Task 04 : Model Selection Document

For training we use several approaches

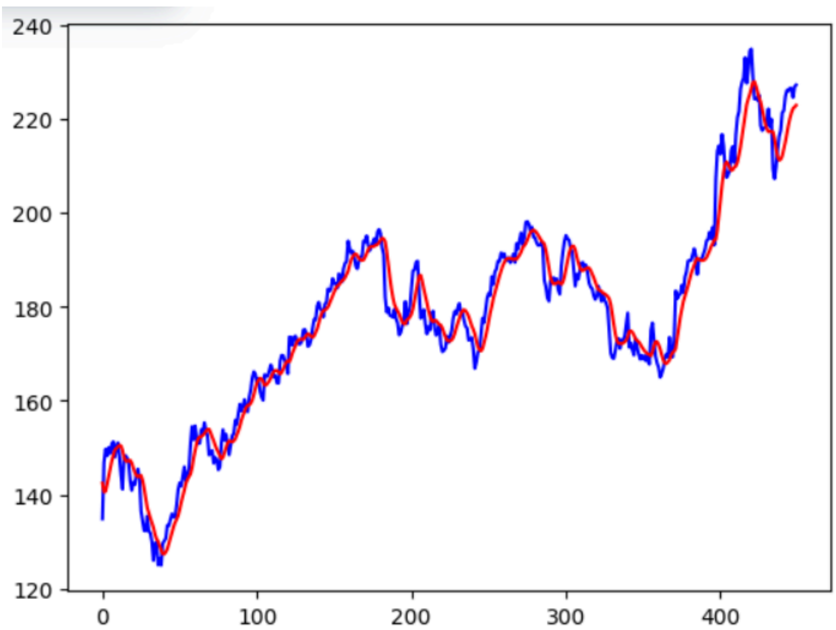
- Training Vanilla Versions of Popular forecasting models
- Training forecasting models by incorporating Engineered features and technical indicators
- Using a CNN model to capture complex time variant dependencies and using a LSTM layer for forecasting

Training Vanilla Versions of Popular forecasting models

We did this approach so we can have a baseline idea about what we can expect.

- We tried models like ARIMA, Prophet LSTM which are the popular choices
- Best results were achieved from LSTM model

But There was a Problem



This is how our predictions looked but this looked **too good to be true.**

Then what we realized was that LTSM model basically has trained to output a lagged version of it's own training data which was not what we wanted

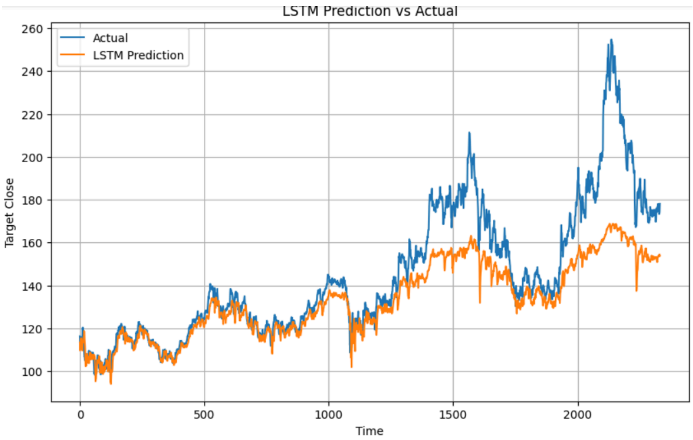
In fact many models may end up predicting a lagged version of stock prices, relying heavily on previous values

So we decided to not go with vanilla versions of these models

Training forecasting models with Engineered features and technical indicators

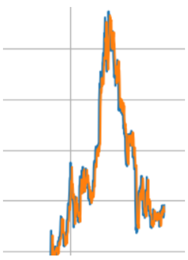
Since we can derive some statistical and technical indicators from existing time series we tried incorporating that with our existing models.

But now our previous results with LSTM actually degraded



We also tried models like Random forest but results were pretty bad

Some auto regressive models actually performed much better but the results were sort of skeptical for us.

