**Big Data Analytics**

**Project Report On**

**Enhanced Stock Price Forecasting: Integrating ARIMA,**

**SARIMA, LSTM, and Ensemble Methods for**

**Reliable Predictions**

***by***

*Archana katta – 811298298 (akatta2@kent.edu)*

*Tirumalasetty Naga Sudha Pavani – 811294971 (ntirumal@kent.edu)*

*Jaladurgam Navya - 811285985 ([jnavya@kent.edu](mailto:jnavya@kent.edu))*

# 1. Introduction

Predicting stock prices is very important in the financial markets. Analysts, Investors, and businesses need good predictions to make smart financial decisions. However, the market's unpredictable and unstable nature adds complexity to this task. This project aims to create a forecasting model using time series analysis with Arima models, supplemented by ensemble methods. With the utilization of these sophisticated methods, our goal is to enhance the precision of forecasting stock prices, enabling individuals to make more informed choices in intricate financial landscapes.

A well-known approach in this field is the Autoregressive Integrated Moving Average (ARIMA) model, famous for its capability to capture patterns of time series like trends and seasonality. Driven by the necessity of making well-informed decisions, stakeholders from different industries employ ARIMA modeling for a variety of purposes. Investors use ARIMA predictions to enhance portfolios and effectively mitigate risk, while financial institutions utilize them for making strategic investment choices and protecting against market volatility. Moreover, ARIMA is crucial in algorithmic trading by facilitating quick trade execution according to predicted price movements, as well as assisting analysts in detecting market trends and anomalies for thorough market analysis.

# 2. Project Description

In this project we will utilize the Time Series analysis, specifically with ARIMA models, to predict future stock prices. ARIMA models are effective at identifying fundamental patterns and trends in past stock market data, serving as a strong basis for making predictions. Yet, to improve the precision and trustworthiness of our forecasts, we employ ensemble methods.

We are doing this by using lots of different ARIMA models trained on different parts of the data or with different setups. This helps us avoid making predictions that are too specific and makes our forecasts more accurate. Also, we're not just sticking to ARIMA – we're also mixing in other forecasting techniques like SARIMA, LSTM. Each method has its own strengths, so by combining them, we can understand more complicated patterns in the data and make our predictions even better.

**Data Preprocessing, model selection (Navya):**

Address missing data, outliers, and inconsistencies in the dataset as part of preprocessing. Format the data into a time series structure. Explore different Time Series models suitable for predicting stock prices, like ARIMA, SARIMA, LSTM, Prophet, etc. Select the most fitting models considering data traits, project needs, and insights from EDA.

**EDA (exploratory data analysis)- (Archana):**

EDA helps us see how the data is spread out, notice any trends, find any weird bits, and understand how different things are connected. We use graphs, charts, and numbers to look at the data in different ways. EDA gives us important clues that help us make decisions and build models.

Model Training:

First, we divide the data into two groups: one to teach the model and the other to see how well it's learning. Then, we use different methods like ARIMA, SARIMA, LSTM to train the model.

**Ensemble Building and Model Evaluation (Pavani):**

Mix the forecasts from each separate model using ensemble methods. Try out various ensemble strategies like averaging, weighted averaging, or stacking. Find the best mix of models and their weights for the ensemble. Check how well each model and the combined model perform using specific measures. We will Compare the results to find which forecasting method is the most accurate and trustworthy. Adjust the models and the combination method based on what's learned from the evaluation.

# 3. Background

Forecasting changes in stock prices is vital for professionals in finance and economics. Hence, researchers continually seek improved methods for stock price prediction. ARIMA models are commonly used for this purpose, analyzing price changes over time. This paper [1] details our development of a stock price prediction model utilizing ARIMA. Findings indicate that ARIMA performs well for short-term predictions and is comparable to other methods for predicting stock prices.

Financial markets are constantly shifting, making prediction challenging. Conventional forecasting techniques may struggle to handle the fluctuations in stock prices. However, Time Series analysis, which examines data changes over time, proves valuable for stock price prediction.

