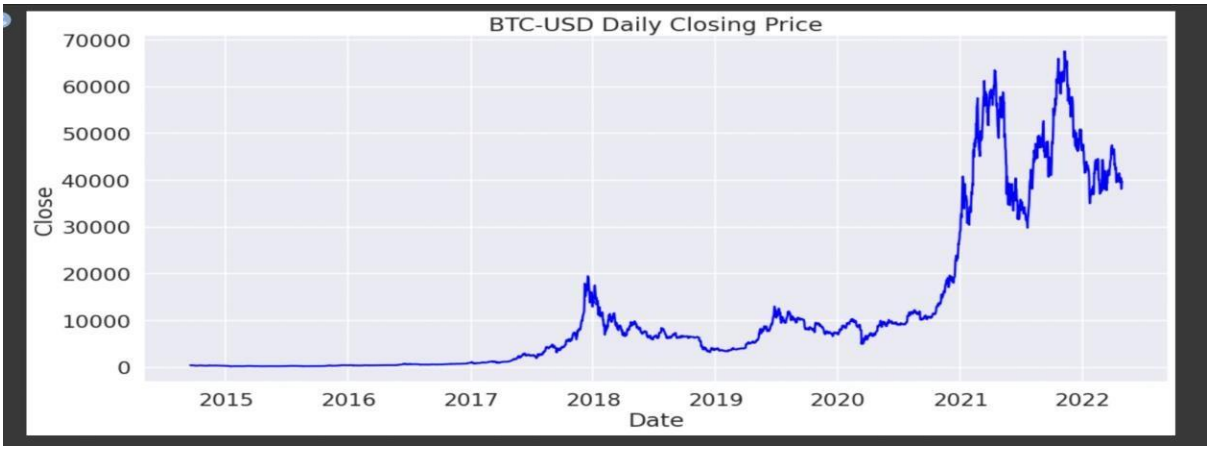


Fig:6.1 Closing Price of BTC-USD

Unlike long-term investing, trading often has a short-term focus. A trader buys a stock not to hold for gradual appreciation but for a relatively quick turnaround, often within a set time period—whether that be a few days, a week, a month, or even a quarter. And of course, day trading, as the name implies, has among the shortest time frames of all. The day trader's analysis may be broken down into hours, minutes, and even seconds—and the time of day when a trade is made can be an important factor to consider. Is there a best day of the week to buy stocks? Or a best day to sell stock? Does a best time of year to buy stocks exist? How about a best month to buy stocks, or to sell them. irst thing in the morning, market volumes and prices can go wild. The opening hours are when the market factors in all of the events and news releases since the previous closing bell, which contributes to price volatility. A skilled trader may be able to recognize the appropriate patterns and make a quick profit, but a less skilled trader could suffer serious losses as a result. So if you're a novice, you may want to avoid trading during these volatile hours, or at least within the first hour.However, for seasoned day traders, the first 15 minutes following the opening bell is prime time, usually offering some of the biggest trades of the day on the initial trends. The opening 9:30 a.m. to 10:30 a.m. Eastern time (ET) period is often one of the best hours of the day for day trading, offering the biggest moves in the shortest amount of time. A lot of professional day traders stop trading around 11:30 a.m. because that is when volatility and volume tend to taper off. Once that happens, trades take longer and moves are smaller with less volume. If your day trading involves index futures such as S&P 500 E-Minis, or an actively traded index exchange-traded fund (ETF) such as the S&P 500 SPDR (SPY), you can begin trading as early as 8:30 a.m. (premarket) and begin tapering off around 10:30 a.m. As with stocks, trading can continue up to 11:30 a.m., but only if the market is still providing opportunities.The middle of the day tends to be the calmest and most stable period of the trading day. During this time, people are waiting for further news to be announced. Because most of the day's news releases have already been factored into stock prices, many are watching to see where the market may be heading for the remainder of the day.Because prices are relatively stable during this period, it's a good time for a beginner to place trades, as the action is slower and the returns might be more predictable.In the last hours of the trading day, volatility and volume increase again. In fact, common intraday stock market patterns show the last hour can be like the first—sharp reversals and big moves, especially in the last several minutes of trading. From 3 p.m. to 4 p.m. ET, day traders are often trying to close out their positions, or they may be attempting to join a late-day rally in the hope that the momentum will carry forward into the next trading day.

```
[ ]   1 # PRESORT MONTHS IN CHRONOLOGICAL ORDER
      2 ordered_months = ['Jan', 'Feb', 'Mar', 'Apr','May','Jun',
      3                   'Jul', 'Aug','Sep', 'Oct', 'Nov', 'Dec']
      4
      5 # GROUP vol_current BY MONTH AND TAKE THE MEAN
      6 data = df.groupby(by=[df.index.month_name()]).vol_current.mean()
      7
      8 # ABBREVIATE MONTH NAME
      9 data.index = [x[:3] for x in data.index]
     10
     11 # SELECT PALETTE
     12 pal = sns.color_palette("GnBu", len(data))
     13
     14 # SORT MONTH BY AVERAGE vol_current
     15 rank = data.argsort().argsort().reindex(ordered_months)
     16
     17 with sns.axes_style("darkgrid"):
     18     fig, ax = plt.subplots(figsize=(18,7))
     19
     20     sns.boxplot(x=[x[:3] for x in df.index.month_name()],
     21                 y=df.vol_current,
     22                 palette=np.array(pal)[rank],
     23                 order=ordered_months)
     24     ax.set(xlabel='',
     25            ylabel='Daily Volatility',
     26            title='Daily Volatility Grouped By Month')
     27     plt.savefig(os.path.join(directory_to_img, "vol_by_month.png"),
     28                 dpi=300, bbox_inches='tight')
     29     plt.show()
```

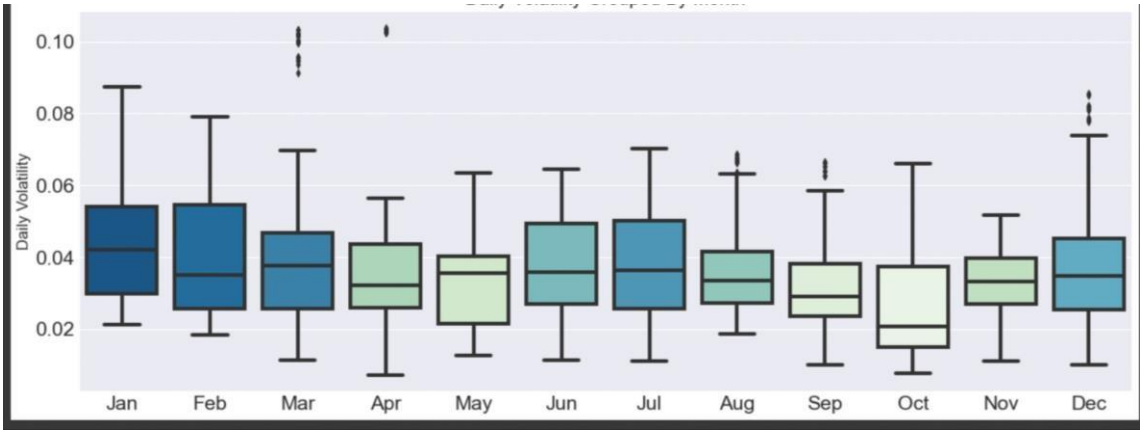


Fig 6.2 Daily volatility is grouped by Month

```
[ ]   1 # GROUP vol_current BY YEAR AND TAKE THE MEAN
      2 data = df.groupby(by=[df.index.year]).vol_current.mean()
      3
      4 # SELECT PALETTE
      5 pal = sns.color_palette("GnBu", len(data))
      6
      7 # SORT MONTH BY AVERAGE vol_current
      8 rank = data.argsort().argsort()
      9
     10 with sns.axes_style("darkgrid"):
     11     fig, ax = plt.subplots(figsize=(18,7))
     12
     13     sns.boxplot(x=df.index.year,
     14                 y=df.vol_current,
     15                 palette=np.array(pal)[rank])
     16     ax.set(xlabel='',
     17            ylabel='Daily Volatility',
     18            title='Daily Volatility Grouped By Year')
     19     plt.savefig(os.path.join(directory_to_img, "vol_by_year.png"),
     20                 dpi=300, bbox_inches='tight')
     21     plt.show()
```

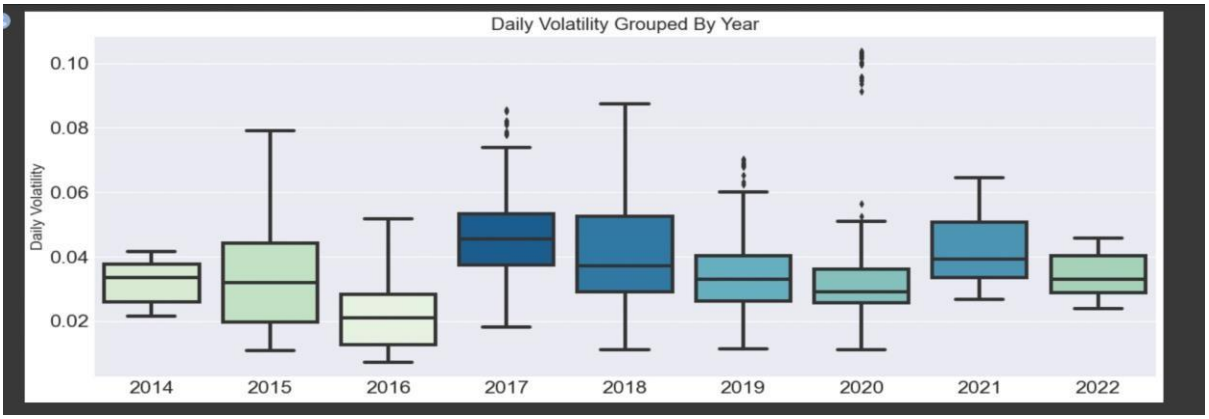


Fig 6.3 Daily volatility is grouped by Year

6.2 SCALING THE DATASET

Many machine learning algorithms perform better when numerical input variables are scaled to a standard range. This includes algorithms that use a weighted sum of the input, like linear regression, and algorithms that use distance measures, like k-nearest neighbors. The two most popular techniques for scaling numerical data prior to modeling are normalization and standardization. **Normalization** scales each input variable separately to the range 0-1, which is the range for floating-point values where we have the most precision. **Standardization** scales each input variable separately by subtracting the mean (called centering) and dividing by the standard deviation to shift the distribution to have a mean of zero and a standard deviation of one. Min-Max scaling is a normalization technique that enables us to scale data in a dataset to a specific range using each feature's minimum and maximum value. Unlike standard scaling, where data are scaled based on the standard normal distribution(with *mean = 0* and *standard deviation = 1*), the min-max scaler uses each column's minimum and maximum value to scale the data series.The scale of data for some features may be significantly different from those of others, which may harm the performance of our models. It is especially the case with algorithms that rely on a measure of distances, such as Neural Networks and KNN.It is also helpful for optimizing machine learning processes like gradient descent and enables convergence to happen faster.It can help improve the performance and speed of the execution of algorithms. Since the data are already scaled-down, complex calculations mainly required to optimise algorithms are faster. It can also be helpful when comparing different datasets or models in terms of their performances.The Min-Max scaler, implemented in sklearn libraries, has been used in many Machine Learning applications such as computer vision, natural language processing, and speech recognition.

```
sns.set_context("paper", font_scale=2)

# VISUALIZE TRAIN/VALIDATION/TEST vol_future BEFORE & AFTER TRAINING
with sns.axes_style("whitegrid"):
    fig, (ax1, ax2) = plt.subplots(nrows=2, ncols=1, figsize=(18,14))

    ax1.plot(df.vol_current, lw=1, color='gray', ls='--',
                label='Current Volatility')
    ax1.plot(y_train, color='blue', label='Original Training Target', lw=2)
    ax1.plot(y_val, color='orange', label='Original Validation Target', lw=2)
    ax1.plot(y_test, color='green', label='Original Test Target', lw=2)

    ax1.title.set_text('Target Future Volatility Before Scaling')

    ax2.plot(transform_volatility_to_scaler(scaler_vol, df.vol_current),
                lw=1, color='gray', ls='--',
                label='Scaled Current Volatility')
    ax2.plot(y_train_scaled, color='blue', label='Scaled Training Target', lw=2)
    ax2.plot(y_val_scaled, color='orange', label='Scaled Validation Target', lw=2)
    ax2.plot(y_test_scaled, color='green', label='Scaled Test Target', lw=2)

    ax2.title.set_text('Target Future Volatility After Scaling')

    ax1.legend(loc='upper left', prop={'size': 13}, frameon=True)
    ax2.legend(loc='upper left', prop={'size': 13}, frameon=True)
    plt.savefig(os.path.join(directory_to_img, 'train_val_test_org_scled.png'),
                dpi=300, bbox_inches='tight')
    plt.show();
```

