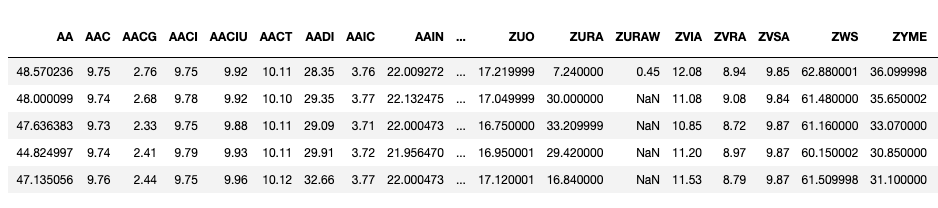
Lab3-part2

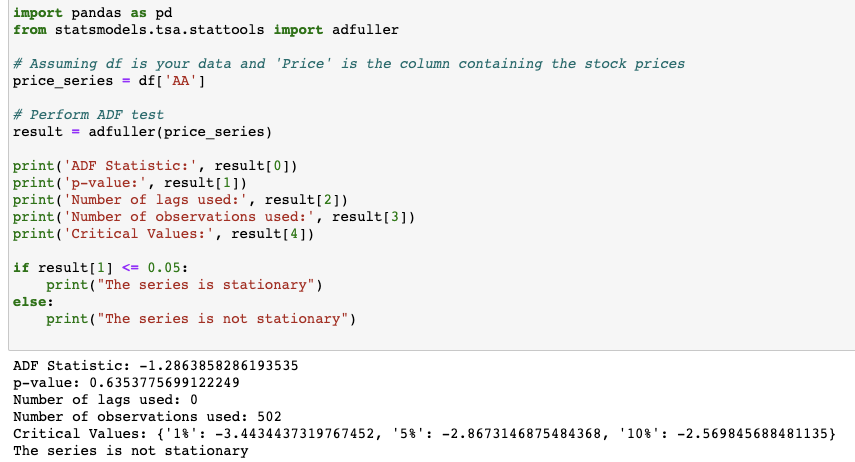
1. Algorithm development

We conceptualized a strategy wherein we would amalgamate the Arima and LSTM models to forecast stock prices. The allocation ratios between the two models will be determined by their accuracy, ensuring the optimal functioning of the hybrid model.

Hence, we did Arima model first.

We do the simple data cleaning include the remove the missing value and outlier

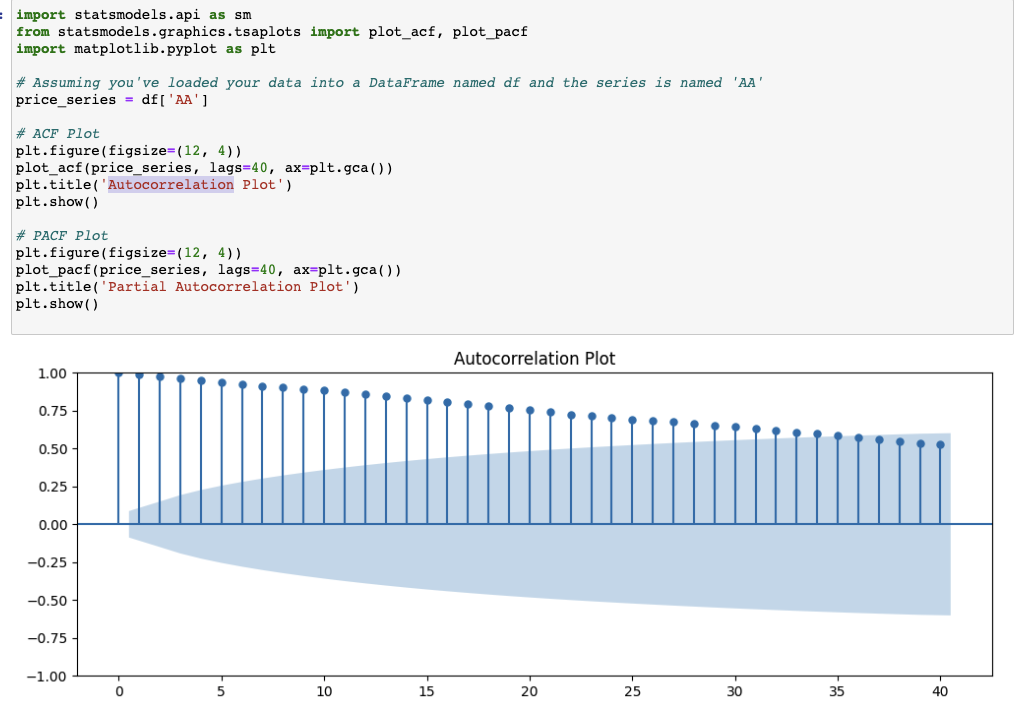
1.We did the Augmented Dickey-Fuller (ADF) test to determine if the model is stationarity or not( If the data isn't stationary, differences of the data can be taken until it becomes stationary.)