The ARIMA model plays a significant role in Time Series analysis, adept at identifying short-term fluctuations and long-term patterns in data. This makes it a preferred option for forecasting, particularly in finance. Despite its popularity, no model is flawless, especially in highly volatile and complex market conditions. To do this IDE is required and machine learning models. [https://ieeexplore.ieee.org/document/7046047 [1](https://ieeexplore.ieee.org/document/7046047%20%5b1)]

* **Software tools (GUI, IDE, existing library, …)**

**Python libraries used:**

1. **Pandas:** A robust Python library, Pandas is employed for cleaning, transforming, and analyzing data. We utilize it for importing the dataset into a Data Frame, managing missing values, and conducting different data tasks.
2. **NumPy:** It is an essential tool for performing numerical calculations in Python. It offers assistance for extensive, multi-dimensional arrays and matrices, as well as a set of mathematical functions to efficiently manipulate these arrays.
3. **Matplotlib:** It is a Python library, is utilized for generating static, animated, and interactive visualizations. We make use of the pyplot module to generate visual representations like histograms, bar charts, scatter plots, etc., for visualizing data distributions and relationships.
4. **Seaborn:** This library utilizes matplotlib as its foundation and offers a user-friendly platform for generating visually appealing statistical graphics. It makes it easier to generate intricate visualizations such as heatmaps, violin plots, and pair plots, which are valuable for exploratory data analysis and extracting information from the data.
5. **scikit-learn:** A machine learning library in Python, scikit-learn offers straightforward and effective tools for data mining and data analysis. We import modules like Label Encoder, StandardScaler, and train test split from scikit-learn for preprocessing and model selection. LabelEncoder is utilized to encode categorical variables, StandardScaler normalizes numerical features, and train\_test\_split divides data into training and testing sets.

* **Required hardware:**
* **Operating system:** windows 11/Mac OS
* **Related programming skills**
* Python Programming Language.
* Familiarity with above mentioned Libraries.
* Basic knowledge of Machine Learning Models.
* Basic Knowledge of statistics.

# 4. Problem Definition

Our main aim is to create a dependable way to predict future stock prices. To do this, we use Time Series analysis with ARIMA models as the core of our forecasting method. We improve our model's accuracy by combining multiple ARIMA models and adding in other forecasting techniques like SARIMA and LSTM.

By using the strengths of each method and trying out different approaches, we want to develop a prediction model that can handle the unpredictable nature of financial markets. This model will give valuable insights for investors, analysts, and businesses.

* A summary of general solutions in project:

The primary objective of this project is to study the historical stock prices of a specific company (Apple Inc.) from December 12, 1980, till 2020, by analyzing stock market data over a period. Various libraries such as Pandas, Matplotlib, Seaborn, and TensorFlow/Keras are employed for tasks involving data preprocessing, visualization, and modeling. The dataset comprises seven columns: Date, Open, High, Low, Close, Adj Close, and Volume. After the dataset is uploaded, the script checks for duplicates or missing values to ensure the data's integrity is preserved. In this specific dataset, there are no repeated dates or empty values, ensuring the data is clean for analysis. This manuscript sets the foundation for future analysis, including predicting upcoming stock prices using methods like LSTM neural networks or ARIMA models. In general, it offers a thorough foundation for examining and grasping the fluctuations of stock market information throughout time.

A screenshot of a computer

Description automatically generated

Fig: Dataset used in the project

Moreover, the below box plots visually represent the distribution and spread of values within each variable. From the box plots, we observe that the Open, High, Low, Close, and Adj Close variables have relatively similar distributions, with the majority of the data clustered around the median, indicating stable price ranges over time. However, there are some outliers present in each variable, particularly in the High and Low prices, suggesting occasional extreme price fluctuations. On the contrary, the Volume variable displays a distinctly varied distribution, featuring a wider range of values and a greater presence of outliers, suggesting substantial fluctuations in trading volume across different periods. These observations offer important data for comprehending the stock's volatility and trading activity, which can be used to enhance forecasting models for making informed decisions in financial markets.

A group of graphs with numbers

Description automatically generated with medium confidence

Further, the 'Date' column to datetime format and setting it as the index of the DataFrame, facilitating time-based analysis. Subsequently, we implemented the line chart depicting the opening stock prices over time illustrates the historical trend and variability in the stock's value. It depicts periods of increase and decrease, emphasizing overall trends or irregularities. The histogram shows how opening stock prices are distributed, revealing how often certain price ranges occur and if prices cluster or spread out. Furthermore, the volume chart illustrates the stock's trading activity throughout time, displaying times of increased or decreased market involvement. Together, these visuals offer a detailed look at the stock's past performance, trading trends, and market influences, helping to understand its actions and guide investment choices.