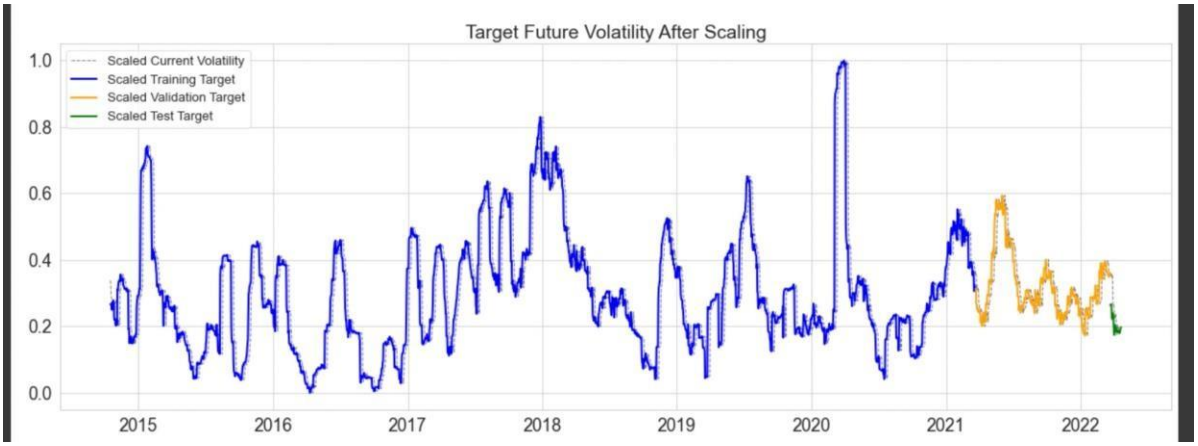


Fig 6.4 Scaling Volatility

6.3 PERFORMANCE METRICS

Usually with financial time series, if we just shift through the historic data trying different methods, parameters and timescales, it's almost certain to find to some strategy with in-sample profitability at some point. However the whole purpose of "forecasting" is to predict the future based on currently available information, and a model that performs best on training data might not be the best when it comes to out-of-sample generalization (or **overfitting**). Avoiding/Minimizing overfitting is even more important in the constantly evolving financial markets where the stake is high.The 2 main metrics I'd be using are **RMSPE (Root Mean Squared Percentage Error)** and **RMSE (Root Mean Square Errors)** with RMSPE prioritized. Timescaling is very important in the calculation of volatility due to the level of freedom in frequency/interval window selection. Therefore I think RMSPE would help capture degree of errors compared to desired target values better than other metrics. Also RMSPE would punish large errors more than regular MAPE (Mean Absolute Percentage Error), which is what I want to do here.RMSE and RMSPE would be tracked across different models' performance on validation set forecasting to indicate their abilities to generalize on out-of-sample data. The **root-mean-square deviation** (**RMSD**) or **root-mean-square error** (**RMSE**) is a frequently used measure of the differences between values (sample or population values) predicted by a model or an estimator and the values observed. The RMSD represents the square root of the second sample moment of the differences between predicted values and observed values or the quadratic mean of these differences. These deviations are called *residuals* when the calculations are performed over the data sample that was used for estimation and are called *errors* (or prediction errors) when computed out-of-sample. The RMSD serves to aggregate the magnitudes of the errors in predictions for various data points into a single measure of predictive power. RMSD is a measure of accuracy, to compare forecasting errors of different models for a particular dataset and not between datasets, as it is scale-dependent.[1]RMSD is always non-negative, and a value of 0 (almost never achieved in practice) would indicate a perfect fit to the data. In general, a lower RMSD is better than a higher one. However, comparisons across different types of data would be invalid because the measure is dependent on the scale of the numbers used.RMSD is the square root of the average of squared errors. The effect of each error on RMSD is proportional to the size of the squared error; thus larger errors have a disproportionately large effect on RMSD. Consequently, RMSD is sensitive to outliers.

```
1
2 plt.style.use(['fivethirtyeight'])
3
4 fig, ax = plt.subplots(figsize=(18,7))
5
6 for i in intervals:
7     if i == 7:
8         alpha = 0.5
9         lw = 1
10    else:
11        alpha = 1.0
12        lw = 2
13    ax.plot(vols_df[i], label=f'{i}-Day Interval Realized Volatility',
14            alpha=alpha, lw=lw)
15
16 ax.set_title('Realized Volatility Using Different Interval Windows', fontsize=21)
17
18 plt.legend(loc='best', prop={'size': 14})
19 plt.savefig(os.path.join(directory_to_img, 'diff_intervals.png'),
20            dpi=300, bbox_inches='tight')
21 plt.show();
```

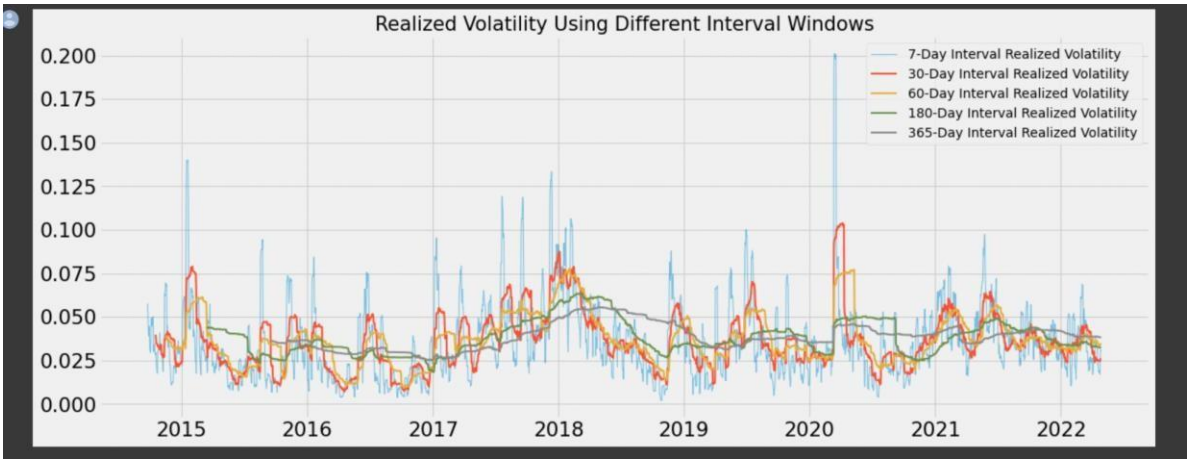


Fig 6.5 Realized Volatility in different interval

6.4 Models Implemented

6.4.1 Baseline model

A baseline is a simple model that provides reasonable results on a task and does not require much expertise and time to build. Common baseline models include linear regression when predicting continuous values, logistic regression when classifying structured data, pretrained convolutional neural networks for vision related tasks, and recurrent neural networks and gradient boosted trees for sequence modeling (Ameisen 2018).

Because it is simple, a baseline is not perfect. For example, the bag of words model for language modeling does not take into account the order of the words occurring in each sentence, hence, it is missing out on a lot of structural information. Also, many baselines largely depend on heuristics embedded in the model or on strong modelling assumptions (i.e. linear models). Finally, from a researcher's perspective, a baseline does not provide cutting-edge research experience which makes it harder to publish.

If baselines are not perfect, then what makes them useful? Let's circle back to the apparel example. Common sense tells us to mail the top 100k loyal customers to maximize our expected return. Loyalty can be measured in terms of three features. If customers shop a lot, spend a lot, and have recently shopped with your store, then they are likely to be loyal (Ramakrishnan 2018). One way to combine these three pieces of information is binning, where we categorize each customer for each category and then simply rank them based on some scale. The top 100k customers are the ones you should mail. In fact, this model is a practical heuristic, called the Recency-Frequency-Monetary Heuristic, which is commonly used in database marketing and direct marketing. It is easy to create, easy to explain, easy to use, and effective.

```python
# PLOTTING MODEL PREDICTIONS VS. TARGET VALUES
def viz_model(y_true, y_pred, model_name):
    sns.set_context("paper", font_scale=1.7)
    plt.rcParams["axes.grid"] = False

    with sns.axes_style("whitegrid"):
        plt.figure(figsize=(18,7))
        plt.plot(x_val_scaled, color='gray',  ls=':',
                 label=f"Scaled Current Daily Volatility")

        plt.plot(y_true, color='blue', lw=2,
                 label=f"Target Volatility")
        plt.plot(y_pred, color='orange', lw=2.5,
                 label=f'Forecasted Volatility')

        # plt.plot(lr_val, color='gray', alpha=0.4,
        #          label='Daily Log Returns')

        plt.title(f'{model_name} \non Validation Data')
        plt.legend(loc='best', frameon=True)
```

	Model	Validation RMSPE	Validation RMSE
0	Mean Baseline	0.249531	0.098188

```python
mean_train_vol
```

```
0.29685338272177286
```

```python
# CREATE SERIES OF PREDICTIONS FOR BASELINE MODEL ON VALIDATION SET
baseline_preds = np.ones(len(val_idx)) * mean_train_vol
baseline_preds = pd.Series(baseline_preds, index=lr_val.index)
```

Fig 6.6 Baseline Model

6.4.2 Random walk Naïve Forecasting

The random walk model is widely used in the area of finance. The stock prices or exchange rates (Asset prices) follow a random walk. A common and serious departure from random behavior is called a random walk (non-stationary), since today's stock price is equal to yesterday stock price plus a random shock.

There are two types of random walks

1. Random walk without drift (no constant or intercept)
2. Random walk with drift (with a constant term)

A time_series said to follow a random walk if the first differences (difference from one observation to the next observation) are random.
Note that in a random walk model, the time series itself is not random, however, the first differences of time series are random (the differences changes from one period to the next).

A random walk model for a time series Xt can be written as

$Xt=Xt-1+et,$

where Xt is the value in time period t, $Xt-1$ is the value in time period $t-1$ plus a random shock et (value of error term in time period t).

Since the random walk is defined in terms of first differences, therefore, it is easier to see the model as

$Xt-Xt-1=et,$

where the original time series is changed to a first difference time series, that is the time series is transformed.

The transformed time series:

- Forecast the future trends to aid in decision making
- If time series follows random walk, the original series offers little or no insights
- May need to analyze first differenced time series

```
random_walk_preds = x_val_scaled
```

```
viz_model(y_val_scaled, random_walk_preds, 'Random Walk Naive Forecasting')
plt.savefig(os.path.join(directory_to_img, 'naive.jpg'),
            dpi=300, bbox_inches='tight')
plt.show();
```

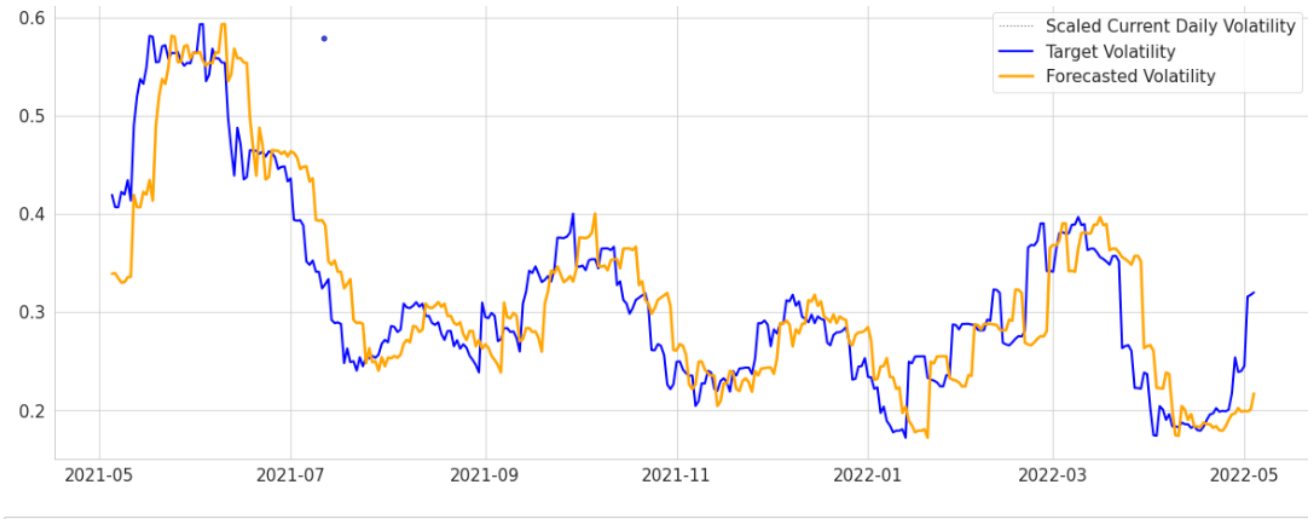


Fig 6.7Random Walk Naïve forecasting

6.4.3 GARCH MODELS

GARCH stands for **Generalized Autoregressive Conditional Heteroskedasticity**, which is an extension of the ARCH model (Autoregressive Conditional Heteroskedasticity).

