Name: Sergey Kurbakov

Project: ML Forex Price Prediction

Introduction.

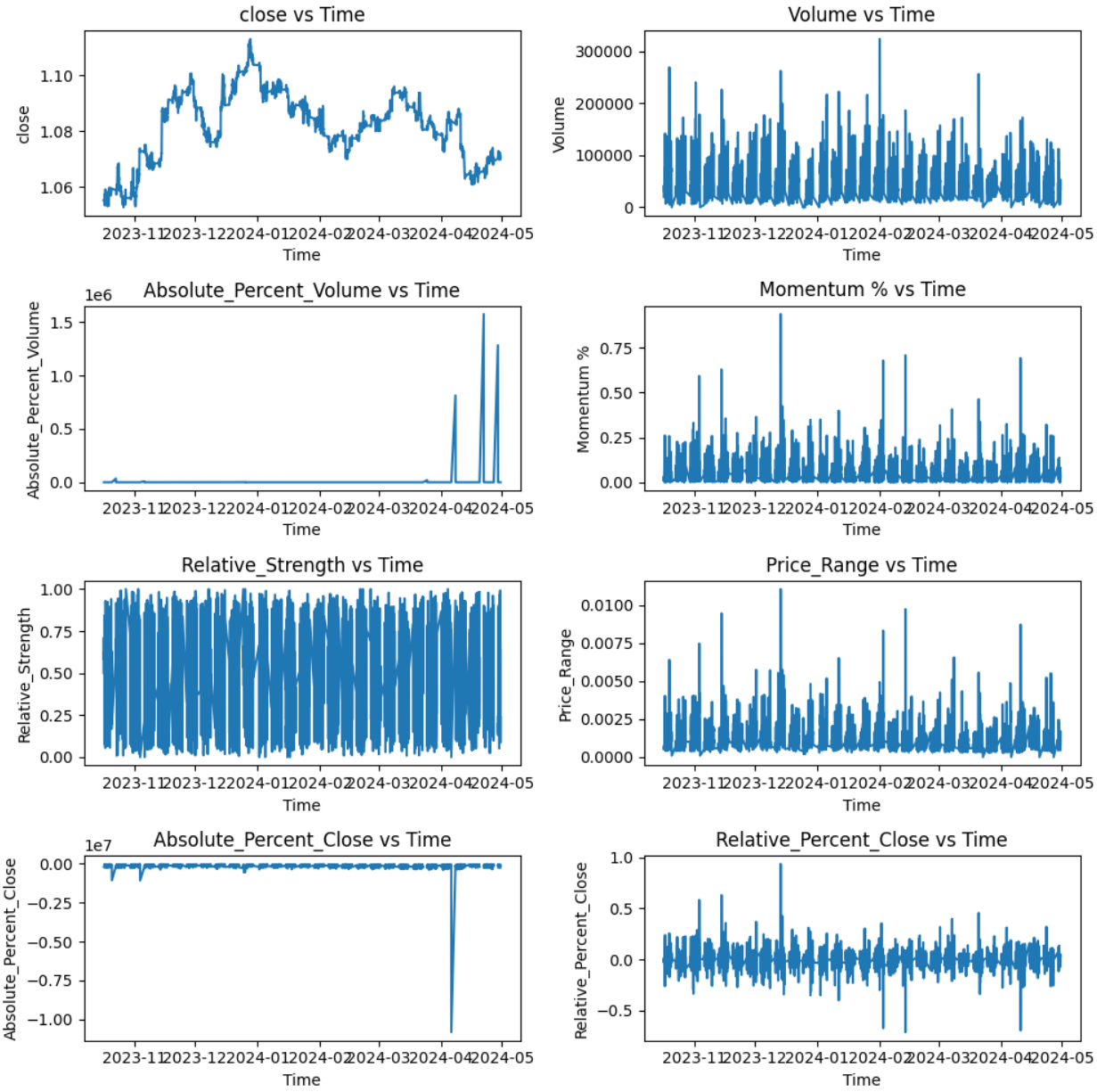
The goal of this project was to predict the forex exchange rate of the EUR/USD forex pair one timestep ahead on an hourly scale. The prediction was to be made based on a 24-hour lookback period. Once the prediction was made, the 24-hour observation window was shifted 12 hours forward in time and then another prediction was made. This was repeated until the end date of the time series. All of this was done on clean, standardized features that had undergone multiple feature selection processes. Selection methods that were used to select the top 13 training features included importance scores of Random Forest, Correlation Matrix, ANOVA F-test and Mutual Information Scores. Then the selected columns with the target column were taken from the data frame and restructured into 2 n-dimensional arrays that could serve as inputs into CNN, LSTM and N-Beats models. The first array contained windows of 24-hour observations of 14 features per timestep (hourly scale, so per hour), which included the target column, since future target movements can depend on it's previous moves in time due temporal dependence of timeseries. In our case, the target was the 1 step change in close price, so it is reasonable to assume dependence between current and previous closing price moves. The second array had the one step ahead targets based on the 24 windows in the first array. After this transformation, 3 models were trained and tuned on these 2 arrays. A simple CNN with one layer of convolutional layer and one layer of dense network on top. Two LSTM models; the first one had only layer and was tuned manually, the other one was tuned using Random Search and Bayesian Optimization Search (both from kerastuner), so that the better hyperparameters from Bayesian Optimization Search determined the model architecture. The final model, called N-Beats model (Neural Basis Expansion Analysis for Interpretable Time Series forecasting), which is a deep-learning network specifically designed for time-series forecasting, was trained and tuned manually. Grid Search (form kerastuner) hyperparameter tuning method was attempted on it. However major issues were encountered as the library from which the N-beats model comes does not seem to be compatible with the TensorFlow library. More specifically, it refuses to return a tensor object which is essential for us to perform Grid Search. Grid Search requires to receive a tensor object as input since it comes from the TensorFlow library. Finally, a new data set was imported. It was cleaned and preprocessed in the same way as the first one. The pretrained and tuned models were further trained on it and their predictions tested with the mean squared error serving as the error measurement. Their prediction losses and cumulative testing errors were then visualized. The results indicate that all of them have very close prediction error values, but the N-Beats has the best validation loss, followed by the simple CNN model, whereas the one-layer LSTM model seems to overfit despite having the lowest MSE value. Thus, the editor's pick is the N-Beats model (shows the most stability and potential).