Following the visualization, we analyzed the opening stock prices exclusively by dropping other columns. This simplification enables a more concentrated analysis of the variable being studied. Moreover, the dataset is divided into two sets - one for training and one for testing - using the year as the criteria. Data up to 2020 is used for the training set, while data starting from 2021 is set aside for testing purposes. This partitioning enables model training and evaluation on distinct time periods, reflecting real-world scenarios more accurately.

Lastly, we performed autocorrelation plots with the train data to investigate the structure of the opening stock prices. Autocorrelation plots visualize the correlation between a variable and its lagged values, helping identify any significant temporal dependencies or patterns in the data. The autocorrelation for the opening stock prices appears to be positive for short lags, then decays to near zero by a lag of 2000. This means that the opening stock price has a positive correlation with its price at lags of up to 2000 days. In general, the detailed examination offered by the graph helps to comprehend the past performance of the stock, guiding choices for investment strategies and risk control.

A graph with a line

Description automatically generated

The auto correlation plot helps in selecting suitable time series forecasting models like ARIMA, SARIMA, and LSTM neural networks.

Significant autocorrelation in specific lagged time periods signals serial correlation in data for ARIMA and SARIMA models, allowing them to capture temporal patterns and dependencies. By combining data from previous observations and utilizing differencing to achieve stationarity, ARIMA and SARIMA models can predict upcoming opening stock prices using historical information.

However, LSTM neural networks possess significant capability in capturing intricate temporal relationships within sequential data such as time series. The autocorrelation plot gives information about the existence and intensity of distant connections in the data, which can be utilized by LSTM models for making precise forecasts. LSTM models are adept at capturing intricate nonlinear patterns and extended relationships, which makes them ideal for forecasting tasks that conventional statistical models may find challenging.

# 5. The Proposed Techniques

**Data preprocessing**:

Loading the dataset of Apple company, after loading the data checking the datatypes of the dataset it has float, object and int datatypes, then finding the missing values and duplicate values in the dataset and the result is there are no missing and duplicate values. Converting 'Date' column to datetime format and set it as index. After doing preprocessing the data looks like below.

A screenshot of a graph

Description automatically generated

**Exploratory Data Analysis**:

Plotted box plots of each column to find out the distribution of data. In each plot the rectangular box shows the interquartile range which contains the middle 50% of the data. The horizontal line inside the box represents the median value. The whiskers extend from the box to the minimum and maximum values. In the “open” plot, we can see that the median value is quite low near the bottom of the plot range. It is a positively skewed distribution. The "High" plot, the median value is also relatively low, but the box extends higher, suggesting a wider spread of data above the median. The distribution of the "Low" values shows a relatively wide range of values, with the median value falling towards the lower end of the distribution. The “close” plot appears to have slightly higher median value compared to the low plot, but both have similar distribution of shape and range. The “adj close” distribution appears positively skewed with the median value towards the lower end and some higher values extending upwards. The distribution of trading "Volume" of data points seems to be clustered around a few distinct values.

**Line chart of opening stock price over time**: The x-axis has date with years and y-axis has opening stock price. This graph shows that the opening stock price remained relatively low, around $25 or below for an extended period. However, later dates on the right side of the graph there is a sharp and significant rise.

A graph showing the price of a stock market

Description automatically generated

**Distribution of opening stock prices:** The x-axis represents the range of opening stock prices, while the y-axis shows the frequency. The histogram reveals that most trading days had an opening stock price around $1, as indicated by the tall blue bar on the far left.

A graph of a number of blue squares

Description automatically generated with medium confidence

**Distribution of volume traded**: The x-axis represents the years from 1980 to 2020 while y-axis shows the trading volume. The graph exhibits several significant spikes in trading volume at various points in time. This is caused due to the buying and selling.

A green line graph with numbers

Description automatically generated

Predicting the open price, to achieve this we dropped the columns like High, Low, Close, Adj Close, and Volume because they represent different aspects of the stock price that are not directly related to the open price.

**Splitting the dataset**: Dataset is splitting into two train and test sets. This is done based on the year in the Date column. In the training set all rows with a year less than or equal to 2020, in the test set, with a year’s greater than 2020.

**Utilizing Date as Indexing:**

Changed index as Date column, because in time-series data as it provides a convenient way to access and analyze data over different time periods.

**Autocorrelation plot**:

The autocorrelation plot visually represents the correlation coefficients between the 'Open' prices and their lagged values at different lags.