GARCH includes lag variance terms with lag residual errors from a mean process, and is the traditional econometric approach to volatility prediction of financial time series.

Mathematically, GARCH can be represented as follows:

$\sigma 2t=\omega+\sum iq\alpha i\epsilon 2t-i+\sum 1p\beta i\sigma 2t-i$σt2=ω+∑iqαiϵt−i2+∑1pβiσt−i2

in which $\sigma 2t$σt2 is variance at time step t and $\epsilon 2t-i$ϵt−i2 is the model residuals at time step t-1

GARCH(1,1) only contains first-order lagged terms and the mathematic equation for it is:

$\sigma 2t=\omega+\alpha\epsilon 2(t-1)+\beta\sigma 2(t-1)$σt2=ω+αϵ(t−1)2+βσ(t−1)2

where αα, ββ and ωω sum up to 1, and ωω is the long term variance.

GARCH is generally regarded as an insightful improvement on naively assuming future volatility will be like the past, but also considered widely overrated as predictor by some experts in the field of volatility. GARCH models capture the essential characteristics of volatility: volatility tomorrow will be close to what it is today (**clustering**), and volatility in the long term will probably **mean revert** (meaning it'd be close to whatever the historical long-term average has been).

```
1 from arch import arch_model
```

```
1 # SET SEED FOR REPRODUCIBILITY
2 np.random.seed(seed)
3
4 gm_1 = arch_model(r_train, p=7, q=7)
5 result_1 = gm_1.fit(disp='off')
6 print()
7 print(result_1.summary())
```

```
                     Constant Mean - GARCH Model Results
==============================================================================
Dep. Variable:                returns   R-squared:                       0.000
Mean Model:             Constant Mean   Adj. R-squared:                  0.000
Vol Model:                      GARCH   Log-Likelihood:               -6276.52
Distribution:                  Normal   AIC:                           12585.0
Method:            Maximum Likelihood   BIC:                           12677.2
                                        No. Observations:                 2349
Date:                Fri, Apr 29 2022   Df Residuals:                     2348
Time:                        19:24:31   Df Model:                            1
                            Mean Model
==============================================================================
                 coef     std err           t       P>|t|     95.0% Conf. Int.
------------------------------------------------------------------------------
mu             0.2272  6.770e-02       3.357  7.889e-04 [9.456e-02,  0.360]
                          Volatility Model
==============================================================================
                 coef     std err           t       P>|t|     95.0% Conf. Int.
------------------------------------------------------------------------------
omega          2.0570       3.380       0.609       0.543   [ -4.567,  8.681]
alpha[1]       0.1632  4.580e-02       3.564  3.659e-04 [7.344e-02,  0.253]
alpha[2]       0.0562       0.185       0.304       0.761   [ -0.306,  0.419]
alpha[3]       0.0000       0.168       0.000       1.000   [ -0.330,  0.330]
alpha[4]       0.1412       0.125       1.130       0.258   [ -0.104,  0.386]
alpha[5]       0.0143       0.207   6.898e-02       0.945   [ -0.392,  0.421]
alpha[6]       0.0635       0.157       0.405       0.685   [ -0.244,  0.371]
alpha[7]       0.0000       0.121       0.000       1.000   [ -0.238,  0.238]
beta[1]        0.0000       1.072       0.000       1.000   [ -2.102,  2.102]
beta[2]        0.2483       0.952       0.261       0.794   [ -1.618,  2.115]
beta[3]        0.0000       0.431       0.000       1.000   [ -0.844,  0.844]
beta[4]        0.2275       0.568       0.400       0.689   [ -0.887,  1.341]
beta[5]        0.0000       0.373       0.000       1.000   [ -0.732,  0.732]
beta[6]        0.0000       0.271       0.000       1.000   [ -0.531,  0.531]
beta[7]  1.8830e-13       0.226   8.342e-13       1.000   [ -0.442,  0.442]
==============================================================================

Covariance estimator: robust
```

```
gm_1 = arch_model(r_train, vol='GARCH', p=1, q=1)
result_1 = gm_1.fit(disp='off')
print()
print(result_1.summary())
```

```
                     Constant Mean - GARCH Model Results
==============================================================================
Dep. Variable:                returns   R-squared:                       0.000
Mean Model:             Constant Mean   Adj. R-squared:                  0.000
Vol Model:                      GARCH   Log-Likelihood:               -6293.14
Distribution:                  Normal   AIC:                           12594.3
Method:            Maximum Likelihood   BIC:                           12617.3
                                        No. Observations:                 2349
Date:                Fri, Apr 29 2022   Df Residuals:                     2348
Time:                        19:24:38   Df Model:                            1
                            Mean Model
==============================================================================
                 coef    std err          t      P>|t|      95.0% Conf. Int.
------------------------------------------------------------------------------
mu             0.2392  6.386e-02      3.745  1.801e-04 [  0.114,  0.364]
                          Volatility Model
==============================================================================
                 coef    std err          t      P>|t|      95.0% Conf. Int.
------------------------------------------------------------------------------
omega          0.6833      0.263      2.601  9.287e-03   [  0.168,  1.198]
alpha[1]       0.1333  3.307e-02      4.032  5.536e-05 [6.851e-02,  0.198]
beta[1]        0.8341  2.996e-02     27.846 1.213e-170   [  0.775,  0.893]
==============================================================================

Covariance estimator: robust
```

Fig 6.8 Forecasted Volatility For basic Garch Model

6.4.4 GJR-GARCH Model

The basic GARCH model assumes positive and negative news have similar impact on volatility. However, in reality the market tends to "*take the stairs up and the elevator down*". In other words, the impact is usually asymmetric, and negative impacts tends to affect the volatility more than positive ones.There's another member in the GARCH family that accounts for assymmetry of shocks reponses called **GJR-GARCH** (short for **Glosten-Jagannathan-Runkle GARCH**).

```
                     Constant Mean - GJR-GARCH Model Results
==============================================================================
Dep. Variable:                    returns   R-squared:                       0.000
Mean Model:                Constant Mean   Adj. R-squared:                  0.000
Vol Model:                     GJR-GARCH   Log-Likelihood:               -5932.89
Distribution:      Standardized Skew Student's t   AIC:                   11879.8
Method:                Maximum Likelihood   BIC:                         11920.1
                                            No. Observations:                2349
Date:                   Fri, Apr 29 2022   Df Residuals:                    2348
Time:                           19:29:11   Df Model:                           1
                              Mean Model
============================================================================
                 coef    std err          t      P>|t|    95.0% Conf. Int.
----------------------------------------------------------------------------
mu             0.1949  5.025e-02      3.878  1.052e-04 [9.639e-02,  0.293]
                           Volatility Model
============================================================================
                 coef    std err          t      P>|t|        95.0% Conf. Int.
----------------------------------------------------------------------------
omega          0.1206      0.105      1.151      0.250    [-8.469e-02,  0.326]
alpha[1]       0.1281  1.825e-02      7.019  2.235e-12     [9.232e-02,  0.164]
gamma[1]      -0.0440  2.048e-02     -2.149  3.167e-02 [-8.415e-02,-3.863e-03]
beta[1]        0.8939  2.878e-02     31.062 7.881e-212       [  0.838,  0.950]
                            Distribution
============================================================================
                 coef    std err          t      P>|t|       95.0% Conf. Int.
----------------------------------------------------------------------------
eta            3.2076      0.161     19.917  2.903e-88      [  2.892,  3.523]
lambda     3.3017e-03  2.280e-02      0.145      0.885 [-4.138e-02,4.798e-02]
============================================================================

Covariance estimator: robust
```

Fig 6.9 Forecasted Volatility using GJR-GARCH model

6.4.5 TARCH Model

There's another member in the GARCH family called **TARCH**, which is short for **Threshold Autoregressive Conditional Heteroskedasticity** (and also known as **ZARCH**). TARCH models the volatility using absolute values (instead of squares). This model is specified using power=1.0 since the default power, 2.0, corresponds to variance processes that evolve in squares. In addition, asymmetric impact is also incorporated into the GARCH framework by using a dummy variable

The volatility process in a TARCH(1,1) model is given by:

$\sigma t=\omega+\alpha|\epsilon t-1|+\gamma|\epsilon t-1|[\epsilon t-1<0]+\beta\sigma t-1$

```
                     Constant Mean - TARCH/ZARCH Model Results
==============================================================================================
Dep. Variable:                           returns   R-squared:                          0.000
Mean Model:                         Constant Mean   Adj. R-squared:                     0.000
Vol Model:                            TARCH/ZARCH   Log-Likelihood:                 -5931.23
Distribution:      Standardized Skew Student's t   AIC:                               11876.5
Method:                         Maximum Likelihood   BIC:                               11916.8
                                                     No. Observations:                   2349
Date:                            Fri, Apr 29 2022   Df Residuals:                       2348
Time:                                    19:51:13   Df Model:                             1
                                 Mean Model
============================================================================
                 coef    std err          t      P>|t|    95.0% Conf. Int.
----------------------------------------------------------------------------
mu             0.1778  5.492e-02      3.237  1.207e-03 [7.015e-02,  0.285]
                           Volatility Model
============================================================================
                 coef    std err          t      P>|t|       95.0% Conf. Int.
----------------------------------------------------------------------------
omega          0.1325      0.267      0.496      0.620      [ -0.391,  0.656]
alpha[1]       0.1489  5.480e-02      2.718  6.574e-03    [4.152e-02,  0.256]
gamma[1]      -0.0384  2.578e-02     -1.490      0.136 [-8.893e-02,1.211e-02]
beta[1]        0.8703      0.129      6.734  1.650e-11      [  0.617,  1.124]
                            Distribution
============================================================================
                  coef    std err          t      P>|t|       95.0% Conf. Int.
----------------------------------------------------------------------------
eta             3.2701      0.249     13.120  2.526e-39      [  2.782,  3.759]
lambda     -1.3327e-03  2.354e-02 -5.663e-02      0.955 [-4.746e-02,4.480e-02]
============================================================================

Covariance estimator: robust
```

Forecasting using the Bootstrap model of constant mean skewed students T's distribution

```
# ROLLING WINDOW FORECAST
# INITIALIZING rolling_forecasts VALUES LIST
rolling_forecasts = []

# ITERATE OVER EACH TIME STEP IN THE VALIDATION SET
for i in range(len(val_idx)):
    # GET THE DATA AT ALL PREVIOUS TIME STEPS
    idx = val_idx[i]
    train = df.returns[:idx].dropna()

    # TRAIN MODEL USING ALL PREVIOUS TIME STEPS' DATA
    model = arch_model(train, p=1, o=1, q=1, power=1.0,
                       dist='skewt')
    model_fit = model.fit(disp='off')

    # MAKE PREDICTION n_future DAYS OUT
    # USING BOOTSTRAP METHOD
    varR = model_fit.forecast(horizon=n_future,
                              reindex=False,
                              method='bootstrap').variance.values
    pred = np.sqrt(np.mean(varR))

    # APPEND TO rolling_forecasts LIST
    rolling_forecasts.append(pred)

t_bs_preds = pd.DataFrame(rolling_forecasts, index=val_idx)
```