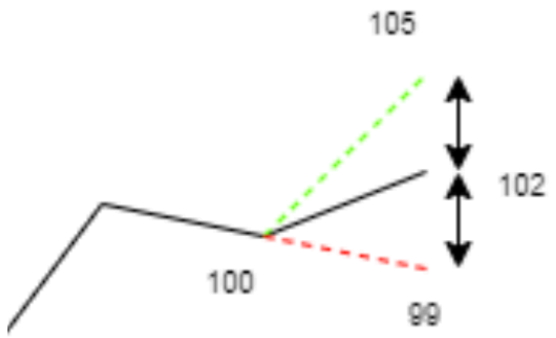


We've now hit a blocker.

Since our previous approaches were not successful we had to think outside the box.
First we tried to realise why stock market forecasting is not easy

Common challenges

1. Error Function Issue



Metrics like RMSE (Root Mean Squared Error), don't account for directional errors.

- RMSE gives the same penalty to both predictions, even though one model would correctly capture the upward trend.

Solution
- Use direction-aware metrics (e.g; Directional Accuracy) along with RMSE to better evaluate performance.

---

2. Scaling Issue

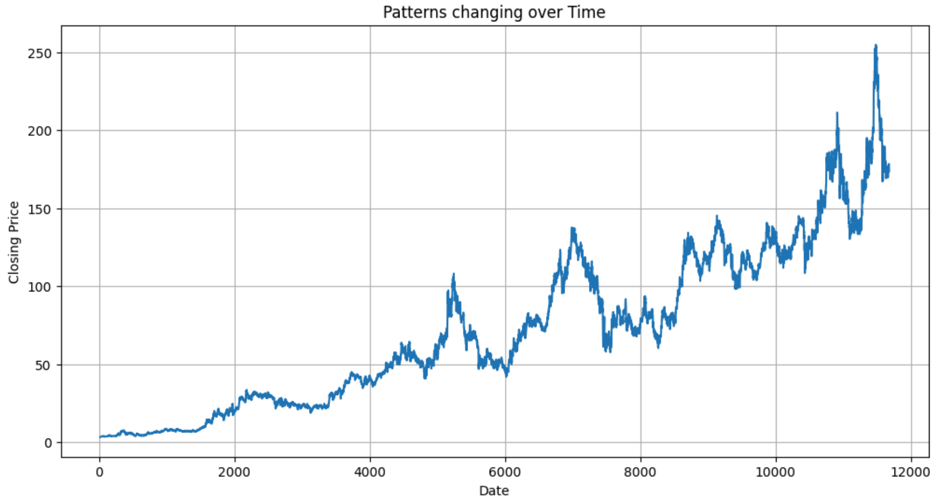
In stock markets, we don't know the maximum or minimum value in advance.
- This makes traditional scaling methods (like Min-Max scaling) less effective.

Solution

- Use a combination of standardization (z-scores) and min-max scaling.

---

3. Patterns changing with time issue



Even though we have 20-30 years of historical data, **older patterns may not apply to the current market conditions.**

- Data-hungry models (like deep learning models) require more training data, but we only have limited usable data.

---

4. Stock Prices being influenced by multiple factors

- Macroeconomic indicators
- Company-specific events
- News
- Market sentiment
- Geopolitical events.

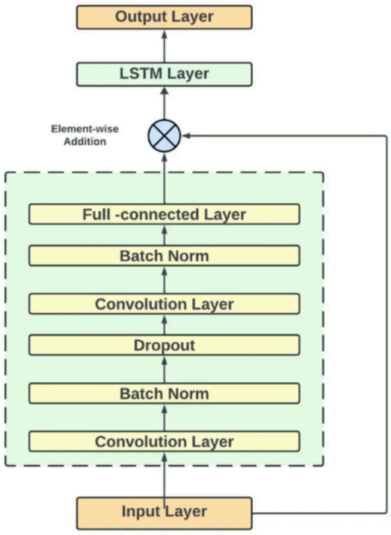
Solution

Use external data sources (like news sentiment or macroeconomic variables) to enrich input features.

- Implement multi-stock models that capture relationships between correlated stocks.