In x-axis the lag contains the date valuesrepresents the lag in the time series, while the y-axis represents the autocorrelation values. In this plot the autocorrelation starts with a high positive value close to 1 for small time gaps and decreases slowly as the time gap increases. At a lag of around 4000, the autocorrelation starts to become slightly negative showing a small decrease in correlation at that point. Afterwards, it fluctuates slightly around zero for larger lags, this shows that there might be some repeating patterns in the data.

**Augmented Dickey-Fuller (ADF) test:**

Using ADF to check the time series data is stationary or not. If the data is non- stationary, can be made stationary through differencing and fit the ARIMA model. Inside class 1, uses ADF method to get the p value of the ADF test on original data. If the value is less than 0.05 then the data is stationery and methods proceed to fit ARIMA model if not, the data is nonstationary. In this case the code takes the first-order difference of the data and performs the ADF test on the differenced data. If the differenced data is less than 0.05 the method proceeds to fit an ARIMA model. The ‘pmdarima’ library is used to automatically select the best ARIMA model parameters (p, d, q) based on the given time series data.

**Machine Learning Model Training:**

* **ARIMA (Autoregressive Integrated Moving Average) Model:** ARIMA is a statistical analysis model that uses time series data to forecast future trends. It combines autoregression (AR), integration (differencing to make the data stationary), and moving averages (MA) to predict and understand future values based on past data.
* **SARIMA (Seasonal ARIMA) Model:** SARIMA extends the ARIMA model by adding seasonal elements to address and forecast seasonal variations in time series data. It includes additional seasonal parameters that model the seasonal AR, differencing, and MA components appropriate for the periodicity of the data.
* **LSTM (Long Short-Term Memory) Model**: LSTM models are a type of recurrent neural network (RNN) suited for sequential data and are capable of learning long-term dependencies. They excel in applications where the context or state across the time series is important, such as speech recognition or, in financial contexts, stock price predictions where past information is crucial for future predictions.

A screenshot of a computer program

Description automatically generated

Fig: Result of the Augmented Dickey-Fuller test

According to the above result the value is 8 autoregressive terms, has been differenced twice, and does not include any moving average components.

**Pseudocode for ARIMA model**:

1. Import pandas, numpy, ARIMA from statsmodels.tsa.arima.model, and matplotlib.pyplot.

Input: order (p,d,q)

2. Define the order (p, d, q) for the ARIMA model

3. Create an ARIMA model instance:

Pass the 'Open' column of the train\_data and the specified order as inputs

4. Fit the ARIMA model using the opg (outer-product gradient) method for parameter estimation:

Use the fit() method with cov\_type="opg"

5. Generate predictions for the test\_data using the fitted ARIMA model and the forecast () method:

Set the number of steps for forecasting equal to the length of the test\_data

**Pseudocode for SARIMA Model:**

1. Import pandas, numpy, SARIMAX from statsmodels.tsa.statespace.sarimax, and matplotlib.pyplot.

2. Create SARIMAX model instance:

Initialize the model with the 'Open’ column of the train\_data and the specified order.

3. Fit the SARIMAX model:

Use the fit () method without any parameters.

4. Generate predictions for the test\_data:

Use the forecast () method with the number of steps equal to the length of the test\_data.

**Pseudocode for LSTM:**

Importing the necessary libraries

# combining train and test set for scaling

combine train\_data['Open'] with test\_data['Open'] to create combined\_data

# Scaled combined data using MinMaxScaler

scale combined\_data using MinMaxScaler to get scaled\_data

# set the no of time steps

set time\_steps to 1

# creating an empty list to train the data

create empty lists X\_train and y\_train

#preparing the training data

for i from 0 to len(train\_data) - time\_steps:

append scaled\_data[i:(i + time\_steps), 0] to X\_train

append scaled\_data[i+ time\_steps, 0] to y\_train

#reshape X\_train for the LSTM input shape

reshape X\_train to (X\_train.shape[0], 1, X\_train.shape[1])

#Building the LSTM model

create a Sequential model called model\_lstm

add LSTM layer with 50 units and input\_shape=(X\_train.shape[1], X\_train.shape[2])

add Dense layer with 1 unit

compile model\_lstm with optimizer=’adam’ and loss=’mean\_squared\_error’

