Data Mining Project Report

Name: Hamiz Siddiqui, M. Shamil Umar

Roll Number: i21-1678, i21-1786

Section: M

Contents

Link to our GitHub Repository: Link

Overview

For this project we selected the monthly stock price from S&P 500 Index dataset, in which we selected the data past 1989, bringing the range from 1st January 1990 until 1st May 2024. The data behind this date was extremely different in terms of the trend to our current data and would only affect the predictions negatively as it would have made the models more biased to the lower values. With the data we can more accurately our current predictions. We combined all the technologies and tools effectively in a blend to create an interactive web-app to check the models and their predictions

Data Preprocessing and Preparation

As mentioned above, we selected the dataset to be from a certain updated range to be the most accurate for our model, as well as have a more effective training turnaround time. The data was converted from daily into monthly by taking a mean for all the month. Then for some models we applied some steps and for others they were not needed. For example, LSTM needed the MinMaxScaler but most other models did not need it. Similarly, the ANN and LSTM needed the last three Closing Prices in order to create a prediction for the current date, so the data was moved according to that desired shape. The data had no need for cleaning or standardization except the one mentioned. After the preprocessing was complete, we moved on to training the different models on them

Model Development

ARIMA

For the ARIMA model we started on checking for the stationarity of the data which revealed that the data in its pure form was not stationary. Using the pmdarima library the data was differentiated and made stationary on the first difference. Using the ACF and PACF plots, as well as some manual checking we found the order of ARIMA to be equal to 5, 1, 0. Then for training the model, a technique called slicing was used in which we train the data on the train set and then for the testing set we get a prediction and then add the actual value into the training data and removing the previous first value out. The predictions are being made in the exact in the same way. This helps to make the model the most accurate it can be as it is trained on the newer values rather than the older data and making a blind guess for the newest values.

SARIMA

The SARIMA model is trained in a similar way to the ARIMA model, using the extra seasonal parameter in the seasonal order parameter to optimize the predictions that we got from it. Again using the same slicing technique, we gained the trained model and its predictions. The newer predictions in SARIMA are better than ARIMA

ETS

Exponential Time Smoothing was a model that we were using for the very first time and it was a trickier one to train and develop. Given the fact the basis of the model is smoothing the curve out to become into a more regression like line, it was not very effective for a very volatile data like that of the stocks. Playing around with the values a lot gave us the final model we did receive which was somewhat accurate in its predictions. By training the model on the trend and dates, smoothing out the seasonality very little and trend about the same we managed to get the line that is cutting through the data in a way to minimize the errors we were measuring the models against

Prophet

Another model we were using for the first time it was a much more straightforward training than the other models in comparison. There was a lot of parameters to play around with and doing so helped us get a model that is somewhat accurate in its trend however the nature of the predictive function of the Prophet model meant that the model was still highly inaccurate for the more volatile data we got in the latter half of the data. The lines drawn in the Prophet prediction are almost stationary and cyclic than the highly extremely rough stock price lines. Overall it was a smooth, fast and simpler model in terms of training

SVR

A model that is built for simple predictions, SVR gave us a lot of trouble in terms of training the model. The base SVR model inside the Sci-Kit Learn library is terrible at being scaled for the larger datasets, which is exactly what we had. The base SVR class would take too long to train and we quickly adopted the LinearSVR model, a more scalable version of the LinearSVR that amongst other things avoids using the kernel trick to save computational cost. After tuning the hyper-parameters of the model manually, we trained and measured the model. Using the same slicing technique as mentioned before, we managed to make the model as accurate as it possible could be

ANN

Using the PyTorch library we designed a neural network that was built on the Linear transformer to convert the data into an effective output for us. Taking in the previous three closes as mentioned already, the data was transformed to accommodate for the shifted values. This style of training helps us to understand the actual trend on a quarterly basis rather than blindly shuffling on the dates like done in Prophet and ETS. After preparing the data and using the MSE to optimize it the model was trained to be highly effective in its predictions, using the same slicing technique to get the correct results for our dataset and beyond it

ANN-ARIMA Hybrid

Merging the power of both these technologies was something that was tricky, however it was something that we managed to do seamlessly. By tinkering the values of the ARIMA predictions using the ANN model we built, the model was a much more effective predictor than all the other models. It leverages the ability to understand the trend from ARIMA and the computational power of the Neural Network to be highly effective in fixing the prediction to the most accurate value which is reflected in its scores

Model Evaluation

Model	R2 Score	MSE
ANN	0.954	31771.44
ARIMA	0.957	31304.74
SARIMA	0.961	28772.81
SVR	0.934	45977.855
ETS	-0.883	796342.75
Prophet	0.314	501481.19
LSTM	-0.671	1165582.55
Hybrid	0.985	10192.53

Front End

For the user interface we decided to go with a simple design that would be easy for anyone to pick up and check as there always is a need for simple interfaces. We show a simple drop down menu for you to select a model to check the model performance on. Once selected we get a line chart on the screen which compares the model predicting the training data and the actual training data itself. This allows us to easily view the performance of the model on the dataset we have. Along that we have the metrics of the models that we got while training them. There is a navbar at the top that allows us to switch to the comparison page which adds another menu to select another model to compare results and metrics on with. Chart.js was used to display the charts to the user while the front end is constructed completely using React and JSX

Back End

The Flask server is more akin to an API in this case rather than a back end server. Since we predicted all the closing prices for the next three years we only need to fetch and display the values rather than calculate the predictions on runtime like usually done in the Flask server setting. From the database we get the values from the table depending on the model that the user has selected and then providing the data to the React app to be displayed

Conclusion

Users of the app can compare and view the results of the combination of the many choices of highly accurate, some less, models that were trained to the highest degree. Each model had its accompanying challenges and problems that we handled and overcame as a team. The Hybrid model is incredibly accurate and useful for predictions, and while the LSTM is theoretically the best model for this, it is very prone to overfitting which it has done so in this case. In conclusion, each model provides its pros and cons and must be considered for predictions or any sort of use but it is clear that using a mix like in Hybrid is the most effective for this case