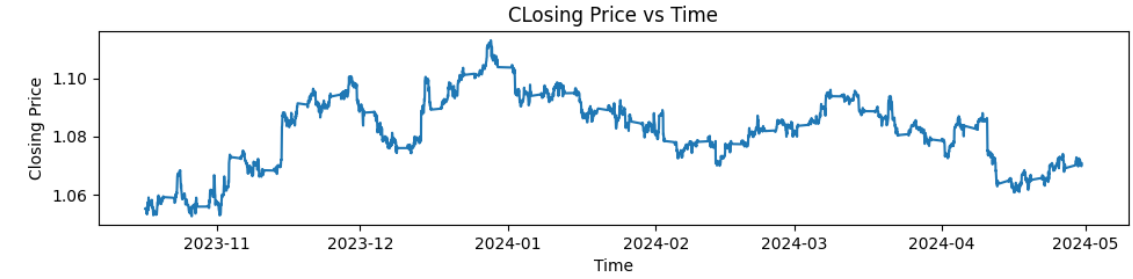
Data.

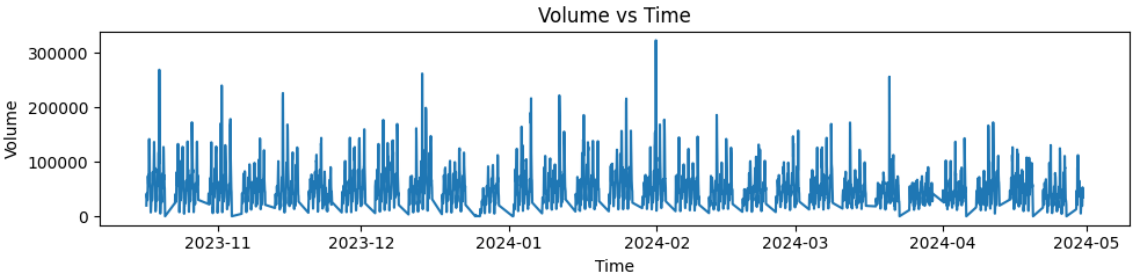
Two time series data tables (both on hourly timeframes) were exported from a trading platform called “TradingView”. The platform “TradingView” in turn aggregates financial data, including forex, from data providers, exchanges and data vendors like NASDAQ, ICE Data Services, etc. It uses API’s to collect raw data on price, volume, bid/ask quotes and other market indicators, cleans it and structures it for the trader. The initial data timeseries was used for training of three different models (CNN, LSTM and N-Beats), their hyperparameter tuning and establishing their respective validation losses. About 70% of the data was used for training and the rest 30% were used for validation across all three models. The start date of the initial data table was 16th of October 2023 and the end data was 30th of April 2024. The second timeseries data table, which is bigger in size than the first one, had a start date of 25th of August 2023 and end date of 20th May 2024. It was used to further train the pretrained tuned models and test their prediction accuracy. The portion used for additional training made up 60% percent and the rest were used for testing. In both instances, validation and testing, the mean squared error was used as a measure of predictive accuracy for all three models.

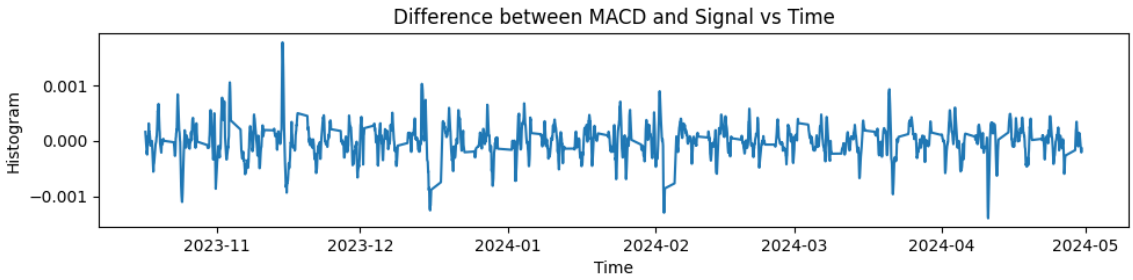
The features of both tables contained information on price dynamics in the form indicators as well as price stamps and trading volume. Some feature engineering was performed to add to the set of training features to better reflect relative price and volume movement (relative to previous activity – lag of 1). The target feature was achieved through feature engineering by differencing the close price data with a lag of 1 hour. Thus, the target was called “Close change”. The training features that made it past the feature selection process included, but not limited to, Volume change, Price Move, Price Range, High change, Low change, Volume, Absolute Percent Close and Relative Percent Close. Some famous technical indicators were also included such as the Histogram (MACD -Signal), Stochastic (%K-%D) and RSI. The benefit of those indicators is that they incorporate lagged data from previous observations in time by aggregating it in some way (usually using Moving Averages). This provides useful information and patterns for prediction of future exchange rate movements due to temporal dependence of the time series on past events. Hence, after data cleaning and engineering the goal of the models was to predict the change in the closing exchange rate between EUR and USD based on the previously mentioned training features over the past 24 hours. Once the prediction was made the rolling observation window was moved 12 hours ahead and a new prediction was made based on partly new observations. We say partly, since the observation window is 24 hours long and the movement ahead is only 12 hours, meaning consecutive 24-hour windows overlap by 12 hours. This was done partly to ensure enough training examples for the models, since without any overlap there would be fewer observation windows. Based on the following 2 figures we can see that features that capture relative price and volume movement behave roughly stationary, where the mean and variance stay constant over time. Therefore, we can use the mean squared error as our accuracy measurement. There are very few outliers, which is to be expected. Those outliers are not mistakes but were likely caused by some major economic or political announcements that caused panic selling or buying. Examples of such include “Price Range vs Time”, “Relative Percent Close vs Time”, “Absolute Percent Close vs Time” and “Volume vs Time”. Some indicators, that