# Train the lstm model

train model\_lstm with X\_train, y\_train for 50 epochs and batch\_size=1

# preparing the test data

create empty lists X\_test and y\_test

get last time\_steps values from scaled\_data and store in inputs

for I from 0 to len(inputs) – time\_steps:

append inputs[i:(i+ time\_steps), 0] to X\_test

append inputs[i + time\_steps, 0] to y\_test

# Reshaping X\_test for LSTM input shape

reshape X\_test to (X\_test.shape[0], 1, X\_test.shape[1])

predict using model\_lstm with X\_test and store the predictions in predictions\_lstm

# Inverse transform the predictions to get original scale

inverse transform predictions\_lstm using scaler

calculate mean squared error between test\_data[‘Open’] and predictions\_lstm and store in mse\_lstm

# 6. Visual Applications

* **GUI design**

Using the Tkinter library, developed a GUI application for forecasting and examining stock prices.

**Library Importation**: The code brings in essential libraries like Tkinter for GUI, pandas for data processing, numpy for mathematical tasks, matplotlib for visualizations, and additional libraries for specialized duties such as ARIMA modeling and LSTM models for forecasting time series data.

**Definition of Class:** In the code, a class named `YourClassName` is created with functions for conducting the Augmented Dickey-Fuller (ADF) test to assess the stationarity of time series data. It also includes a method `class1` that applies the ADF test to ascertain stationarity and starts an auto ARIMA model depending on the stationarity outcome.

**Data Handling Functions:** There are various functions available for managing data, including loading datasets from CSV files, dividing data into training and testing sets, as well as generating predictions and evaluating metrics for ARIMA, SARIMA, and LSTM models.

**Setting up the GUI:** Tkinter is used to create the main GUI window. The components consist of a header label, a dataset loading button, and a results display label.

**Event Handlers:** The function `on\_predict\_click` is executed upon the user's click on the "Load Dataset" button. The dataset is loaded, divided into training and testing sets, autocorrelation plots are created, the ADF test is conducted, and finally, predictions are made with ARIMA, SARIMA, and LSTM models. Finally, the GUI shows the outcomes.

**Showing outcomes:** The GUI presents the results of each model, showcasing metrics like Mean Squared Error (MSE), Root Mean Squared Error (RMSE), R-squared, and Adjusted R-squared.

**Graphs:** The first graph shown is the autocorrelation plot, used to analyze the autocorrelation pattern in the time series data. Nevertheless, the current version does not include plotting predictions for each model.

Mainloop: This function from the Tkinter library is invoked to begin the GUI application.

In short, this program offers an easy-to-use interface for importing stock price data sets, conducting analysis, and evaluating the predictive accuracy of various models such as ARIMA, SARIMA, and LSTM. It is an all-encompassing tool utilized for analyzing and predicting time series data specifically in the field of predicting stock prices.

A screen shot of a computer screen

Description automatically generated

A screen shot of a computer program

Description automatically generated

A screen shot of a computer program

Description automatically generated

A screen shot of a computer program

Description automatically generated

A screen shot of a computer program

Description automatically generated

A screen shot of a computer program

Description automatically generated

After loading the dataset into the GUI, the autocorrelation plot is displayed to visualize the autocorrelation structure of the time series data. A screenshot of a computer

Description automatically generated

Autocorrelation plot display in GUI

A graph with a line

Description automatically generated

Afterwards, the process of selecting the ARIMA model begins. This procedure includes testing various ARIMA parameter combinations to determine the most suitable model.

The results shown indicate the outcome of the automated ARIMA process, indicating the optimal ARIMA model for the dataset according to the Akaike Information Criterion (AIC). A smaller AIC value reflects a more optimal model fit.

In this instance, the most suitable ARIMA model is identified as ARIMA(8,2,0)(0,0,0)[0], showing the sequence of the autoregressive (AR) part, differencing (I) part, and moving average(MA) part.

A screenshot of a computer program

Description automatically generated

Following the display of ARIMA results, the code proceeds to fit a SARIMA model, choose the optimum one using AIC, make predictions, and assess performance using metrics such as Mean Squared Error (MSE), Root Mean Squared Error (RMSE), R-squared, and Adjusted R-squared. Next, an LSTM model is fitted, data is preprocessed, the model is trained, predictions are made on the test set, and evaluation metrics are displayed to finish the analysis.

A screenshot of a computer

Description automatically generated

# 7. Experimental Evaluation

**Experimental settings**

The experiment's dataset consists of historical Apple Inc. stock price data (AAPL). Daily entries of stock market parameters including Open, High, Low, Close, Adjusted Close, and Volume are included in the dataset. The experimental evaluation focused on comparing the performance of three models: ARIMA model, SARIMA model, LSTM model.