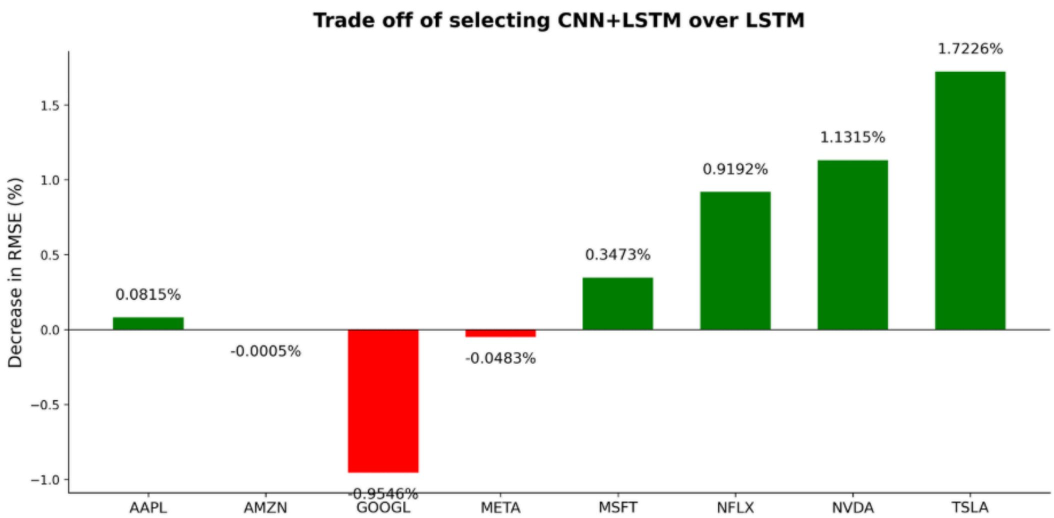
Using a CNN model with LSTM

We used an unconventional approach of combining a RESNET model with an LSTM layer.

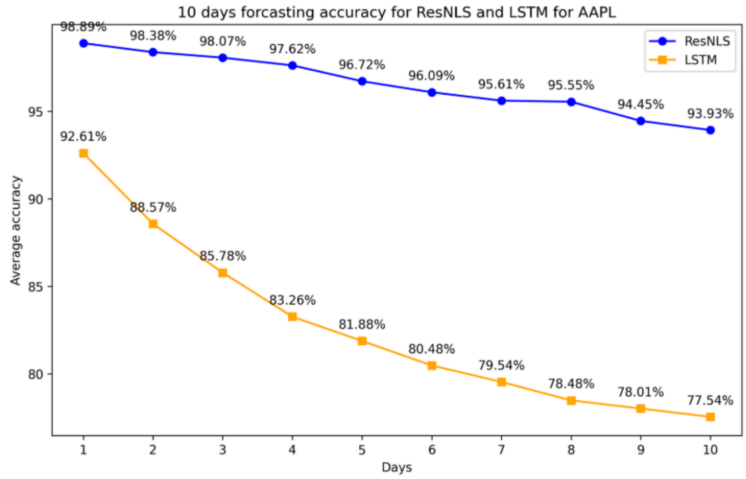


- Use CNN to extract features from historical stock prices.
- Apply pointwise transformation to CNN features.
- Feed transformed features into LSTM to capture temporal dependencies.

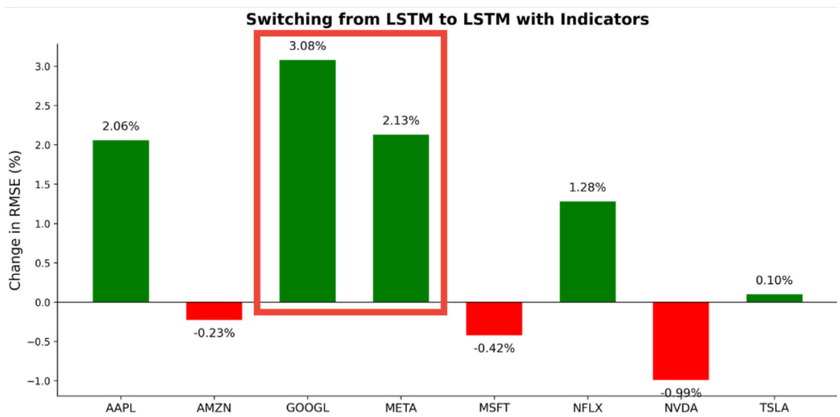
We tried implementing models for publically available stock datasets with the above model to get a clear idea about whether it is better or not.