Fig 6.10 Bootstrapped forecasted Volatility

Simulation Forecasting of Tarch model

```
# ROLLING WINDOW FORECAST
# INITIALIZING rolling_forecasts VALUES LIST
rolling_forecasts = []

# ITERATE OVER EACH TIME STEP IN THE VALIDATION SET
for i in range(len(val_idx)):
    # GET THE DATA AT ALL PREVIOUS TIME STEPS
    idx = val_idx[i]
    train = df.returns[:idx].dropna()

    # TRAIN MODEL USING ALL PREVIOUS TIME STEPS' DATA
    model = arch_model(train, p=1, o=1, q=1, power=1.0,
                       dist='skewt')
    model_fit = model.fit(disp='off')

    # MAKE PREDICTION n_future DAYS OUT
    # USING SIMULATION METHOD
    vaR = model_fit.forecast(horizon=n_future,
                             reindex=False,
                             method='simulation').variance.values
    pred = np.sqrt(np.mean(vaR))

    # APPEND TO rolling_forecasts LIST
    rolling_forecasts.append(pred)
```

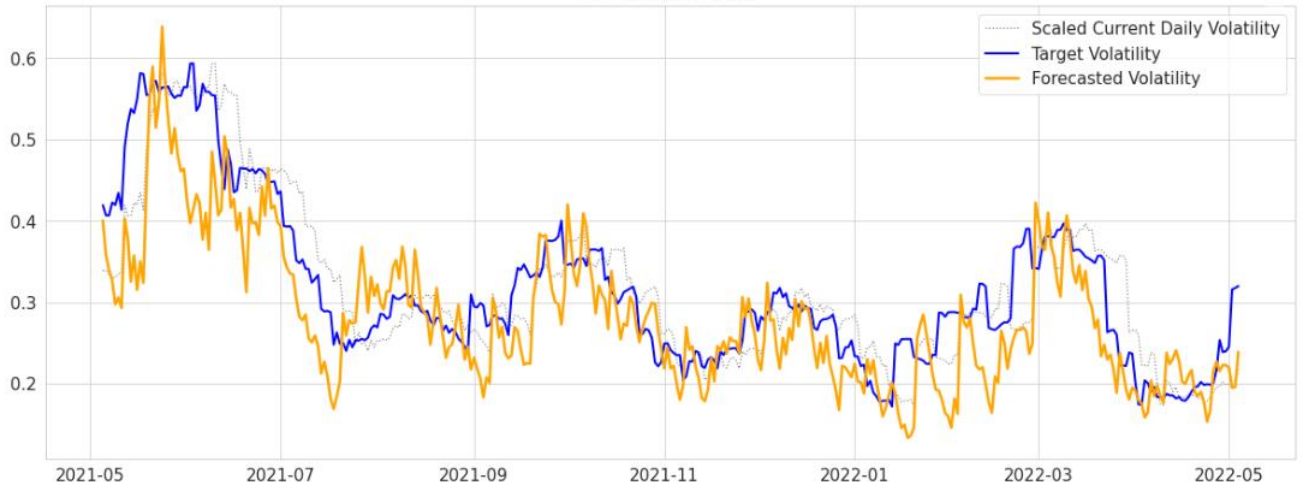
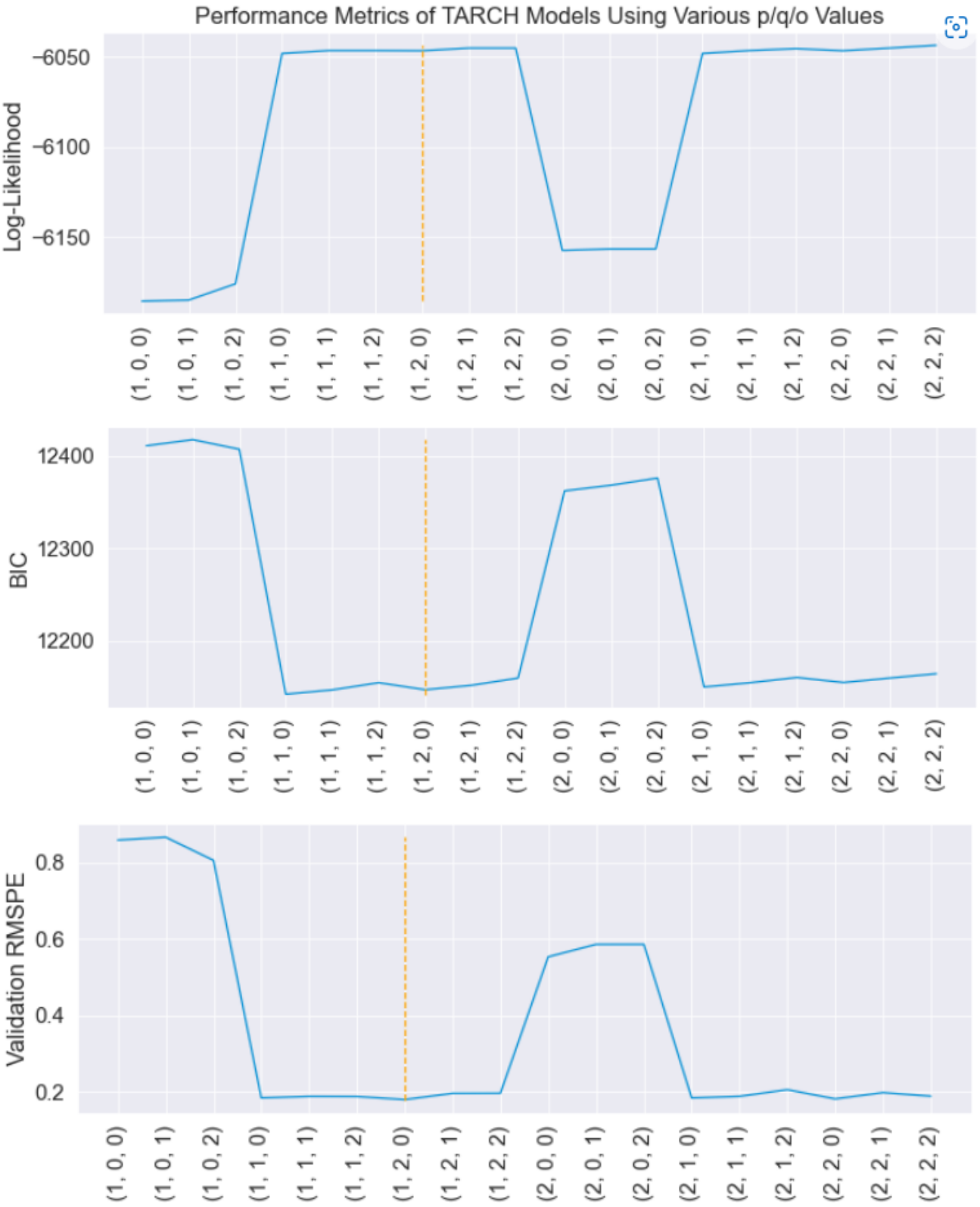


Fig 6.11 Simulation based forecasted Volatilty

Hyper Parameter Tuning for TARCH

Hyperparameter tuning is an essential part of controlling the behavior of a machine learning model. If we don't correctly tune our hyperparameters, our estimated model parameters produce suboptimal results, as they don't minimize the loss function. This means our model makes more errors. In practice, key indicators like the accuracy or the confusion matrix will be worse. n machine learning, we need to differentiate between parameters and hyperparameters. A learning algorithm learns or estimates model parameters for the given data set, then continues updating these values as it continues to learn. After learning is complete, these parameters become part of the model. For example, each weight and bias in a neural network is a parameter.Hyperparameters, on the other hand, are specific to the algorithm itself, so we can't calculate their values from the data. We use hyperparameters to calculate the model parameters. Different hyperparameter values produce different model parameter values for a given data set.Hyperparameter tuning consists of finding a set of optimal hyperparameter values for a learning algorithm while applying this optimized algorithm to any data set. That combination of hyperparameters maximizes the model's performance, minimizing a predefined loss function to produce better results with fewer errors. Note that the learning algorithm optimizes the loss based on the input data and tries to find an optimal solution within the given setting. However, hyperparameters describe this setting exactly. Itertool is one of the most amazing Python 3 standard libraries. This library has pretty much coolest functions and nothing wrong to say that it is the gem of the Python programing language. Python provides excellent documentation of the itertools but in this tutorial, we will discuss few important and useful functions or iterators of itertools.The key thing about itertools is that the functions of this library are used to make memory-efficient and precise code. According to the official definition of itertools, "**this module implements a number of iterator building blocks inspired by constructs from APL, Haskell, and SML**." In simple words, the number of iterators can together create 'iterator algebra' which makes it possible to complete the complex task. The functions in itertools are used to produce more complex iterators.


Performance Metrics of TARCH Models Using Various p/q/o Values
Log-Likelihood
−6050
−6100
−6150
BIC
12400
12300
12200
Validation RMSPE
0.8
0.6
0.4
0.2
(1, 0, 0)
(1, 0, 1)
(1, 0, 2)
(1, 1, 0)
(1, 1, 1)
(1, 1, 2)
(1, 2, 0)
(1, 2, 1)
(1, 2, 2)
(2, 0, 0)
(2, 0, 1)
(2, 0, 2)
(2, 1, 0)
(2, 1, 1)
(2, 1, 2)
(2, 2, 0)
(2, 2, 1)
(2, 2, 2)

FINAL TARCH

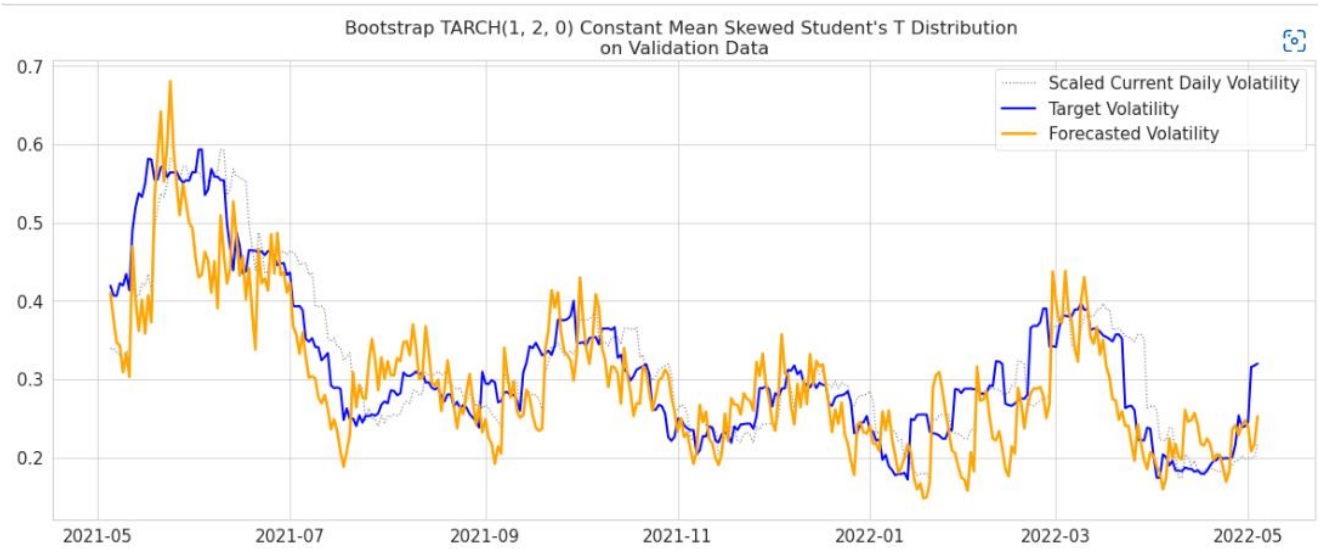


Fig 6.12 Forecasted Volatility of TARCH(1,2,0)

6.5 NEURAL NETWORKS

While GARCH remains the gold standard for volatility prediction within traditional financial institutions, there has been an increasing numbers of professionals and researchers turning to Machine Learning, especially Neural Networks, to gain insights into the financial markets in recent years.Traders' theory of the market being inherently efficient (Efficient Market Hypothesis or EHM) states that share prices reflects all information and consistently outperforming the overall market is impossible. The more efficient a market is, the more random and unpredictable the returns will be, and thus **a perfectly efficient market will be completely unpredictable**. There are other arguments against EMH, and ones of the most prominent one is based on **Behavioral Finance**: compared to the human history of 200,000 years, the market has not been around for that long. For example, equity options have only been traded in liquid, transparent market since the CBOE opened in 1973; and the average lifetime of an S&P500 company is approx. 20 years. It means that some psychological tendencies of human beings have 200,000 years of evidence behind them, and that a lot of the movements of the markets that were driven by participants' behaviors will likely repeat itself at a later point. Therefore the market system cannot be totally random, it must have some patterns. Those patterns are extremely difficult to exploit due to the multitude of factors that interact and drive the market.It'd be interesting to see how Neural Networks perform compared to the traditional GARCH models.