**ARIMA (AutoRegressive Integrated Moving Average) model:**

Imported necessary libraries such as pandas, numpy, statsmodels (ARIMA model), and matplotlib.pyplot for data manipulation, numerical operations, ARIMA modeling, and visualization, respectively.We fit an ARIMA model to the 'Ope­n' column data from the training data. The ARIMA model orde­r had the (p, d, q) values. We traine­d the model with the fit () function. It use­d the outer-product-of-gradient me­thod to get good numbers. After training, the­ model\_fit object had the ARIMA mode­l with the estimated parameters, this can be used for making predictions.

A computer screen shot of a program code

Description automatically generated

Fig: Code for ARIMA model

**SARIMAX (Seasonal AutoRegressive Integrated Moving Average with eXogenous factors):**

SARIMAX model to forecast the "Ope­n" column of the training data. First, we start the mode­l with the "Open" column data from the training datase­t and with the number for "order". The­ "order" has values for the SARIMAX mode­l things. Then, we use fit () to make­ the model work with the training data. Fit () finds the­ right numbers for the model base­d on the data. At last, predictions1 has the future­ values for the "Open" column. The­ length is the same as the­ test data.

A screen shot of a computer code

Description automatically generated

Fig: Code for SARIMA

**LSTM (Long Short-Term Memory):**

Importing the required libraries for the neural networks, scaling data, evaluating and model performance. Combining the Open column data form both taring and testing datasets, Scaling the combined data to ensure that all values are between 0 and 1 by using MinMAxScaler. We are setting the number of time steps to 1. This parameter defines how many previous time steps of data the model will use to make predictions. Preparing the training dataset by creating input-output pairs. Creating as LSTM network layer with 50 units and dense layer with one unit. Compiling the model using the Adem optimizer and mean squared error loss function. Then, we train the model using the training data for 50 epochs with a batch size of 1.

Preparing the test data is the same as training dataset. After that using the trained LSTM model to make predictions on the testing dataset. We Scaled the data before training, so we needed to inverse the scale the predictions to get them back to original data.

**A computer screen shot of a program code

Description automatically generated**

**A computer screen shot of text

Description automatically generated**

Fig: Code for LSTM

**Visual Predictions**

**ARIMA model:**

A graph showing a number of data

Description automatically generated with medium confidence

Fig: ARIMA predictions plot

A screenshot of a computer screen

Description automatically generated

Fig: Actual and the predicted values of ARIMA

In x-axis represents years and y-axis represents price. In the starting part of the plot the train data fluctuates around a low and stable level, while the predictions of ARIMA closely follow the actual data. By seeing the plot in the future, the ARIMA model predicted the open price of the stock market is going to increase gradually. When we compare the actual and the predicted values the result is below. The result shows that the predicted values have smaller differences than the actual values. For example, on the date 2021-01-04, the ARIMA model predicted a value of 135.092977, while the actual observed value (Open) was 133.520004.

**SARIMA model:**

A graph showing a number of data

Description automatically generated with medium confidence

Fig: SARIMAX predictions plot

The blue line is the train data. In the starting point the train data is low and stable for few years. But gradually it is increasing slowly and sharply at the end. The orange line shows the SARIMA predictions in the dataset. These are increasing rapidly. When we compare the actual and predicted values for the SARIMAX at the beginning the values are slightly higher than the actual values. But there are a lot of variances in some of the predicted and actual values. This discrepancy causes the gap between the predicted and actual values to widen over time.

A screen shot of a computer

Description automatically generated

Fig: Result of the actual and predicted values

**LSTM model:**

A graph with blue and orange lines

Description automatically generated

Fig: LSTM model

The x-axis represents the date and the y -axis represents the values. The blue line represents the test data, and the orange line represents the LSTM predictions. It is evident that the LSTM Predictions line follows the general trend and patterns of the Test Data line closely. The below represents the predictions for LSTM and test data. While comparing the predicted and test data values. Some values are close to the test data some or not.

A screenshot of a computer code

Description automatically generatedA screenshot of a computer

Description automatically generated

Fig: Predicted and actual data values

**Metrics Evaluation**

Mean Squared Error (MSE) and Root Mean Squared Error (RMSE): The average forecast error is measured using these parameters. The percentage of the dependent variable's variation that can be predicted from the independent variable (or variables) is measured by the R-squared and Adjusted R-squared statistics.