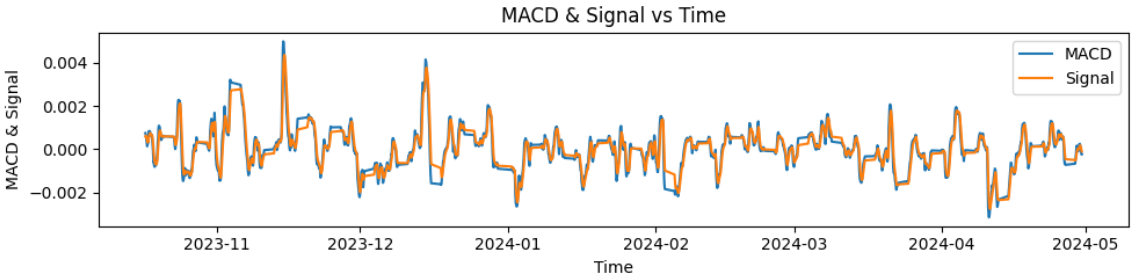
passed the feature selection process also exhibit stationarity, such as “Difference between MACD and Signal vs Time” and “RSI vs Time”.

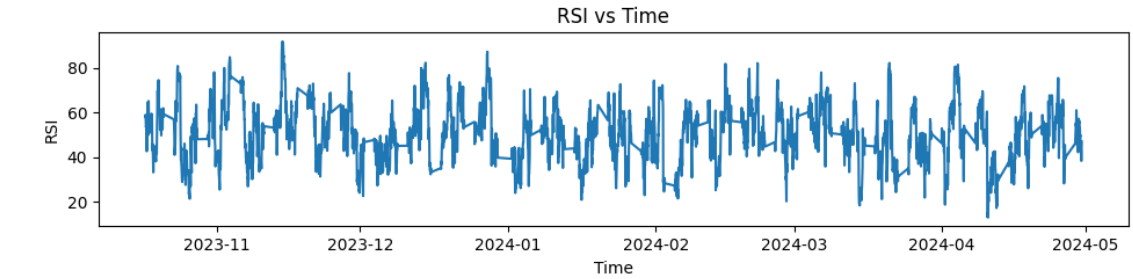


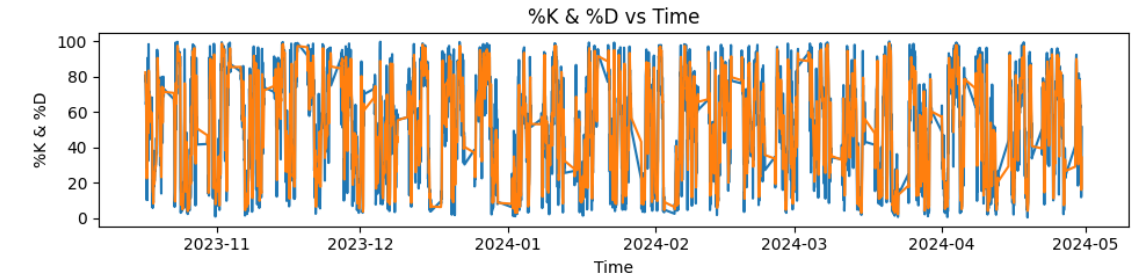

CLosing Price vs Time
Closing Price
1.10
1.08
1.06
2023-11
2023-12
2024-01
2024-02
2024-03
2024-04
2024-05
Time


Volume vs Time
Volume
300000
200000
100000
0
2023-11
2023-12
2024-01
2024-02
2024-03
2024-04
2024-05
Time


Difference between MACD and Signal vs Time
Histogram
0.001
0.000
−0.001
2023-11
2023-12
2024-01
2024-02
2024-03
2024-04
2024-05
Time


MACD & Signal vs Time
MACD & Signal
0.004
0.002
0.000
−0.002
MACD
Signal
2023-11
2023-12
2024-01
2024-02
2024-03
2024-04
2024-05
Time