6.5.1 Neural Networks Baseline Metrics

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 flatten (Flatten)           (None, 14)                0

 dense (Dense)               (None, 1)                 15

=================================================================
Total params: 15
Trainable params: 15
Non-trainable params: 0
_________________________________________________________________
None
```

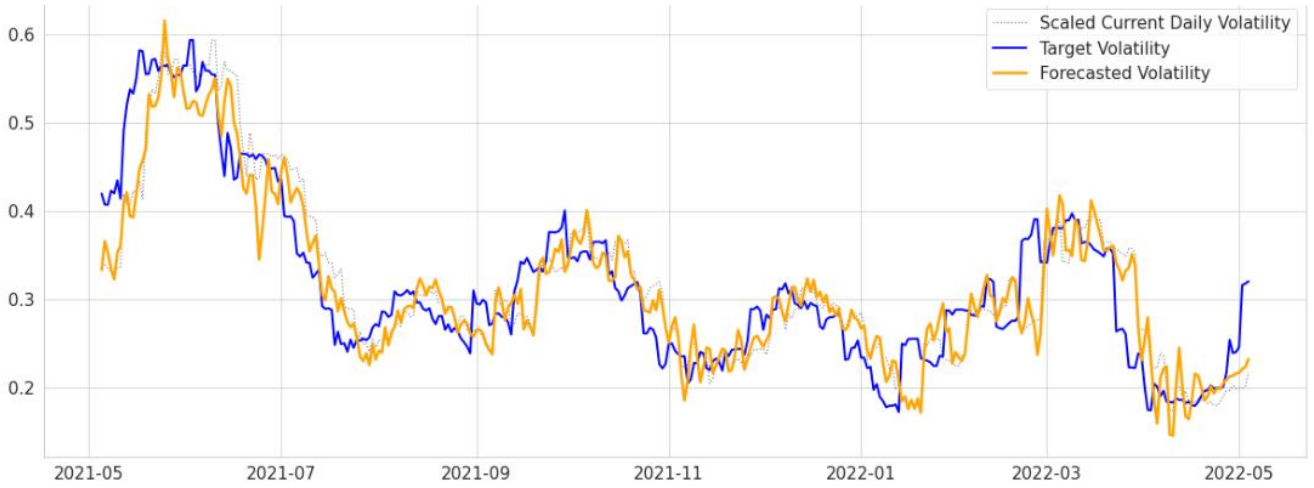
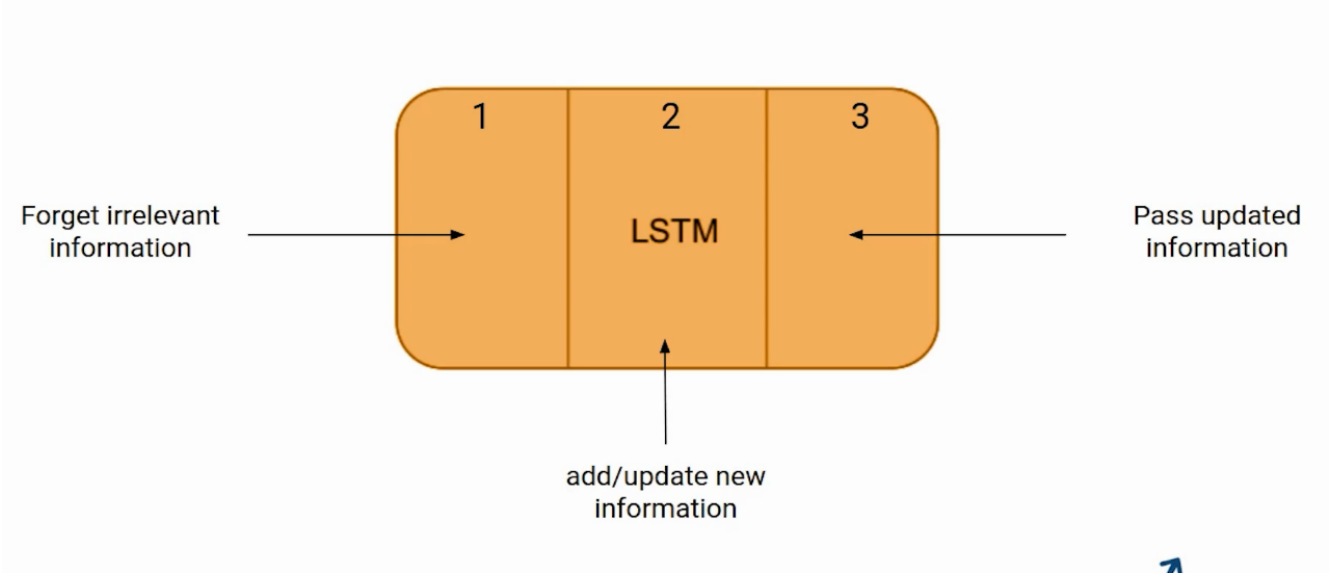


Fig 6.13 Forecasted Volatility for fully connected network

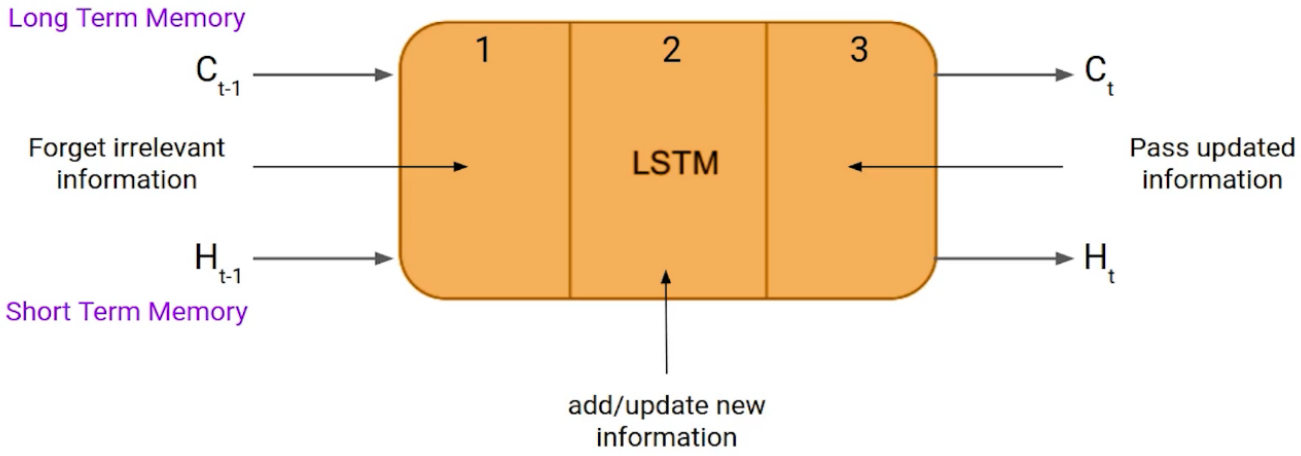
6.5.2 LONG SHORT TERM MEMORY

6.5.2.1 1-Layered LSTM using RNN with 14 days lookback window on the validation date.

Long Short Term Memory Network is an advanced RNN, a sequential network, that allows information to persist. It is capable of handling the vanishing gradient problem faced by RNN. A recurrent neural network is also known as RNN is used for persistent memory. At a high-level LSTM works very much like an RNN cell. Here is the internal functioning of the LSTM network. The LSTM consists of three parts, as shown in the image below and each part performs an individual function.



The first part chooses whether the information coming from the previous timestamp is to be remembered or is irrelevant and can be forgotten. In the second part, the cell tries to learn new information from the input to this cell. At last, in the third part, the cell passes the updated information from the current timestamp to the next timestamp.These three parts of an LSTM cell are known as gates. The first part is called **Forget gate, the** second part is known as **the Input gate** and the last one is **the Output gate**. Just like a simple RNN, an LSTM also has a hidden state where H(t-1) represents the hidden state of the previous timestamp and Ht is the hidden state of the current timestamp. In addition to that LSTM also have a cell state represented by C(t-1) and C(t) for previous and current timestamp respectively.Here the hidden state is known as Short term memory and the cell state is known as Long term memory.



Forget Gate

In a cell of the LSTM network, the first step is to decide whether we should keep the information from the previous timestamp or forget it. Here is the equation for forget gate.

Forget Gate:

- $f_t = \sigma(x_t * U_f + H_{t-1} * W_f)$

Let's try to understand the equation, here

- Xt: input to the current timestamp.
- Uf: weight associated with the input
- Ht-1: The hidden state of the previous timestamp
- Wf: It is the weight matrix associated with hidden state

Later, a sigmoid function is applied over it. That will make ft a number between 0 and 1. This ft is later multiplied with the cell state of the previous timestamp as shown below.

$C_{t-1} * f_t = 0$ ...if $f_t = 0$ (forget everything)

$C_{t-1} * f_t = C_{t-1}$ ...if $f_t = 1$ (forget nothing)

If ft is 0 then the network will forget everything and if the value of ft is 1 it will forget nothing. Let's get back to our example, The first sentence was talking about Bob and after a full stop, the network will encounter Dan, in an ideal case the network should forget about Bob.

Input Gate

Let's take another example

"Bob knows swimming. He told me over the phone that he had served the navy for four long years."So, in both these sentences, we are talking about Bob. However, both give different kinds of information about Bob. In the first sentence, we get the information that he knows swimming. Whereas the second sentence tells he uses the phone and served in the navy for four years.Now just think about it, based on the context given in the first sentence, which information of the second sentence is critical. First, he used the phone to tell or he served in the navy. In this context, it doesn't matter whether he used the phone or any other medium of communication to pass on the information. The fact that he was in the navy is important information and this is something we want our model to remember. This is the task of the Input gate.Input gate is used to quantify the importance of the new information carried by the input. Here is the equation of the input gate

Input Gate:

- $i_t = \sigma(x_t * U_i + H_{t-1} * W_i)$

Here,

- Xt: Input at the current timestamp t
- Ui: weight matrix of input

- Ht-1: A hidden state at the previous timestamp
- Wi: Weight matrix of input associated with hidden state

Again we have applied sigmoid function over it. As a result, the value of I at timestamp t will be between 0 and 1.

New information

- $N_t = \tanh(x_t * U_c + H_{t-1} * W_c)$ (new information)

Now the new information that needed to be passed to the cell state is a function of a hidden state at the previous timestamp t-1 and input x at timestamp t. The activation function here is tanh. Due to the tanh function, the value of new information will between -1 and 1. If the value is of Nt is negative the information is subtracted from the cell state and if the value is positive the information is added to the cell state at the current timestamp.However, the Nt won't be added directly to the cell state. Here comes the updated equation

$C_t = f_t * C_{t-1} + i_t * N_t$ (updating cell state)

Here, Ct-1 is the cell state at the current timestamp and others are the values we have calculated previously.

Output Gate

Now consider this sentence

"Bob single-handedly fought the enemy and died for his country. For his contributions, brave_____ ."During this task, we have to complete the second sentence. Now, the minute we see the word brave, we know that we are talking about a person. In the sentence only Bob is brave, we can not say the enemy is brave or the country is brave. So based on the current expectation we have to give a relevant word to fill in the blank. That word is our output and this is the function of our Output gate.

Here is the equation of the Output gate, which is pretty similar to the two previous gates.

Output Gate:

- $o_t = \sigma(x_t * U_o + H_{t-1} * W_o)$

Its value will also lie between 0 and 1 because of this sigmoid function. Now to calculate the current hidden state we will use Ot and tanh of the updated cell state. As shown below.

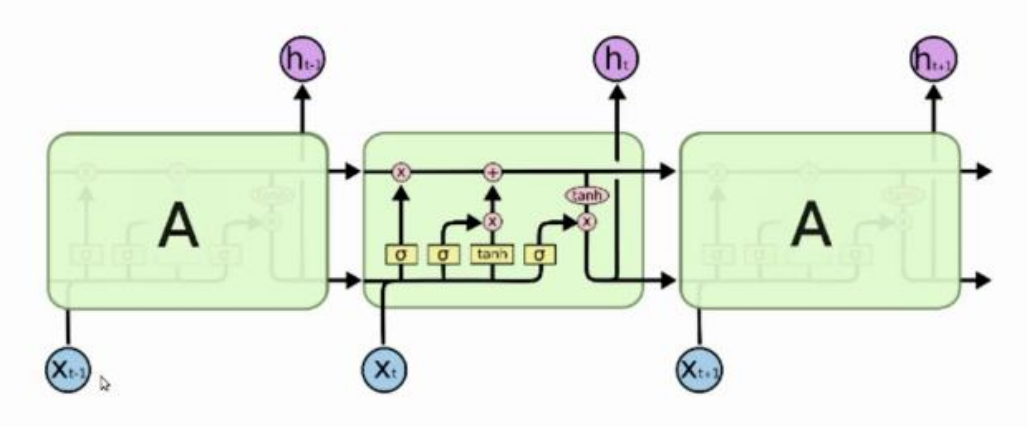
$$H_t = o_t * \tanh(C_t)$$

It turns out that the hidden state is a function of Long term memory (Ct) and the current output. If you need to take the output of the current timestamp just apply the SoftMax activation on hidden state Ht.

$$\text{Output} = \text{Softmax}(H_t)$$

Here the token with the maximum score in the output is the prediction.

This is the More intuitive diagram of the LSTM network.



