DATA MINING PROJECT REPORT

Group members:
I21-1697 Laiba Khawar
I20-0847 Fatima Asim

Introduction:

This report delves into a thorough analysis of historical stock market data sourced from "SPX.csv". Spanning nearly a century, from December 30, 1927, to November 4, 2020, this dataset offers a rich source of information on the Standard & Poor's 500 Index, enabling us to uncover trends, calculate returns, and employ various analytical techniques to enhance our understanding of the market.

Data Overview and Preprocessing:

Upon loading the dataset into a Pandas DataFrame, the first step was to ensure data consistency and integrity. This involved converting the date column to datetime format and setting it as the index, facilitating time-based operations. A cursory examination revealed no missing values, affirming the dataset's completeness. Additionally, an initial visualization of trading volume over time provided a glimpse into market activity, setting the stage for further analysis.

Investment Returns Calculation:

Calculating investment returns is paramount for assessing profitability and making informed investment decisions. By computing the difference between closing and opening prices and normalizing by the opening price, we derived a return metric that offers insights into the performance of the stock market over time.

Visualizing Closing Prices:

Visualization is a powerful tool for trend analysis, and Plotly enabled us to visualize closing prices dynamically. Leveraging features like a range slider and selectors, we gained deeper insights into long-term trends, volatility, and potential patterns in the stock market, empowering us to make more informed investment decisions.

Technical Indicators:

Technical indicators play a crucial role in market analysis, offering insights into price trends, momentum, and potential buy or sell signals. Simple Moving Average (SMA) and Exponential Moving Average (EMA) provided valuable insights into the direction of price trends, while Relative Strength Index (RSI) and Moving Average Convergence Divergence (MACD) offered momentum and trend-following signals, enriching our understanding of market dynamics.

Time Series Decomposition:

Decomposing the time series into its constituent components—trend, seasonality, and remainder—unveiled underlying patterns and irregularities. By implementing decomposition techniques, we gained insights into long-term trends, seasonal fluctuations, and residual variations, enhancing our ability to interpret market behavior effectively.

Stationarity Test (ADF Test) and Differencing:

Ensuring the stationarity of the time series is essential for reliable analysis. The Augmented Dickey-Fuller (ADF) test revealed non-stationarity, prompting the application of differencing techniques to stabilize the series. By differencing the data, we transformed it into a stationary form, enabling robust time series analysis and facilitating more accurate forecasting.

Arima model:

The ARIMA model configuration utilized the auto_arima function to automatically select optimal parameters (p, d, q), streamlining setup and enhancing accuracy. Parameter selection was aided by ACF and PACF plots, ensuring efficient modeling of autocorrelation and partial autocorrelation. The dataset was partitioned into training (1987-2018) and validation (2019-2020) sets for model training on historical data and evaluation on unseen future data. Mean Squared Error (MSE) was employed for model evaluation, with lower MSE values indicating superior predictive performanceIn the training and testing process, the ARIMA models are trained on historical stock price data with different differencing techniques applied. Daily differencing focuses solely on removing trends from the data, which may overlook underlying seasonality patterns, leading to relatively higher MSE values during testing. However, seasonal differencing effectively captures both trend and seasonality components, resulting in lower MSE values and improved forecasting accuracy. By incorporating the seasonal lag, this technique ensures a more comprehensive understanding of the data's cyclic behavior, enhancing predictive performance. Conversely, the combination of daily and seasonal differencing, while theoretically promising, does not yield a substantial improvement in accuracy compared to seasonal differencing alone. This underscores the importance of considering seasonality in time series forecasting, particularly in financial markets where seasonal trends can significantly influence stock prices. Thus, the ARIMA model with seasonal differencing emerges as the optimal choice for this dataset, reflecting its ability to capture and leverage seasonality for more accurate predictions.

ANN model:

The provided code snippet outlines a comprehensive process for developing, training, and evaluating a neural network model for time series forecasting using Keras. Initially, the data is prepared by creating input-output pairs through the `create_sequences` function, ensuring the model's access to sequential data. Following this, data normalization is performed using MinMaxScaler to maintain consistency in feature scales. The model architecture, defined within the `create_model` function, comprises two dense layers with 100 neurons each, incorporating batch normalization and dropout layers to mitigate overfitting. The model is trained on the prepared data with early stopping enabled, optimizing the Adam optimizer and minimizing mean squared error loss. Upon completion of training, predictions are generated for the test data, facilitating a visual comparison between actual and predicted values through a plotted graph. Finally, the trained model is saved in HDF5 format for subsequent usage. This holistic approach encompasses data preprocessing, model creation, training, evaluation, and storage, offering a streamlined framework for building robust time series forecasting models.

