**A TIME SERIES PREDICTOR FOR STOCK MARKET PRICES: BITCOIN CRYPTOCURRENCY**

**PROBLEM STATEMENT:**

The stock market since its renowned inception by the East India trading Co. well back into the colonial era, has become a remarkable powerhouse of investment and wealth building; beneficial to both private investors as well as the businesses whose shares are being traded and valued. The terrible downside of this is the associated risk, caused by market forces and dare I say human whimsy? With that said though, there are a multitude of factors used to determine risk, one of which is market trend: Is the stock value rising or falling in short.

Thus the purpose of this project as an initial step is to train a time series model which predicts stock prices with an %85 - %95 accuracy by April 10, 2024 as a piece for a larger model, ultimately directed towards automating ideal trading actions depending on pre-defined investing goals.

As a piece of a much larger project in the long term, we will endeavor to achieve the best reasonable accuracy and precision possible with this model, between %85-%95 at least, for proof of concept, in predicting stock market values. Measured most likely using MAE, and RMSE values for the average distance between predicted and true values.

This will be a general application model which SOLELY predicts future prices derived from previous price points in the Time Series. We will look to improve upon this prediction by taking into account predictions from other models and factors in order to make the best prediction possible but this as stated before is the long-term goal. This project will ONLY focus on the Time Series data.

We will need to worry a lot about variance, as the stock market is a very noisy place. We will need to take great care in preparing and transforming our data to give the model the best odds of success possible within reason. Since we will only be focused on Time Series, we will be limiting ourselves to the variations of ARIMA models and LSTMs.

A short list of stakeholders would be: Investors, Entrepreneurs, CEO's, Private Investors, Financial Advisors, Portfolio Managers, Financial Security, and Risk Management. We will potentially pulling data from these sources: NASDAQ, Quandl, OpenFin, MarketStack, FinnHub, Polygon, and https://www.kaggle.com/datasets/finnhub/sp-500-futures-tick-data-sp

**DATA WRANGLING:**

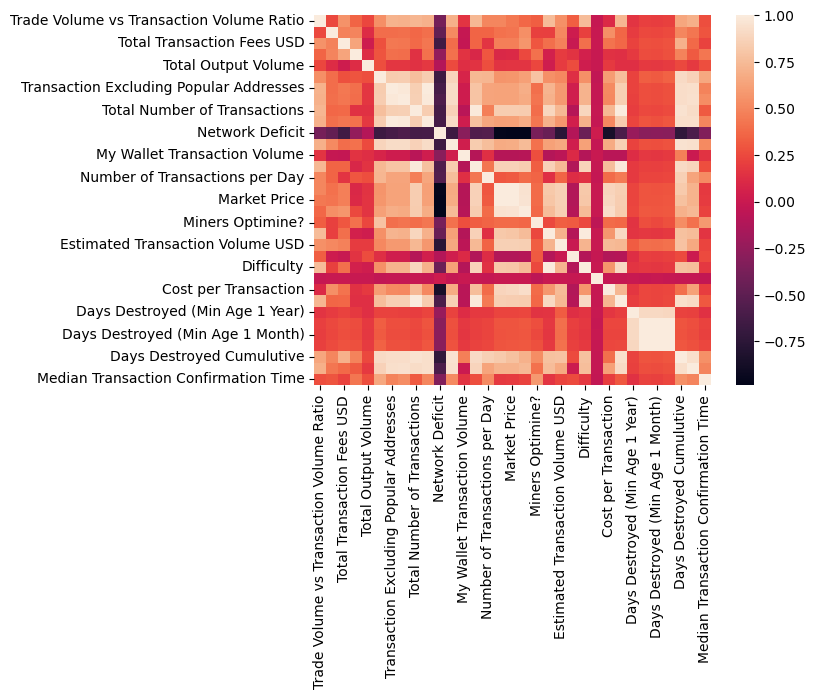
Data was first pulled from the NASDAQ API after navigating which sources were available which didn’t require paying a subscription to NASDAQ. Ultimately it was decided in order to fit within the time constraints, it would be better to select a single stock or currency to model. Thus Bitcoin became the target of our analysis.

Bitcoin presented a couple of interesting problems, fortunately inconsistent datetimes was not one of them, however there were issues with how the data was pulled down, it was in completely the wrong format. It came down as a 3 column stream consisting of Dates, an Identifier Code, and a value, many of which weren’t necessarily monetary values. After reviewing the Bitcoin repository documentation we were able to determine what the Identifier Codes meant and transformed the DataFrame of streamed data such that these codes were our column headers and we translated them into their meanings.