```
# FORECASTING ON VALIDATION SET
lstm_1_preds = forecast(lstm_1, val_idx)

# SCALING OUTPUT TO MINMAXSCALER FITTED TO TRAINING CURRENT VOLUME
lstm_1_preds_scaled = scale(scaler_vol, lstm_1_preds)
```

```
# PLOTTING PREDICTIONS VS. TARGET VALUES ON VALIDATION SET
viz_model(y_val_scaled, lstm_1_preds_scaled,
          f"1-Layered LSTM RNN (20 units) with {n_past}-Day Lookback Window")
plt.savefig(os.path.join(directory_to_img, 'lstm_1.png'),
            dpi=300, bbox_inches='tight')
plt.show();
```

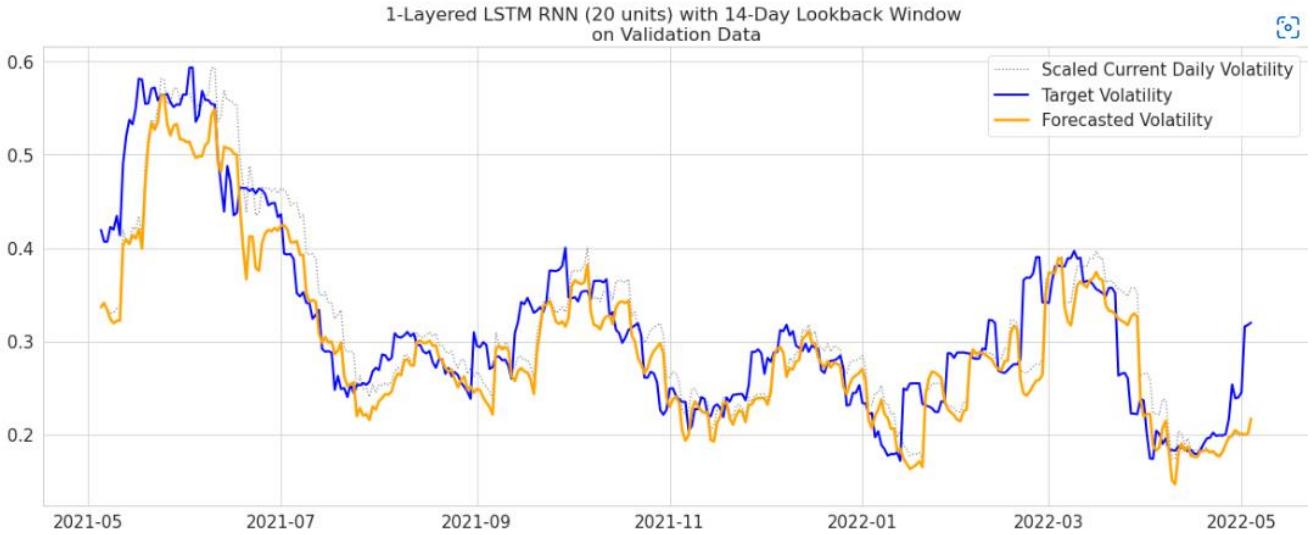
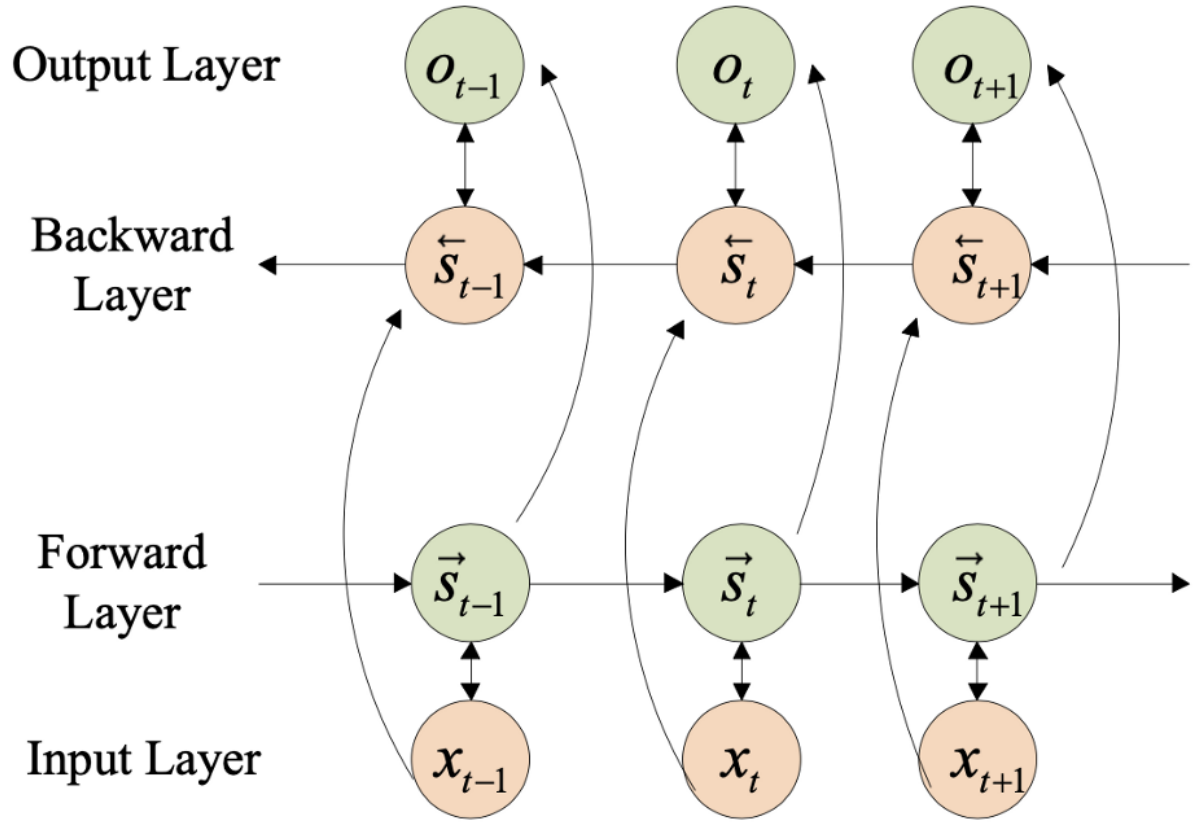


Fig 6.14 Forecasted Volatility for 1 layered LSTM

6.5.2.2 Two Layered bidirectional LSTM

Bidirectional long-short term memory (BiLSTM) is the technique of allowing any neural network to store sequence information in both ways, either backward or forward. Our input runs in two ways in bidirectional, distinguishing a BiLSTM from a standard LSTM. We can have the input flow in both directions; to store past and future information at any time step. Nevertheless, normal LSTMs allow input flow in one direction (forward or backward). Further research shows that there's an extension of LSTM cell called **Bidirectional LSTM**, which could potentially be better in this case by providing additional context to the models. Since all timesteps of the input sequence are already available, Bidirectional LSTM could train 2 instead of 1 LSTMs on the same input sequence:1st one on the inputs as-is 2nd one on the reversed copy of the inputsThis could help provide additional context to the networks, and usually produces faster and fuller learning on the problem.



```
# FORECASTING ON VALIDATION SET
lstm_2_preds = forecast(lstm_2, val_idx)

# SCALING OUTPUT TO MINMAXSCALER FITTED TO TRAINING CURRENT VOLUME
lstm_2_preds_scaled = scale(scaler_vol, lstm_2_preds)
```

```
# PLOTTING PREDICTIONS VS. TARGET VALUES ON VALIDATION SET
viz_model(y_val_scaled, lstm_2_preds_scaled,
          f"2-Layered Bidirectional LSTMs (32/16 units) with {n_past}-Day Lookback Window")
plt.savefig(os.path.join(directory_to_img, 'lstm_2.png'),
            dpi=300, bbox_inches='tight')
plt.show();
```

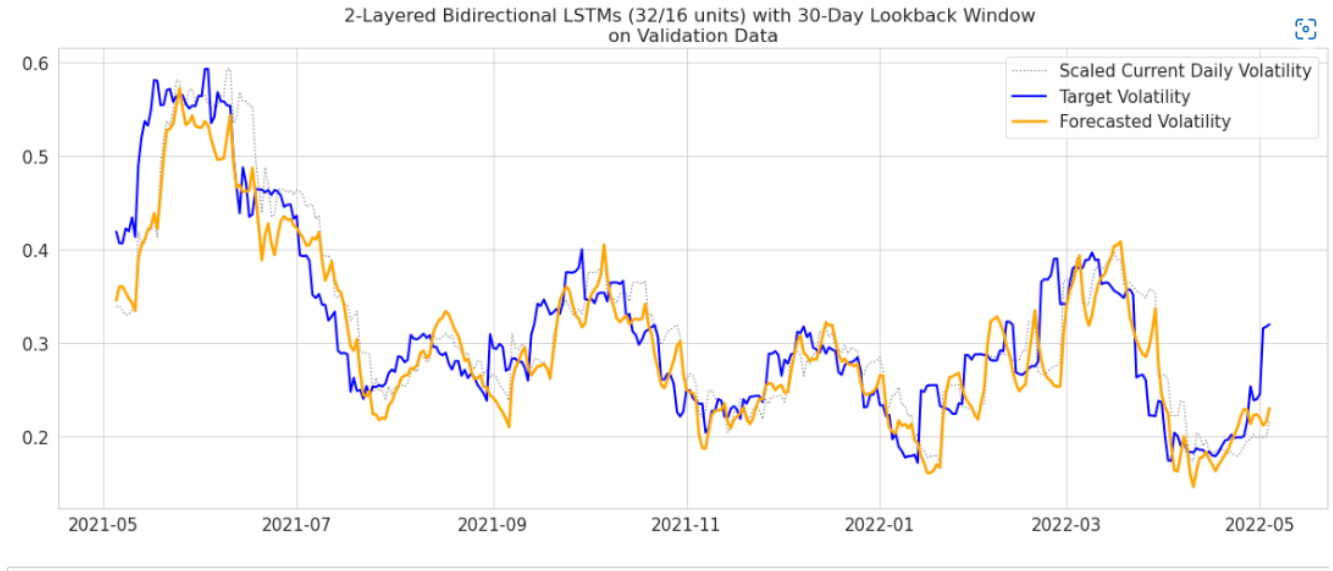
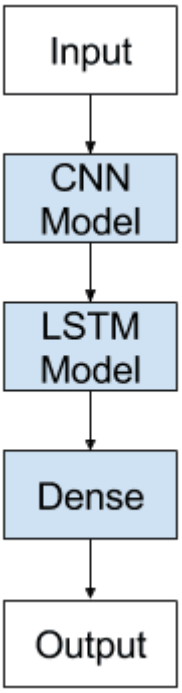


Fig 6.15 Two layered Bidirectional LSTM

6.5.2.3 CNN-LSTM

The CNN LSTM architecture involves using Convolutional Neural Network (CNN) layers for feature extraction on input data combined with LSTMs to support sequence prediction.CNN LSTMs were developed for visual time series prediction problems and the application of generating textual descriptions from sequences of images (e.g. videos). Specifically, the problems of:**Activity Recognition**: Generating a textual description of an activity demonstrated in a sequence of images.**Image Description**: Generating a textual description of a single image.**Video Description**: Generating a textual description of a sequence of images.This architecture was originally referred to as a Long-term Recurrent Convolutional Network or LRCN model, although we will use the more generic name "CNN LSTM" to refer to LSTMs that use a CNN as a front end in this lesson.This architecture is used for the task of generating textual descriptions of images. Key is the use of a CNN that is pre-trained on a challenging image classification task that is re-purposed as a feature extractor for the caption generating problem.