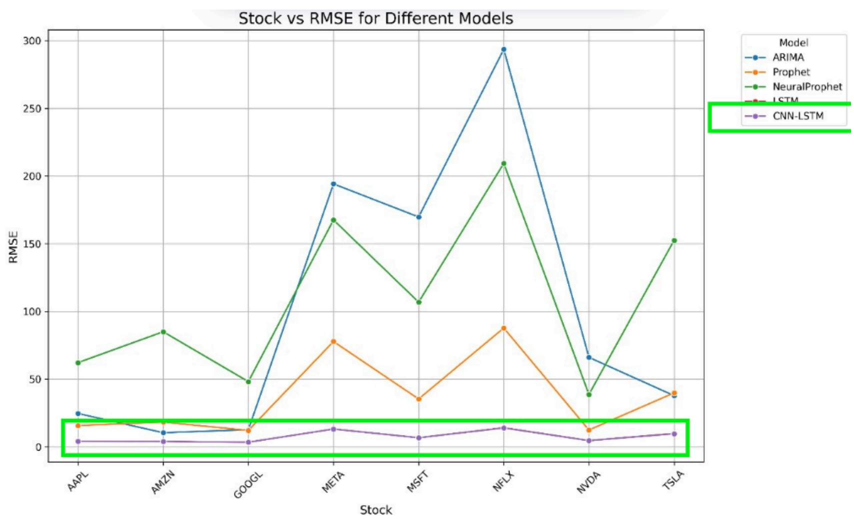
From those tests we were able to conclude that This model actually performs better than vanilla LSTM



Also we were able to improve the accuracy further by incorporating some technical indicators with above model

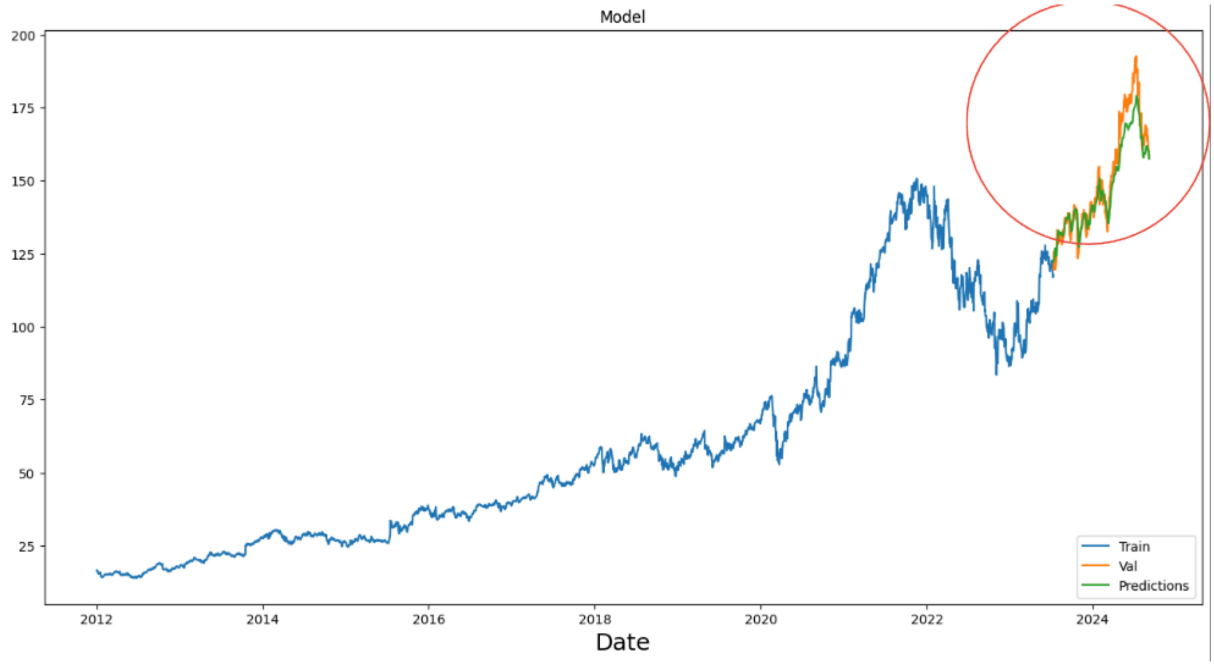


Then we compared this model with all the other models we've already tested



Improvements

In the CNN + LSTM approach we used even when the model identifies the trend, it finds it difficult to capture the trend accurately.



Solution

Add XGboost (Stacking) to capture the level

Before using XGB	After using XGB