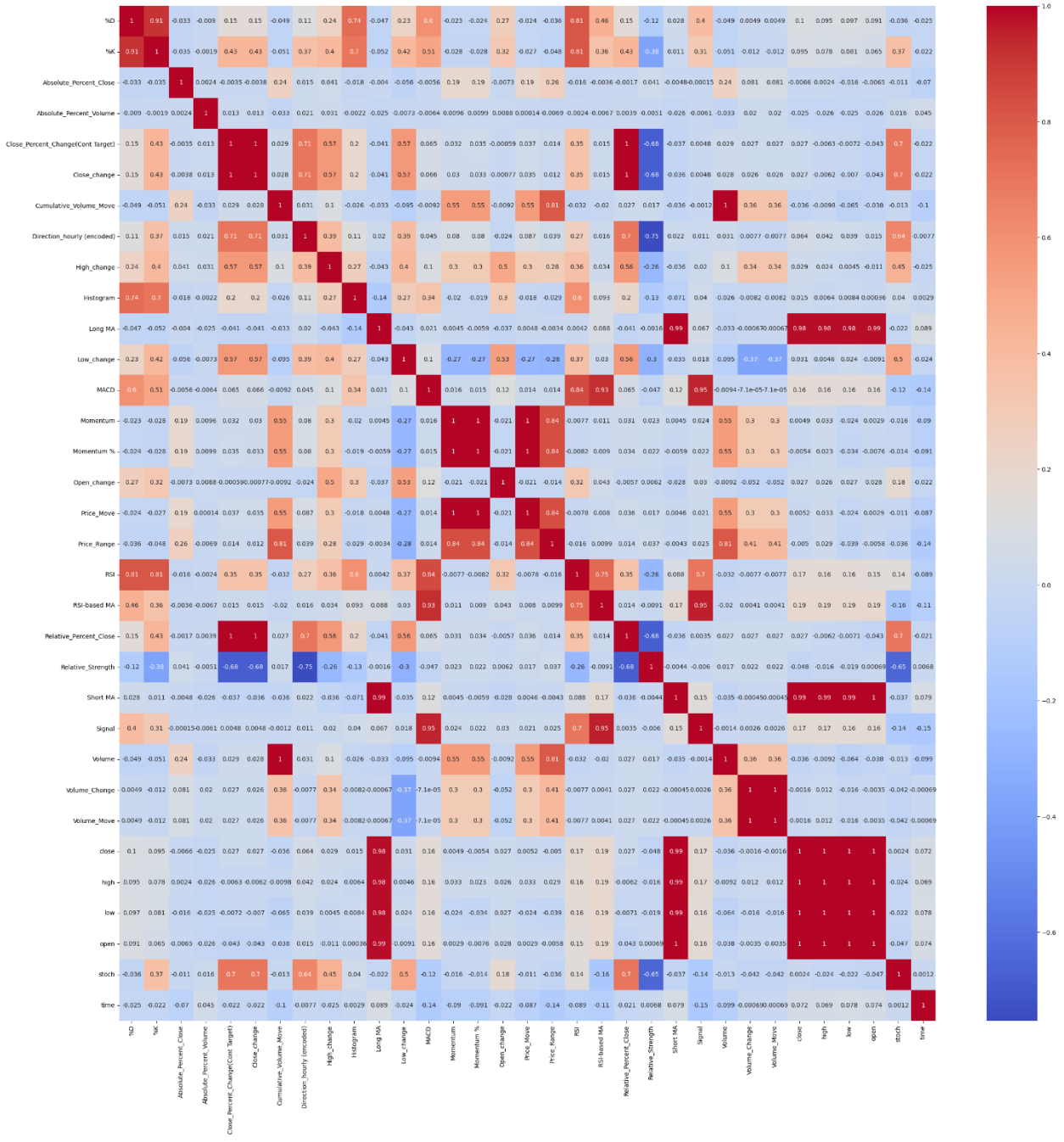

RSI vs Time
RSI
80
60
40
20
2023-11
2023-12
2024-01
2024-02
2024-03
2024-04
2024-05
Time