Exponential Smoothing (ETS) Model:

The provided code segments illustrate the application of different time series forecasting techniques. In the first part, a Seasonal Autoregressive Integrated Moving Average with Exogenous Factors (SARIMAX) model is fitted to weekly resampled data of stock prices. The model configuration includes both non-seasonal and seasonal orders, adjusted for weekly seasonality. Subsequently, forecasts are generated for the validation period, and confidence intervals are computed. These results are visualized alongside the actual validation data, offering insights into the model's predictive performance. Additionally, the fitted SARIMAX model is saved for future use.Moving on, the second part focuses on Exponential Smoothing methods, namely Simple Exponential Smoothing (SES), Double Exponential Smoothing (DES), and Triple Exponential Smoothing (TES). Each model is trained on the historical data, and forecasts are generated for the validation period. The Akaike Information Criterion (AIC) and Bayesian Information Criterion (BIC) values are computed for model comparison, providing insights into their relative goodness-of-fit. Finally, the forecasts are plotted against the actual validation data, offering a visual comparison of their performance. Overall, these techniques provide valuable tools for time series forecasting, each with its strengths and suitability for different types of data and forecasting scenarios.

Prophet Model:

The provided code employs Facebook's Prophet library for time series forecasting. Initially, the data is prepared by renaming the index to 'ds' (date) and renaming the 'Close' column to 'y', aligning with Prophet's requirements. The Prophet model is then initialized with specified seasonality modes, including additive seasonal components for yearly, weekly, and daily seasonality. Subsequently, the model is fitted to the prepared data, enabling it to learn from historical patterns and trends. A future DataFrame is created to generate forecasts for the validation period, with the frequency set to daily ('D'). The forecasts, along with their associated confidence intervals, are plotted against the historical data using Plotly, providing a visual representation of the model's predictions. This process allows for an intuitive understanding of how well the Prophet model captures the underlying patterns in the data and its ability to forecast future values accurately.

Support Vector Regression (SVR) Model:

In this code snippet, a Support Vector Regression (SVR) model is implemented for time series forecasting. Initially, the data is prepared by shifting the 'Close' prices by one time step to create lagged features, which are then used as input features ('X_train') for training the model. The SVR model is trained using a pipeline, incorporating StandardScaler for feature scaling and SVR with a radial basis function (RBF) kernel. The trained model is then used to predict future 'Close' prices for the validation period ('X_test'). The forecasted values are plotted alongside the actual validation data, enabling visual comparison of the model's performance. This approach leverages the flexibility and non-linearity of SVR to capture complex relationships in the time series data, offering an alternative method for time series forecasting.

Hybrid Model:

In this hybrid model, both ARIMA (AutoRegressive Integrated Moving Average) and ANN (Artificial Neural Network) techniques are combined to enhance forecasting accuracy. Firstly, an ARIMA model is fitted to the differenced data, which helps capture the linear relationships and trends in the time series. The ARIMA model forecasts future values, providing an initial prediction. Subsequently, an ANN model is trained to learn and correct the residuals or errors between the ARIMA forecast and the actual values. The residuals are reshaped and scaled as input features for the ANN. The ANN, with its ability to capture non-linear patterns and complex relationships, then predicts the corrections needed to refine the ARIMA forecast. These corrections are added to the ARIMA forecast to generate the final forecast, which is a combination of the linear and non-linear components captured by both models. This approach leverages the strengths of both ARIMA and ANN, resulting in a more accurate and robust forecasting model capable of capturing both linear and non-linear patterns in the time series data. The visual representation of the actual data, ARIMA forecast, and corrected forecast illustrates the improvement achieved through the hybrid ARIMA-ANN model, providing valuable insights into its predictive performance.

LSTM:

In this LSTM (Long Short-Term Memory) model, the training data is reshaped and scaled using MinMaxScaler to normalize the values. The LSTM architecture is then constructed, comprising two LSTM layers with 128 and 64 neurons respectively, followed by two dense layers with 25 and 1 neuron(s) respectively. The model is compiled using the Adam optimizer and mean squared error loss function. It is trained on the training data for 25 epochs with a batch size of 8. After training, the model is saved. For prediction, the test data is reshaped and scaled using the same MinMaxScaler, then fed into the trained model to generate forecasts. These forecasts are subsequently inverse-transformed to their original scale using the scaler. The Mean Squared Error (MSE) between the actual and predicted values is computed as 673.863, indicating the accuracy of the LSTM model. Visualized using Plotly, the actual closing prices and LSTM forecasts are plotted against time, providing a clear comparison of the model's performance in predicting future stock prices.

Conclusion:

In conclusion, this analysis offers a comprehensive exploration of historical stock market data, providing valuable insights into trends, returns, and market dynamics. Through meticulous preprocessing, visualization, and application of analytical techniques, stakeholders are equipped with actionable insights for investment decision-making and strategic planning, laying the groundwork for future research and analysis in the dynamic world of finance. In summary, each model offers distinct advantages for time series forecasting. The ARIMA model, leveraging auto_arima, streamlines parameter selection, and MSE evaluation, ensuring accuracy. ANN, with its deep learning architecture, captures complex patterns effectively. ETS models capture various time series components, Prophet excels in handling seasonal patterns, while SVR offers robust regression capabilities. Each model caters to specific forecasting needs, providing a diverse toolkit for accurate predictions.

Github link:

https://github.com/fatima344/Comprehensive-Forecasting-System-with-User-Interface-for-Multiple-Sectors/blob/main/README.md