```
# FORECASTING ON VALIDATION SET
lstm_3_preds = forecast(lstm_3, val_idx)

# SCALING OUTPUT TO MINMAXSCALER FITTED TO TRAINING CURRENT VOLUME
lstm_3_preds_scaled = scale(scaler_vol, lstm_3_preds)
```

```
# PLOTTING PREDICTIONS VS. TARGET VALUES ON VALIDATION SET
viz_model(y_val_scaled, lstm_3_preds_scaled,
          f"1 Conv1D 2 Bidirect LSTM layers (32/16), {n_past} days look back, batch={batch_size}")
plt.savefig(os.path.join(directory_to_img, 'lstm_3.png'),
            dpi=300, bbox_inches='tight')
plt.show();
```

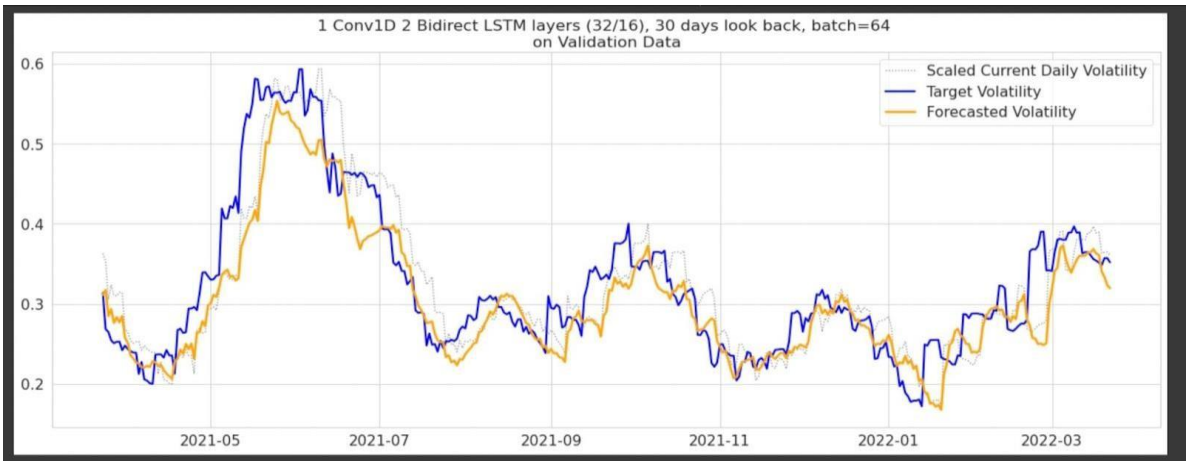


Fig 6.16 Forecasted Volatiltiy for 1 Conv 1D 2 Bidirect LSTM

6.5.2.4 Multivariate 2 Bidirect LSTM layers

```
# FORECASTING ON VALIDATION SET
lstm_5_preds = forecast_multi(lstm_5, val_idx)

# SCALING OUTPUT TO MINMAXSCALER FITTED TO TRAINING CURRENT VOLUME
lstm_5_preds_scaled = scale(scaler_vol, lstm_5_preds)
```

```
# PLOTTING PREDICTIONS VS. TARGET VALUES ON VALIDATION SET
viz_model(y_val_scaled, lstm_5_preds_scaled,
          f"Multivariate 2 Bidirect LSTM layers (32/16 units), {n_past} days look back")
plt.savefig(os.path.join(directory_to_img, 'lstm_5.png'),
            dpi=300, bbox_inch='tight')
plt.show();
```

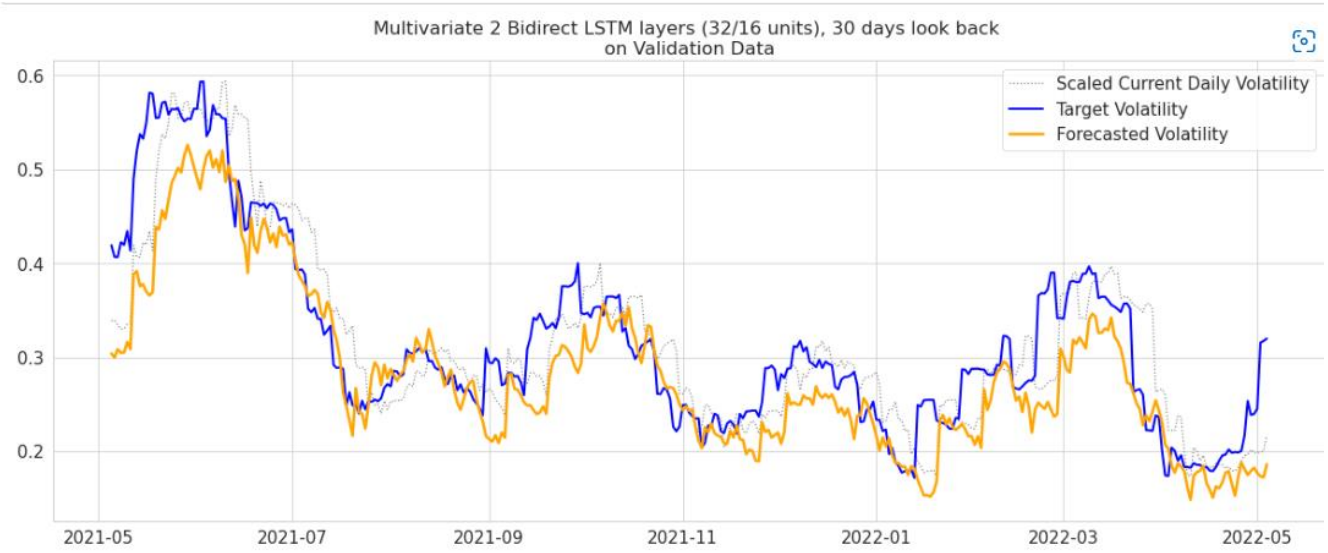


Fig 6.17 Multivariate 2 Bidirect LSTM Layers

6.5.2.5 Multivariate 3 Bidirect LSTM layers

```
# FORECASTING ON VALIDATION SET
lstm_6_preds = forecast_multi(lstm_6, val_idx)

# SCALING OUTPUT TO MINMAXSCALER FITTED TO TRAINING CURRENT VOLUME
lstm_6_preds_scaled = scale(scaler_vol, lstm_6_preds)
```

```
# PLOTTING PREDICTIONS VS. TARGET VALUES ON VALIDATION SET
viz_model(y_val_scaled, lstm_6_preds_scaled,
          f"Multivariate 3 Bidirect LSTM layers (64/32/16 units), {n_past} days look back")
plt.savefig(os.path.join(directory_to_img, 'lstm_6.png'),
            dpi=300, bbox_inch='tight')
plt.show();
```

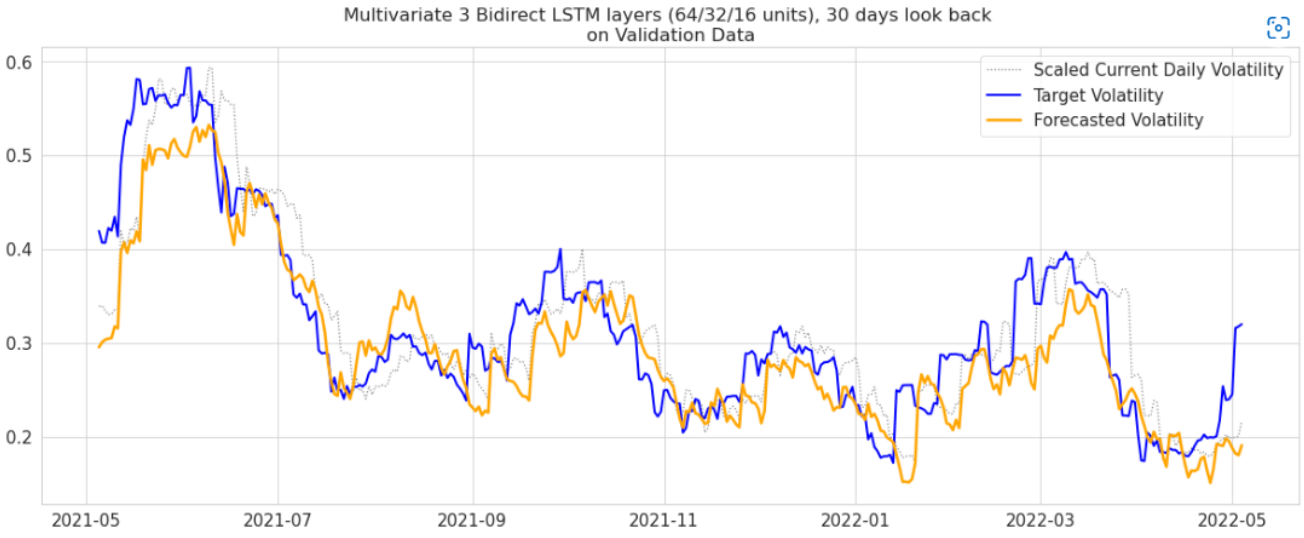


Fig 6.18 Multivariate 3 Bidirect LSTM Layers

6.5.2.6 Multivariate 4 Bidirect LSTM layers

```
# FORECASTING ON VALIDATION SET
lstm_7_preds = forecast_multi(lstm_7, val_idx)

# SCALING OUTPUT TO MINMAXSCALER FITTED TO TRAINING CURRENT VOLUME
lstm_7_preds_scaled = scale(scaler_vol, lstm_7_preds)
```