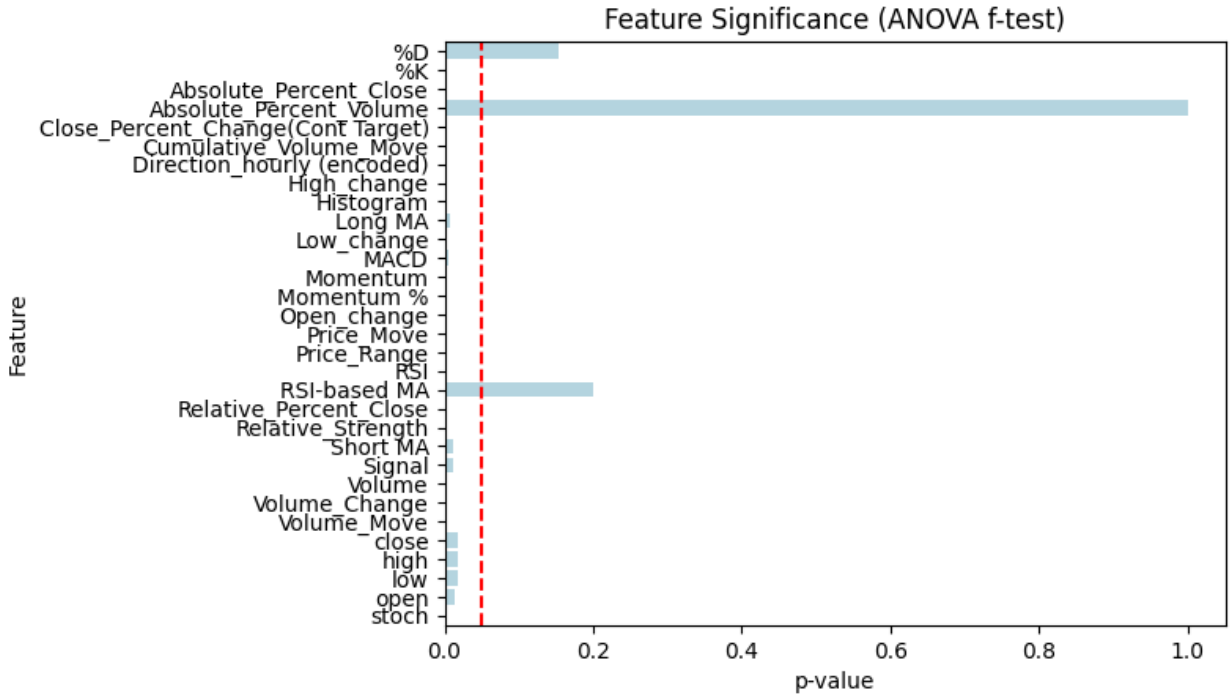

%K & %D vs Time
%K & %D
100
80
60
40
20
0
2023-11
2023-12
2024-01
2024-02
2024-03
2024-04
2024-05
Time

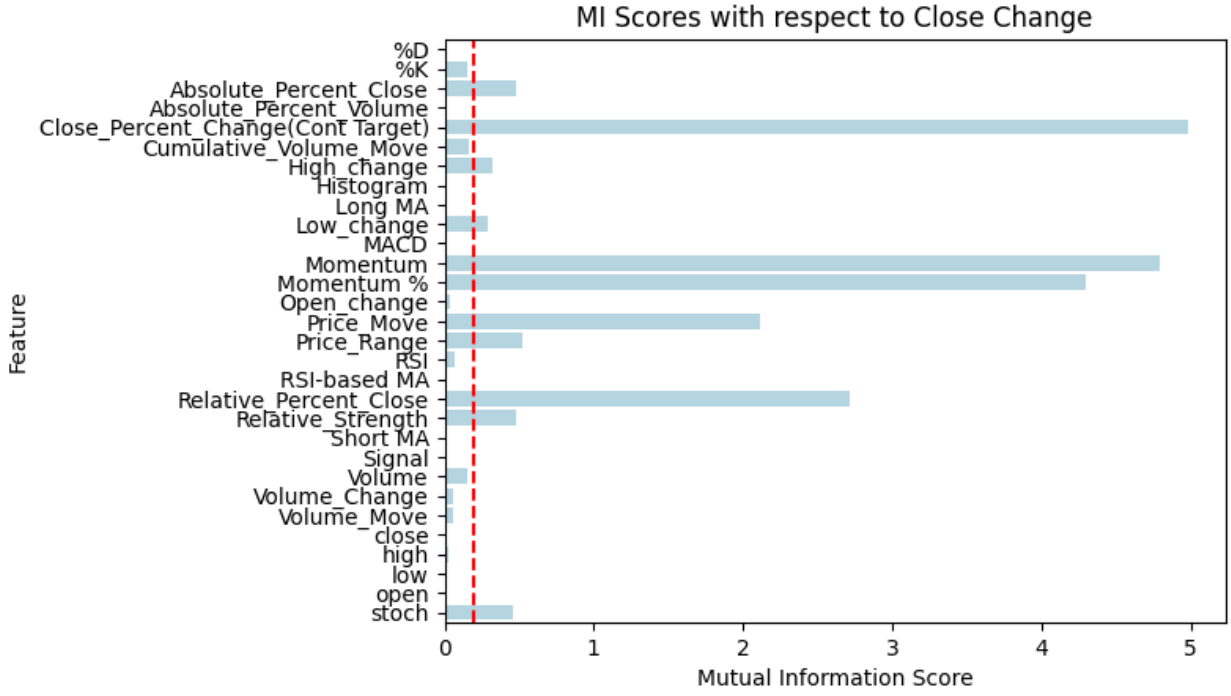
Analyses.