Upon doing this it became abundantly clear that some features were discontinued after 2016, which was confirmed from the NASDAQ Bitcoin documentation. Upon graphing the Time Series data, it was shown that these columns really weren’t going to be relevant and so we dropped them from the DataSet.

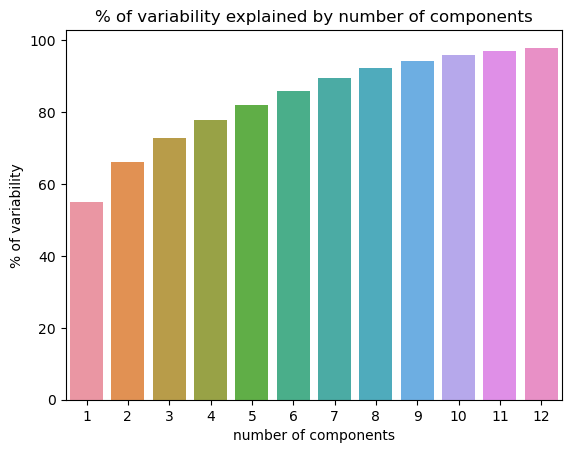
After that there were trace data points towards the end of the dataset that had missing values in random columns with no particular pattern to them. They were few so we dropped them. Also there was a stretch of points which held missing values which we imputed using a linear algorithm to connect from beginning to end of the gap. Thus our data was ready to be explored.

**EXPLORATORY DATA ANALYSIS**

I traditionally like to start with Correlation Heatmap. It provides me with a visual place to start really delving into the data. Thus this is what we found: The data though holds the appearance of being random, it is heavily correlated.

Since the ultimate goal is to create a Price predictor we looked at data that was either highly positively correlated with Market Price or highly negatively correlated. Our Master list being: Total Transaction Fees, Total Bitcoins, Total Number of Transactions, My Wallet Number of Users, Market Capitalization, Miners Revenue, Estimated Transaction Volume USD, Estimated Transaction Volume, Difficulty, Cost Percent of Transaction Volume, Cost per Transaction, API Blockchain Size, and Median Transaction Confirmation Time.

We quickly found when looking at correlations to the correlations, and up to 3 layers deep data started running around in circles. Thus we decided to take advantage of PCA analyses to reduce the dimensionality of the data to make it more manageable and to determine the Eigen values that most contributes to the explainable variance of the data. We ultimately arrived at 12 PCAs to account for 95% of the plausible variance to make accuracy precise as possible.



Our Eigen values were determined to be: Difficulty,

Market Capitalization, Estimated Transaction Volume USD,

USD Exchange Trade Volume,

Total Number of Transactions,

Hash Rate, My Wallet Number of Users,

Miners Revenue,

Total Bitcoins,

Total Output Volume, Total Transaction Fees, USD,

Cost Percent of Transaction Volume

With that we prepared to engineer these PCAs to create Time Series forecasts.

**FEATURE ENGINEERING**

The first step was to double check the stationarity of the PCAs, we found there were 4 columns which needed to be transformed prior to being fed to the model, This determination was made using the Adfuller method which indicated that, PCAs 1, 2, 6, and Market Price needed to be transformed, each needing 2-3 differences and Market Price requiring a Log function to reduce these data sets to Stationary.

This was for the ARIMA and SARIMAX models which require stationarity however and LSTM doesn’t care so we cultivated 2 data sets to divide into X, y and train, test, val sets for the modeling. What we did have to confirm is if the data largely conformed to normal distributions for the LSTM which thankfully for us it did! Now we could proceed to forecast.