```
# PLOTTING PREDICTIONS VS. TARGET VALUES ON VALIDATION SET
viz_model(y_val_scaled, lstm_7_preds_scaled,
          f"Multivariate 4 Bidirect LSTM layers (128/64/32/16 units), {n_past} days look back")
plt.savefig(os.path.join(directory_to_img, 'lstm_7_preds.png'),
            dpi=300, bbox_inches='tight')
plt.show();
```

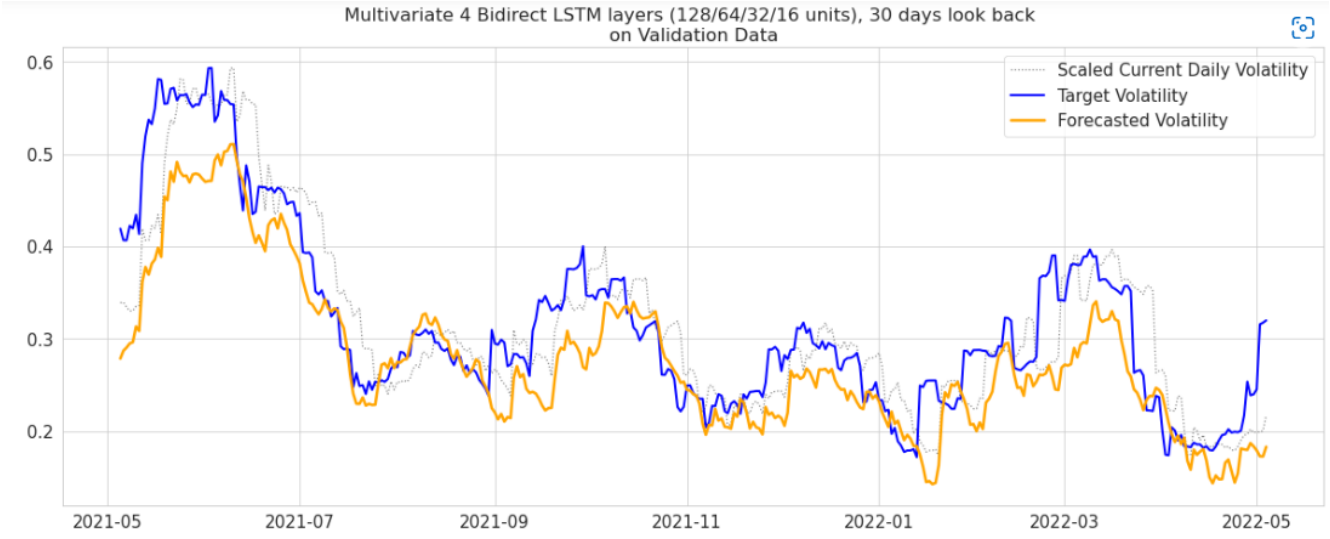


Fig 6.19 MultiVariate 4 Bidirect LSTM Layers

CHAPTER 7
EXPERIMENTATION

Following are the experimentations performed:

1. We took the Bitcoin data from Yahoo finance and since it was data from 2014 we have used min max scaler to scale down the entire data so that it becomes easier for training the models.

2. The main variable used for training till Univariate LSTM is Closing price, hence closing price plays an important role in forecasting volatility.

3. We have experimented the data on many models, including time series models, traditional models and deep learning models

4. We have experimented it on multivariate and univariate lstm as well.

5. The data split is

    Training-84%

    Validation-15%

    Testing -1%

6. For deep learning models we have experimented on Adam and sgd optimizer

7. We have tested the data for future 30 days

CHAPTER 8
RESULTS AND TESTING

We use the same architecture as multivariate LSTM with 2 layers , a lookback window of 30 days and a batch size of 64 days. We first create the dataset that combines training and validation set. Then a multivariate bidirectional LSTM is created and layers are added. The training is stopped if validation RMSPE is not improved by the end.In terms of performance on the validation set , final LSTM model has an RMSPE of 0.156677, which is roughly 4.42% better than the best performing variant of the GARCH models found - TARCH(1,2) with an RMSPE of 0.200954. TThe final LSTM model has an RMSPE of 0.0534 on the Test set (which is the most recent 30 days of which future volatility data is available for comparison). Since RMSPE indicates the average magnitude of the error in relation to the actual values, an RMSPE of 0.0534 would translate to a magnitude accuracy of 91.9%~ 92% on the average 7-day horizon daily volatility forecasting within the period .

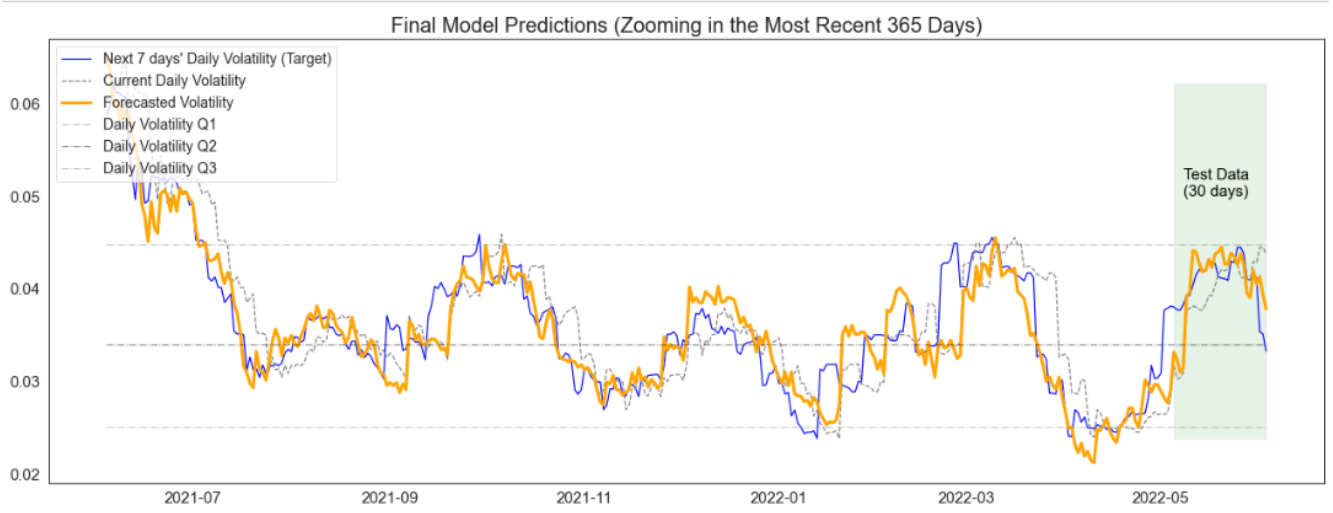


Fig 8.1 Final predictions on test data

```
In [207]: perf_df.to_pickle("performance_df.pkl")

In [208]: print('RMSPE on Test Set:', RMSPE(y_test, y_test_preds))
          RMSPE on Test Set: 0.08099981329090655

In [209]: 1-RMSPE(y_test, y_test_preds)
Out[209]: 0.9190001867090934
```

	Model	Validation RMSPE	Validation RMSE
10	1 Conv1D 2 Bidirect LSTM layers (32/16), n_past=30, batch=64	0.132709	0.045521
15	Multivariate 2 Bidirect LSTM layers (32/16 units), n_past=30, batch=32, tanh	0.135316	0.046871
9	2 layers Bidirect LSTM (32/16 units), n_past=30	0.135903	0.043851
8	LSTM 1 layer 20 units, n_past=14	0.143019	0.047186
13	Multivariate Bidirect LSTM 3 layers (64/32/16 units), n_past=30	0.144384	0.048810
7	Simple LR Fully Connected NN, n_past=14	0.148061	0.044936
12	Multivariate Bidirect LSTM 2 layers (32/16 units), n_past=30	0.155884	0.055082
1	Random Walk Naive Forecasting	0.156799	0.047765
14	Multivariate 4 Bidirect LSTM layers (128/64/32/16 units), n_past=30, batch=64	0.165676	0.058299
6	Bootstrap TARCH(1, 2, 0), Constant Mean, Skewt Dist	0.177444	0.058197
4	Bootstrap TARCH(1,1), Constant Mean, Skewt Dist	0.186190	0.064162
5	Simulation TARCH(1,1), Constant Mean, Skewt Dist	0.192764	0.067221
3	Analytical GJR-GARCH(1,1,1), Constant Mean, Skewt Dist	0.245227	0.081287
0	Mean Baseline	0.284708	0.101387
11	2 Bidirect LSTMs (32/16), n_past=30, batch=64, SGD lr=6.9e-05	0.332860	0.141609
2	GARCH(1,1), Constant Mean, Normal Dist	0.522454	0.170346

Fig 8.2 Accuracies

CHAPTER 9
CONCLUSION AND FUTURE SCOPE

In terms of performance on the validation set), my final LSTM model has an RMSPE of 0.156677, which is roughly 4.42% better than the best performing variant of the GARCH models found - TARCH(1,2) with an RMSPE of 0.800954. Traders does not need to make perfectly accurate forecast to have a positive expectation when participating in the markets, they just need to make a forecast that is both correct and more correct than the general consensus. With GARCH still being the most popular volatility forecasting model, Multivariate LSTM could potentially give investors an advantage in terms of higher forecasting accuracy.The final LSTM model has an RMSPE of 0.0834 on the Test set (which is the most recent 30 days of which future volatility data is available for comparison). Since RMSPE indicates the average magnitude of the error in relation to the actual values, an RMSPE of 0.0834 would translate to a magnitude accuracy of 91.9% on the average 7-day horizon daily volatility forecasting However, since financial time series data are constantly evolving, no model would be able to consistently forecast with high accuracy level forever. The average lifetime of a model is between 6 months to 5 years, and there's a phenomenon in quant trading that is called alpha decay, which is the loss in predictive power of an alpha model over time. In addition, according to Sinclair (2020), researchers have found that the publication of a new "edge" or anomaly in the markets lessens its returns by up to 58%.These models therefore require constant tweaking and tuning based on the most recent information available to make sure they stay up-to-date and learn to evolve with the markets.There's potential application of WaveNet in the forecasting of volatility, and to explore that option in the future.In addition, it's common knowledge that economic events could affect markets' dynamics. Since cryptocurrencies have certain nuances that are different from other stocks and commodities', incorporating the regular economic calendars' events might not be the most relevant. I am currently still doing more research on collecting significant events that could have driven Bitcoin movements, and would like to incorporate that in another Multivariate LSTM set of models in the future to hopefully improve predictive power even more.Eventually I want to experiment with higher frequencies (ie. intra-day), and also different bucketing intervals as well.

REFERENCES

1. Jacopo De Stefani, Olivier Caelen1 , Dalila Hattab2 , and Gianluca Bontempi Machine Learning Group, Departement d'Informatique, Universit´e Libre de Bruxelles, Boulevard du Triomphe CP212, 1050 Brussels, Belgium Worldline SA/NV R&D, Bruxelles, Belgium1 Equens Worldline R&D, Lille (Seclin), France2 {jacopo.de.stefani,gianluca.bontempi}@ulb.ac.be olivier.caelen@worldline.com dalila.hattab@equensworldline.com

2. Machine Learning for Realised Volatility Forecasting

2. Machine Learning for Realised Volatility ForecastinEghbal Rahimikia and Ser-Huang PoonThis revision: November 10, 2021

3. Christiansen, C., Schmeling, M., & Schrimpf, A. A comprehensive look at financial volatility prediction by economic variables, Journal of Applied Econometrics, Vol 27, no 6, pp 956-977, August 2012

4. Tamal Datta Chaudhuri Principal, Calcutta Business School, Diamond Harbour Road, Bishnupur – 743503, 24 Paraganas (South), West Bengal Indranil Ghosh Assistant Professor, Calcutta Business School, Diamond Harbour Road, Bishnupur – 743503, 24 Paraganas (South), West Bengal" Forecasting Volatility in Indian Stock Market using Artificial Neural Network with Multiple Inputs and Outputs" International Journal of Computer Applications (0975 – 8887) Volume 120 – No.8, June 2015

5. Sheikh Mohammad Idrees Department of Computer Science and Engineering, Jamia Hamdard, New Delhi, India. M. Afs. Parul Agarwal Department of Computer Science and Engineering, Jamia Hamdard, New Delhi, India
har Alam Department of Computer Science and Engineering, Jamia Hamdard, New Delhi, Ind

APPENDIX A

ARCH MODEL

In econometrics, the autoregressive conditional heteroskedasticity (ARCH) model is a statistical model for time series data that describes the variance of the current error term or innovation as a function of the actual sizes of the previous time periods' error terms; often the variance is related to the squares of the previous innovations. The ARCH model is appropriate when the error variance in a time series follows an autoregressive (AR) model; if an autoregressive moving average (ARMA) model is assumed for the error variance, the model is a generalized autoregressive conditional heteroskedasticity (GARCH) model. ARCH models are commonly employed in modeling financial time series that exhibit time-varying volatility and volatility clustering, i.e. periods of swings interspersed with periods of relative calm. ARCH-type models are sometimes considered to be in the family of stochastic volatility models, although this is strictly incorrect since at time *t* the volatility is completely pre-determined (deterministic) given previous values

APPENDIX B

GARCH MODEL

Generalized AutoRegressive Conditional Heteroskedasticity (GARCH) is a statistical model used in analyzing time-series data where the variance error is believed to be serially autocorrelated. GARCH models assume that the variance of the error term follows an autoregressive moving average process. Although GARCH models can be used in the analysis of a number of different types of financial data, such as macroeconomic data, financial institutions typically use them to estimate the volatility of returns for stocks, bonds, and market indices. They use the resulting information to help determine pricing and judge which assets will potentially provide higher returns, as well as to forecast the returns of current investments to help in their asset allocation, hedging, risk management, and portfolio optimization decisions.

GARCH models are used when the variance of the error term is not constant. That is, the error term is heteroskedastic. Heteroskedasticity describes the irregular pattern of variation of an error term, or variable, in a statistical model.

PUBLISHING PAPER DETAILS

Paper has been published in the **International Journal of Engineering Research and Technology** (IJERT) with ISSN:2278-0181.

Paper Title: VOLATILITY PREDICTION
Paper Id: IJERTV11IS050187
Date of Publishing: 25th May, 2022