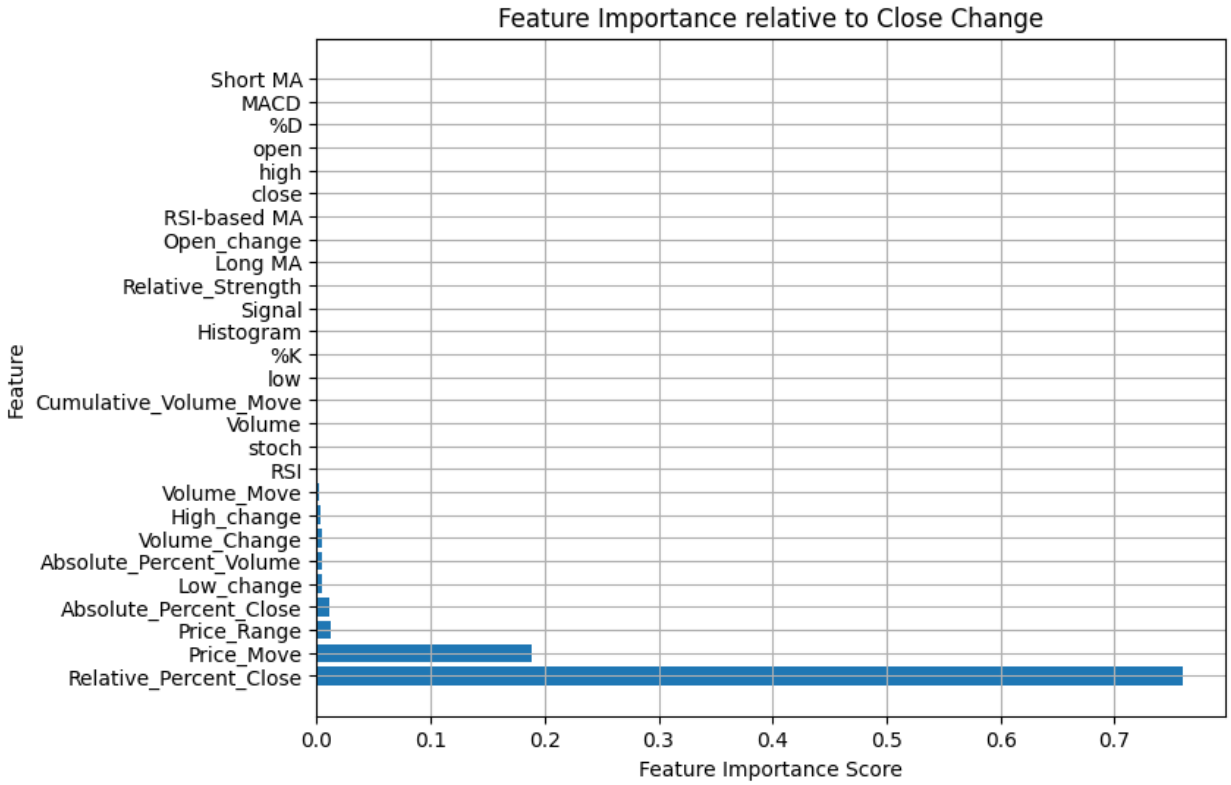
Prior to transformation of data frame into input arrays, we first perform several steps of features selection. First, we compute a correlation matrix, that shows all the correlations between features in the data frame. Since our target feature is the "Close change" price we only look through that column and pay attention to the correlations that are closer to the extremes, 1 or -1. Based on this we observe that the Stochastic indicator, RSI, Low change, High change, %K, %D (somewhat) have decent correlations with our target. Next, we do an ANOVA f-test between the target variable and training features. It doesn't give us any new information since it shows that most training features are the threshold p-values (0.05) and thus have some sort of statistically significant relationship with the target. After that the mutual information score was computed between the target and each of the training variables. A higher score means that given the knowledge about some training feature, the information entropy of the target variables gets reduced. In other words, there is more certainty about the outcome of the target given knowledge about some training feature. As can be observed, a bunch more training features can be added to the list such as Price range, Price move, Absolute Percent Close, Relative Percent Close. Finally, we fit a Random Forest to the data and access the method's importance scores, which is calculated by averaging the decrease in impurity of prediction that results by making decisions at features' nodes across decision trees. The importance scores further confirm our list of selected features. The reason why some of the features could be on this list but are not, like Momentum or Momentum %, is because they are already defined in terms of immediate Close Price change. So, it would be a bit redundant to include them in our training set that is aimed at predicting Close Price change. So based on the above and some domain knowledge we select the following training columns: [Stochastic, RSI, Low change, High change, Histogram, Volume, Volume change, Price Range, Price Move, %K, %D, Relative Percent Close, Absolute Percent Close].

Below from top to bottom are shown the correlation matrix of the features in the first data frame, ANOVA f-test with respect to the close price change (continuous target), mutual information scores relative to the close price change and importance scores between training features and close price change obtained from random forest method.


%D
%K
Absolute_Percent_Close
Absolute_Percent_Volume
Close_Percent_Change(Cont Target)
Close_change
Cumulative_Volume_Move
Direction_hourly (encoded)
High_change
Histogram
Long MA
Low_change
MACD
Momentum
Momentum %
Open_change
Price_Move
Price_Range
RSI
RSI-based MA
Relative_Percent_Close
Relative_Strength
Short MA
Signal
Volume
Volume_Change
Volume_Move
close
high
low
open
stoch
time
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6


Feature Significance (ANOVA f-test)
%D
%K
Absolute_Percent_Close
Absolute_Percent_Volume
Close_Percent_Change(Cont Target)
Cumulative_Volume_Move
Direction_hourly (encoded)
High_change
Histogram
Long MA
Low_change
MACD
Momentum
Momentum %
Open_change
Price_Move
Price_Range
RSI
RSI-based MA
Relative_Percent_Close
Relative_Strength
Short MA
Signal
Volume
Volume_Change
Volume_Move
close
high
low
open
stoch
Feature
0.0
0.2
0.4
0.6
0.8
1.0
p-value


MI Scores with respect to Close Change
%D
%K
Absolute_Percent_Close
Absolute_Percent_Volume
Close_Percent_Change(Cont Target)
Cumulative_Volume_Move
High_change
Histogram
Long MA
Low_change
MACD
Momentum
Momentum %
Open_change
Price_Move
Price_Range
RSI
RSI-based MA
Relative_Percent_Close
Relative_Strength
Short MA
Signal
Volume
Volume_Change
Volume_Move
close
high
low
open
stoch
Feature
0
1
2
3
4
5
Mutual Information Score



Once feature selection is done and the data is transformed into 2 arrays, as described in the above sections, to serve as input into our models we can start training the models.