**ARIMA model:**

There are significant issues with the ARIMA model's performance indicators, which are utilized to forecast stock prices. With an average difference of 113.705 between the model's predicted values and the actual values, the Mean Squared Error (MSE) is 12,928.8073 and the Root Mean Squared Error (RMSE) is 113.705, indicating significant mistakes. More importantly, the model performs worse than a straightforward model that would predict the data's mean value, as seen by the severely negative R-squared value of -44.7062 and the Adjusted R-squared value of -44.8311. These measures make it abundantly evident that the ARIMA model does not accurately match this dataset and does not adequately represent the underlying patterns.

**A close up of numbers

Description automatically generated**

Fig: ARIMA model evaluation

**SARIMA model:**

The provided metrics demonstrate how poorly the ARIMA model, which is used to anticipate stock prices, performed. Large mistakes are shown by a Mean Squared Error (MSE) of 12,928.807, which indicates high average squared disparities between the actual and forecast stock values. The model's predictions are said to differ from actual values by an average of 113.705 units, according to the Root Mean Squared Error (RMSE) of 113.705. Depending on the circumstances surrounding the data, this deviation might be significant. Furthermore, the model performs much worse than a straightforward model that would just predict the mean of the observed data, as seen by the severely negative values of R-squared (-44.706) and Adjusted R-squared (-44.831). The model also fails to explain the variability of the data.

**A close up of numbers

Description automatically generated**

Fig: SARIMA model evaluation

**LSTM model:**

The model fits the data rather well, as evidenced by these performance metrics: With an average squared difference between the estimated values and the actual values that is quite small, the Mean Squared Error (MSE) of 24.6695 indicates a nearly perfect match. An indication of a reasonably good forecast accuracy is the Root Mean Squared Error (RMSE) of 4.9668, which indicates that the average prediction error is less than 5 units. Furthermore, the model's ability to use its predictors effectively without adding unnecessary complexity is demonstrated by the fact that both the R-squared and Adjusted R-squared values are above 0.91, meaning that more than 91% of the variability in the dependent variable is fully explained by the model.

**A number and text on a white background

Description automatically generated**

Fig: LSTM model evaluation

**Performance Report**

**Model Precision:**

* ARIMA/SARIMAX: The MSE and RMSE scores of these models show that they did rather well. They work especially well at capturing seasonal trends and linear correlations.
* LSTM: Depending on how its parameters are adjusted and the characteristics of the time series, the LSTM model may be able to surpass conventional models in capturing intricate patterns.

**Computational Efficiency:**

* ARIMA/SARIMAX: Because they need less computational complexity than LSTM, these models can often be trained more quickly. They work well on the CPU and don't require GPU acceleration.
* LSTM: Needs greater time and computing power during training, particularly when the network design gets more intricate. A GPU was used for training in order to speed up the procedure.

**Robustness and Scalability:**

* Traditional Models: Although quicker and using less resources, they might not be able to manage non-linear interactions to the same extent as LSTM.
* LSTM: Can scale effectively with increased data amount and complexity, albeit at the expense of higher processing load. It also has great robustness to non-linear patterns.

**Result**

The analysis of the three models—LSTM, SARIMAX, and Arima—using the stock price dataset of Apple Inc. (AAPL) shows varying degrees of model fit and predicting accuracy. With abnormally low R-squared values, the ARIMA model performed substantially worse than expected, suggesting that it was unable to adequately capture the intricacies of the dataset. By comparison, the performance of the LSTM and SARIMAX models was significantly better. With its low Mean Squared Error (MSE) and high R-squared values, the LSTM model in particular demonstrated its power in handling non-linear data patterns and suggested a strong capacity to forecast future stock prices with notable accuracy. Comparably, the SARIMAX model, which took into account external influences and seasonal fluctuations, also produced accurate predictions, but with marginally lower precision than the LSTM.

Overall, the machine learning approach—particularly the LSTM—proved to be quite effective, indicating that it is a suitable method for complicated financial time series forecasting problems, whereas traditional statistical approaches failed with this dataset.

**8. Future Work**

Future efforts can involve expanding the dataset and improving our techniques in order to further improve the precision and dependability of our forecasting models. A wider context affecting stock prices may be obtained by include other macroeconomic variables, such as inflation, interest rates, GDP growth rates, and other pertinent financial measures. Furthermore, moving averages, RSI (Relative Strength Index), and MACD (Moving Average Convergence Divergence) are examples of technical indicators used in advanced feature engineering that may be used to enhance forecasts and modify the input signals of the model.