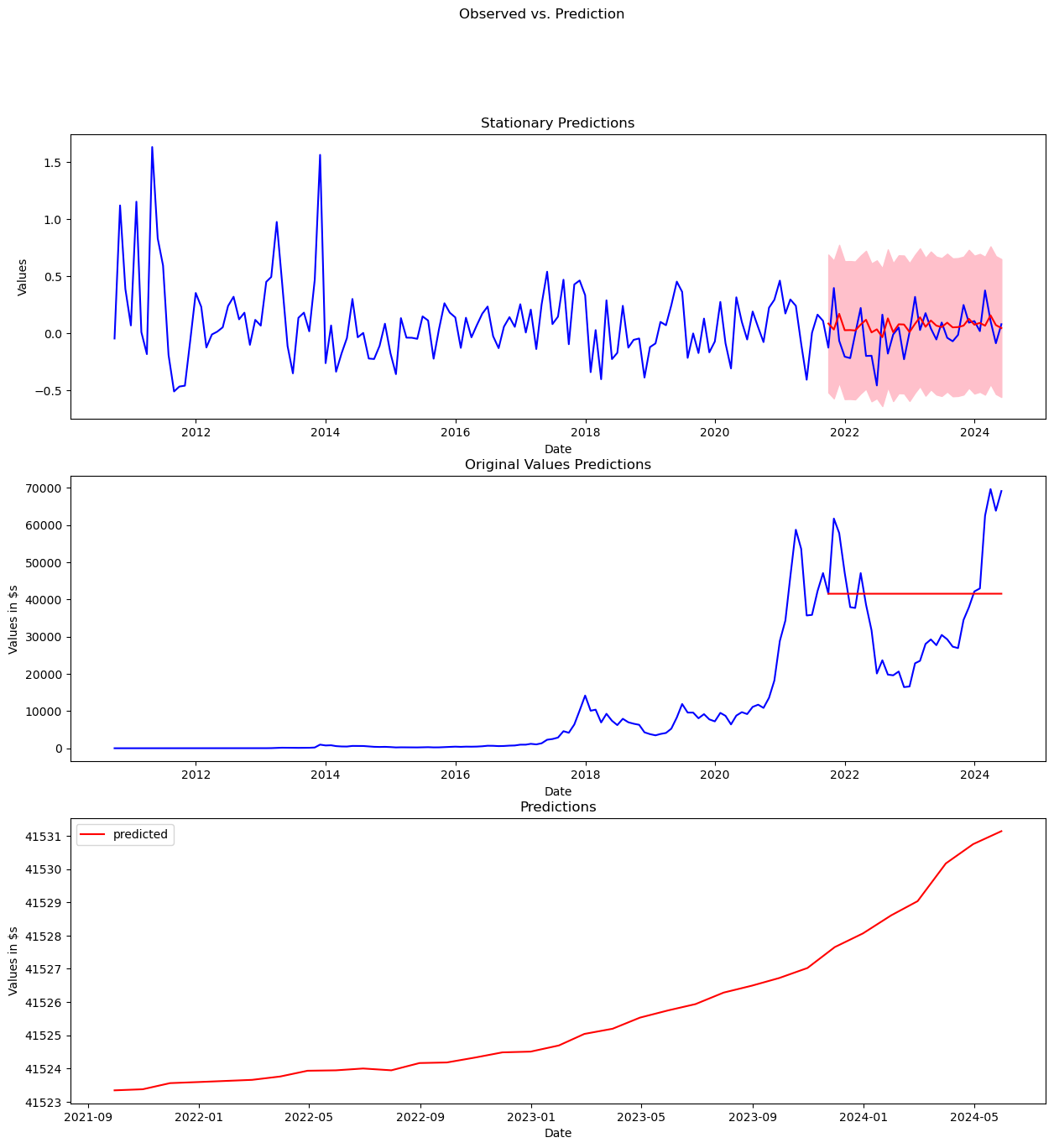
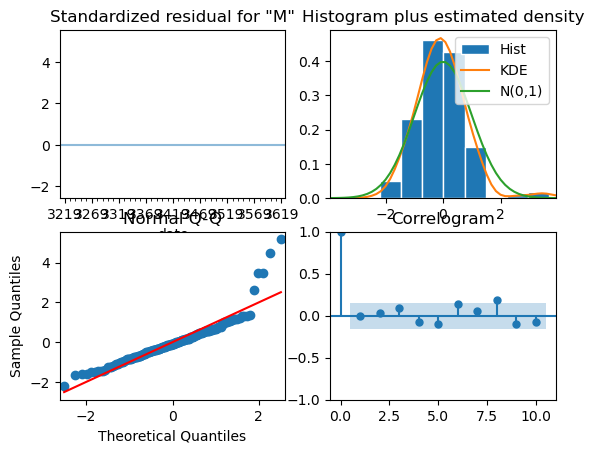
**MODEL PREPROCESSING:**

As Previously mentioned, we cultivated 2 datasets to meet the needs of SARIMAX models and their variants, as well as a deep learning LSTM model. For the ARIMA model we split at 75/25 for Train/Test data and for the LSTM we performed a 75/15/10 split for Train, Validation, And Test sets respectively. All Data sets were scaled using either the Standard scaler() or the MinMax() scaler.