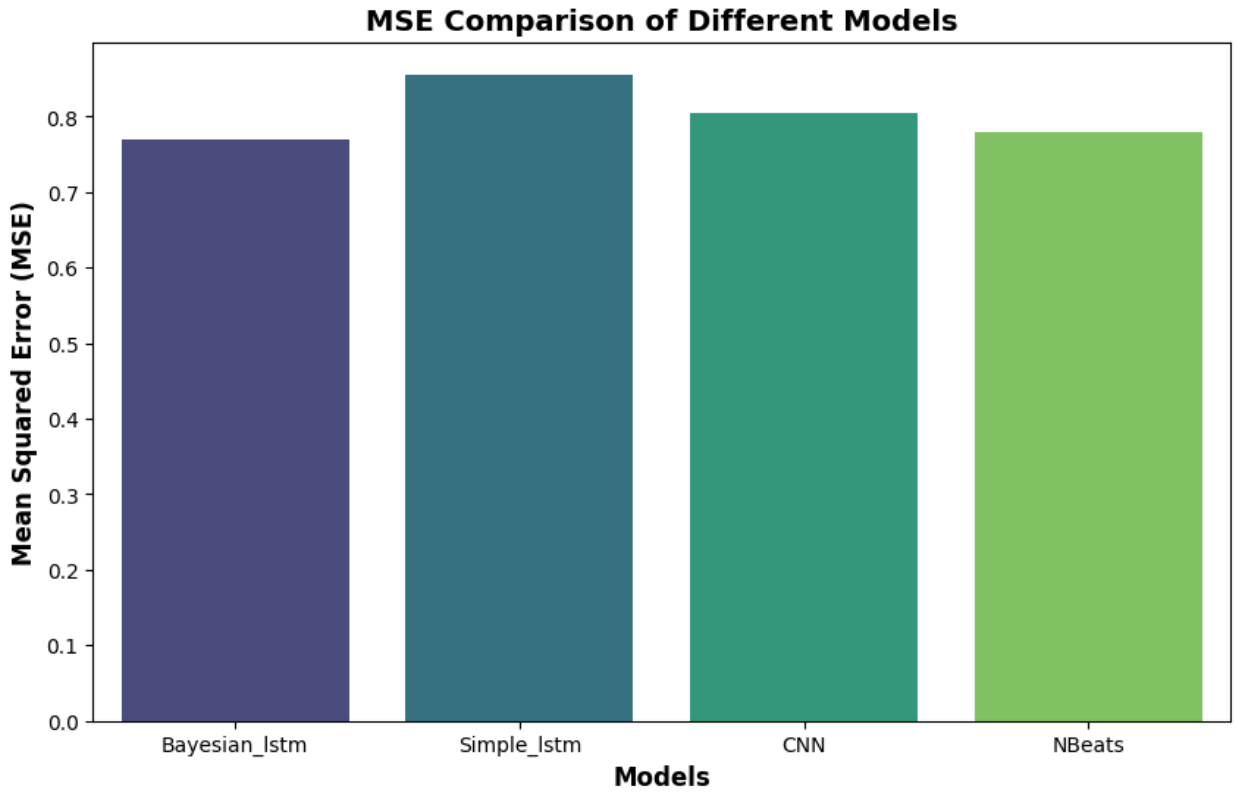
First a CNN model was built, trained and tuned manually. During tuning it quickly became apparent that this model with 2 or more CNN layers overfits. Thus, the model was simplified by reducing the number of layers, adding in a dropout layer and an Average Pooling layer for better generalization. For the same purpose the dropout rate was increased from an initial 0.2 to a 0.4. A dense network was added on top of the CNN layer for depth of learning, however once again the number of neurons had to be decreased from 32 and 16 to 8 neurons due to overfitting. Max Pooling layer was also tried instead of the Average Pooling layer, but it produced worse validation loss (0.83-0.85 compared to 0.73-0.75 of Average Pooling). It was experimented between SGD and ADAM, where the SGD optimizer came up superior. Optimal learning rate turned out to be about 0.0001, below that the validation loss stops improving and above that it starts oscillating without converging. Due to overfitting the number of filters was reduced from 32 to 16; reducing the number further does not improve performance. Different activation functions where tried, such as ReLu, Elu, tanh, sigmoid, but ReLu gave the best performance.

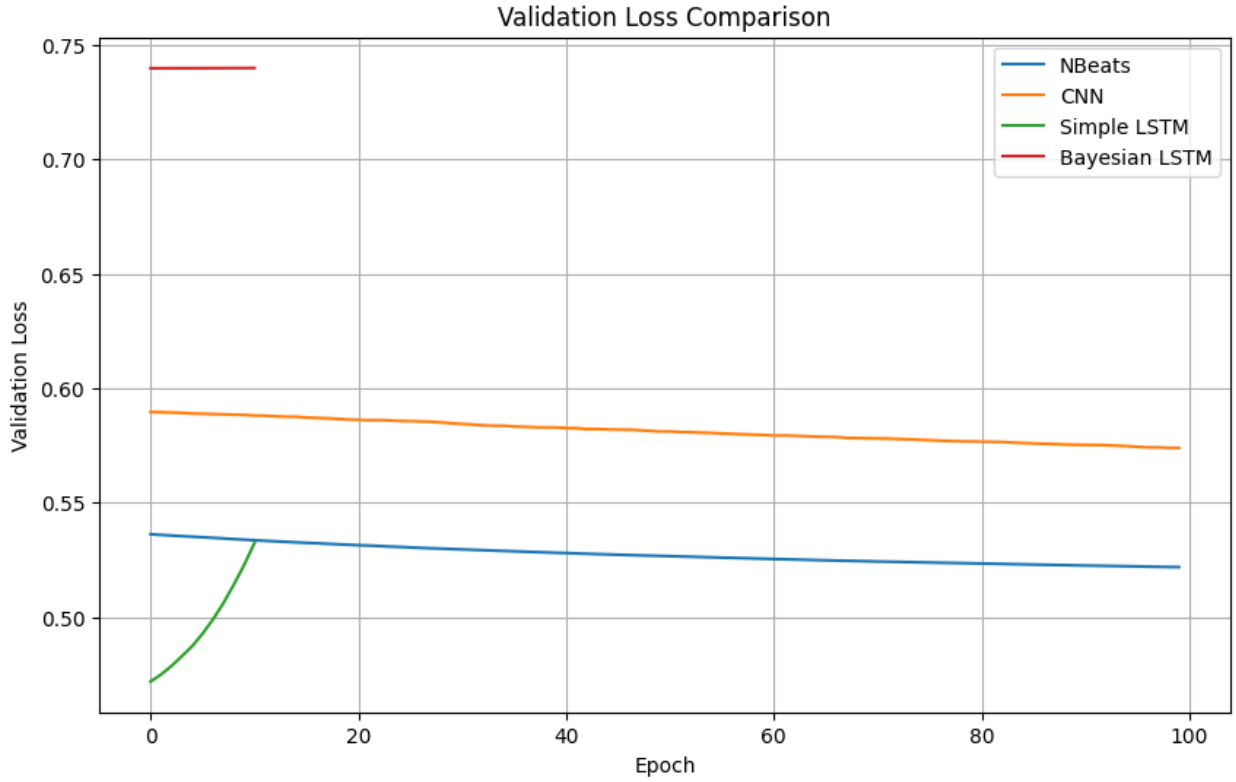
Next two LSTM models were trained and tuned. One was achieved through hyperparameter tuning using Random Search and Bayesian Optimization Search. The other one, with a simpler architecture, was manually tuned. Out of the automated searches the best set of hyperparameters was picked based on lower validation loss. Bayesian Optimization yielded better parameters, where there are 3 lstm layers in the model each having 128 units, the optimizer is SGD with optimal learning rate of roughly 0.0001, dropout rate of 0.2 and a ReLu activation function. The