Model hybridization experiments provide an additional improvement path. We may produce a more reliable forecasting tool by fusing the advantages of several models, such as merging LSTM's expertise with non-linear patterns with ARIMA's linear modeling skills. Long-range relationships within the data may be better learned with more research into deep learning architectures such as GRU (Gated Recurrent Units) or attention techniques.

**Possible Project Extensions**

Several project additions might increase the applicability and usability of our models beyond the existing framework. By applying the created models to stock markets in other locations, for example, geographic expansion might enable us to generalize the forecasting power across diverse economic and regulatory situations. Segmenting the stock market according to industries like technology, healthcare, and finance and developing specific models for each based on its own characteristics would be an interesting next step.

Another important advantage might come from incorporating sentiment analysis into our models. Through the use of natural language processing tools, we can analyze financial news, reports, and social media in order to assess market sentiment and its effect on stock prices, thus providing an additional level of predictive capability. Furthermore, the creation of risk assessment tools that not only predict prices but also offer perceptions into the possible volatility and hazards connected to certain equities may prove to be an invaluable asset for traders and financial experts.

Our project may evolve from a traditional forecasting tool into a full suite of financial analysis tools as a result of these upcoming developments and possible additions. Our goal is to improve our models' usability and give traders, analysts, and financial institutions with relevant information by expanding their breadth and technological complexity.

**Conclusion**

Using the AAPL stock price dataset, the experimental evaluation of the ARIMA, SARIMAX, and LSTM models yields informative results on the trade-offs between conventional statistical techniques and cutting-edge deep learning methods. For simpler predictions, ARIMA and SARIMAX are fast and effective; but, at the cost of greater processing demands, LSTM has a greater capacity to model intricate patterns. The particulars of the forecasting task, such as the availability of computer resources, the dataset's complexity, and the acceptable trade-off between accuracy and performance, would therefore determine which model would be used.

# 9. References

1. Bettiza, Martaleli & Pramesti, Anisa & Malik, Riani & Nurfalinda,. (2023). An Evaluation on the Wind Speed Forecasting Model Using the ARIMA and SARIMA Methods Based on the MSE Value. 241-245. 10.1109/ICOIACT59844.2023.10455792.

<https://www.researchgate.net/publication/378749760_An_Evaluation_on_the_Wind_Speed_Forecasting_Model_Using_the_ARIMA_and_SARIMA_Methods_Based_on_the_MSE_Value>

1. Nabipour, Mojtaba & Nayyeri, Pooyan & Jabani, H. & Mosavi, Amir & Salwana, Ely & Band, Shahab. (2020). Deep Learning for Stock Market Prediction. Entropy. 22. 840. 10.3390/e22080840.

<https://www.researchgate.net/publication/343339583_Deep_Learning_for_Stock_Market_Prediction>

1. Mokhtar, Kasypi & Mhd Ruslan, Siti Marsila & Bakar, Anuar & Jeevan, Jagan & Othman, Mohd. (2022). The Analysis of Container Terminal Throughput Using ARIMA and SARIMA. 10.1007/978-3-030-89988-2\_18.

<https://www.researchgate.net/publication/358614243_The_Analysis_of_Container_Terminal_Throughput_Using_ARIMA_and_SARIMA>

1. Siami Namini, Sima & Tavakoli, Neda & Siami Namin, Akbar. (2018). A Comparison of ARIMA and LSTM in Forecasting Time Series. 1394-1401. 10.1109/ICMLA.2018.00227.

<https://ieeexplore.ieee.org/document/8614252>

1. Sirisha, Uppala & Belavagi, Manjula & Attigeri, Girija. (2022). Profit Prediction Using ARIMA, SARIMA and LSTM Models in Time Series Forecasting: A Comparison. IEEE Access. PP. 1-1. 10.1109/ACCESS.2022.3224938.

<https://ieeexplore.ieee.org/document/9964190>

1. Kumar, Th & Das, Himanish & Choudhary, Upasana & Dutta, Prayakhi & Guha, Debarati & Laskar, Yeasmin. (2021). Analysis and Prediction of Air Pollution in Assam Using ARIMA/SARIMA and Machine Learning. 10.1007/978-981-16-1119-3\_28.

<https://ouci.dntb.gov.ua/en/works/45LMMMv7/>