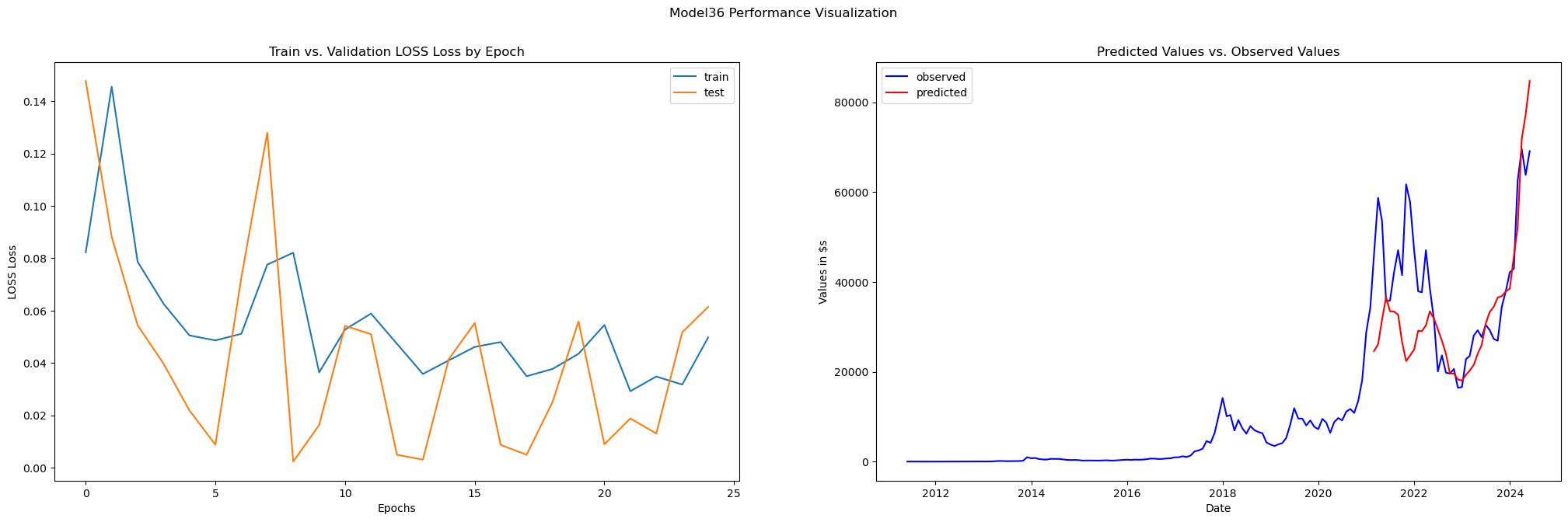
**MODELING:**

We selected three model types to test, ARIMA, LSTM, and SARIMAX. However during the course of hyperparameter tuning, Our ARIMA and SARIMAX models were reduced to simple MA, and MAX models. After confirming functional input and Visualization of model performance and metrics we proceeded to hyper parameter tune the models with the following results:

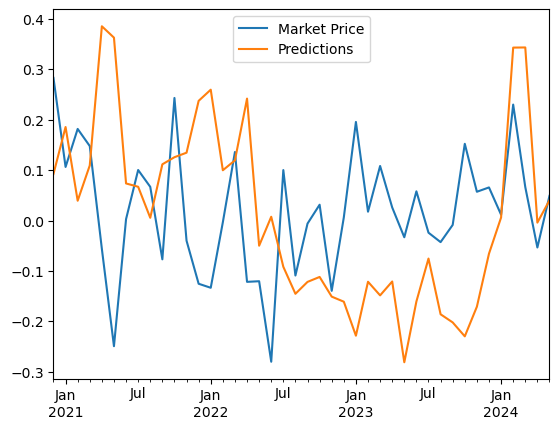
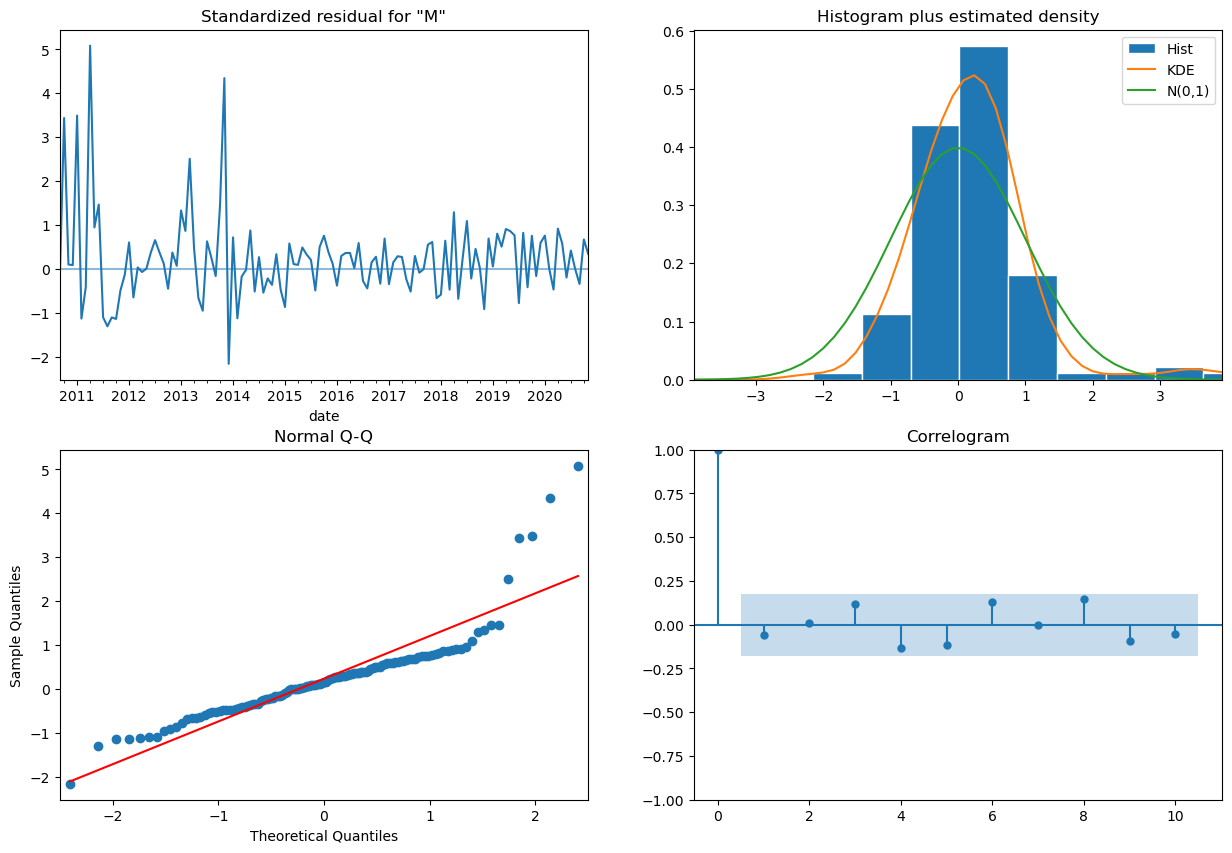
* ARIMA (0,0,1) or Moving-Average Model



* LSTM (12 Exogenous variables, 1 endogenous) Model 36



* SARIMAX(0,0,1) Moving-Average with Exogenous variables



There were many trials for the LSTM model and though there were models that performed better in their metrics, due to their weight and seemingly more concerning, extreme value predictions towards the end of the Market Price time-series.

**INSIGHTS:**

First and foremost, as far as good forecasting goes. The LSTM model performed far superior compared to the other two. This is in part due to 2 factors, I think. The LSTM doesn’t require a transform of the data to be stationary, it predicts on original values (scaled). The second is because we were able to window the data with the LSTM model. Which window size will be provided in the included model\_metrics.txt file. We were unable to find a way to window the predictions from the resultant MA, and MAX models which leads to that ‘random slop’ of predictions. It was likely noticed that we forewent transforming the stationary data back to original data for the MAX model as after having seen this kind of data before we could infer that it would result in predictions awfully similar to the MA model.

From working with the data it became readily apparent that working with the original data as is, is not sufficient to formulate a predictor. Even when performing EDA on the Data, searching for Correlated and Causal relationships within the data, it very quickly turned into running around in circles. I would go so far as to say that perhaps the single most important step made in this entire process was to use Principal Component Analysis to simplify the dimensionality and make it serviceable.

To further this project I would like to try this model pipeline on other Currency data sets, to verify if it is capable of accurately forecasting Market Value of said currency. I would also like to investigate if I wasn’t too conservative in my hyper parameter tunings of the ARIMA and SARIMAX models. Truthfully I thought they would have performed better. I would also investigate if there’s a way to create a lighter version of the LSTM model to make it predict faster, fast enough that it would be more valid in production, plausibly a GRU model or a CNN model.