other simpler lstm model was manually tuned similarly to the previous CNN model. Just like the previous model it was overfitting, a Batch Normalization layer was added a dropout layer as well with rate 0.2. Again, SGD performed better than Adam, with the same pattern of learning rate, where 0.0001 is the golden middle (smaller it shows little to no improvement and if larger it starts oscillating. ReLu is still the activation function of choice, over tanh and Elu. Manual tuning showed that for that particular model, 32 units and 1 layer is ideal; with 65 units it starts overfitting, same with 2 or more layers; 8 units and the model can't converge on its' validation loss. One potential reason why so many model iterations overfit the data is that it might be caused by the 12-hour overlap between the consecutive 24-hour observation windows. So that when the LSTM or CNN model receive the next time sequence of features, it already saw some of them in the previous input. The model might memorize or become biased towards it.

Finally, a N-Beats model was also fitted to the data. N-Beats is a deep learning model designed specifically for timeseries forecasting. It was attempted to tune this model using RandomSearch. However, unfortunately the package is not compatible with TensorFlow. To resolve this it was tried wrapping N-beats model in a TensorFlow method, hoping that it would return a tensor object for the TensorFlow class. Therefore, it was not possible to use the GridSearch hyperparameter tuner from Keras which requires a tensor input. Otherwise subclassing the model was tuned manually. By default the model has two blocks that each have a certain number of units in them (32 in our case). The blocks can either be generic, or specialized for trend prediction and seasonality prediction. Not much of a difference, since it has little impact on validation loss. The optimizer of choice is still SGD and learning rate is 0.001. Other optimizers like Adam and alternative learning rates increase the validation loss. Number of units per layer and the thetas-dim tuple has little to no impact on validation set improvement. Even if the thetas-dim is heavily skewed to one of the sides. Experimenting with different batch sizes between 8, 16, 32 and 64 it became apparent that there is not that much difference between any of the 4 in terms of impact on model performance, so 16 was picked just like for previous models.

After all the of the 4 models were tuned, prediction were generated and the mean squared error of each visualized in a bar plot. The models were also compared based on their validation losses on the second dataset. However, since callbacks were employed, some models have shorter loss curves due to divergence or very quick convergence (no improvement).


MSE Comparison of Different Models
Mean Squared Error (MSE)
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
Bayesian_lstm
Simple_lstm
CNN
NBeats
Models


Validation Loss Comparison
NBeats
CNN
Simple LSTM
Bayesian LSTM
Validation Loss
0.50
0.55
0.60
0.65
0.70
0.75
0
20
40
60
80
100
Epoch

Results/Conclusions.

Based on the above we would like to pick one or two best models out of the mix to potentially improve on them and implement them for back testing on real time market data. Most promising model would be the N-Beats model. Even though it has the second lowest mean squared error, it has the best validation loss curve out of the 4 models, since it went all the 100 epochs and still was slowly decreasing at the end, which indicates stability and great potential for improvement. The second-best model would be the CNN model due to its great loss curve. As can be seen in the bar plot the mean squared errors are very close for all models, so it shouldn't play a big part in the model picking process.

Some recurring themes that were observed throughout the project are that the SGD optimizer consistently outperformed the Adam optimizer. Generally, greater model complexity (more layers) would cause performance decrease (increase in validation loss), which was probably caused by pattern memorization in the previous overlaps of the 24-hour observation windows. The consistent overfitting might have something to do with the way the input data for the models was structured in the arrays, where consecutive 24-hour windows would overlap on a 12-hour period. Across all models the optimal learning rate was 0.0001. Most models were overfitting, meaning the validation loss was stagnant while the training loss would decrease. So, throughout most models, with the exception of the N-Beats model, layers such as Dropout or Batch Normalization had to be added in to increase the generalization capability of the models.