2.ARIMA models are generally denoted as ARIMA (p,d,q) where p is the order of autoregressive model, d is the degree of differencing, and q is the order of moving-average model

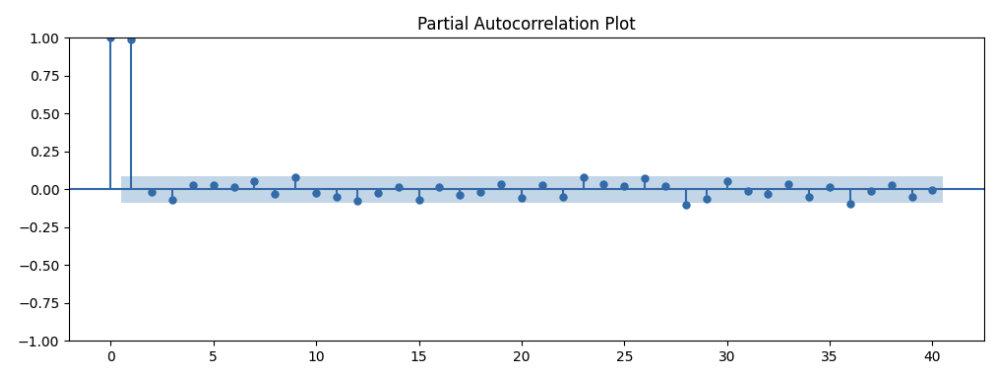
We pick ARIMA(2,2,1) for the stock AA

The autocorrelation plot (ACF) gives correlations of the series with its lags. It helps in determining the MA(q) term.



ACF (Autocorrelation Function) Plot:

* An ACF that trails off suggests a potential MA(q) term. The number of significant lags before the trail off can give an idea of 'q'. Above the plot show us that the first two lags are significant and then the plot trails off, since p =2



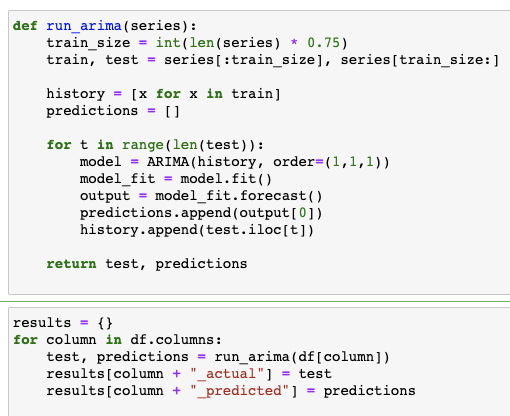
PACF (Partial Autocorrelation Function) Plot:

* A PACF that trails off suggests a potential AR(p) term. The number of significant lags before the trail off can suggest 'p'. Ithe first two lags are significant and then the plot trails off since p=2.

If your data is non-stationary (which you may have detected using the ADF test or visual inspection), then you need to difference the series. The value of

d is the number of differences required to make the series stationary. Typically,

d=1 is often sufficient.



Fun run\_arima, is defined. This function can predict future values of a given time series using the ARIMA model. Here's a breakdown:

Data Splitting: The time series data is split into a training set (75% of the data) and a test set (the remaining 25%).

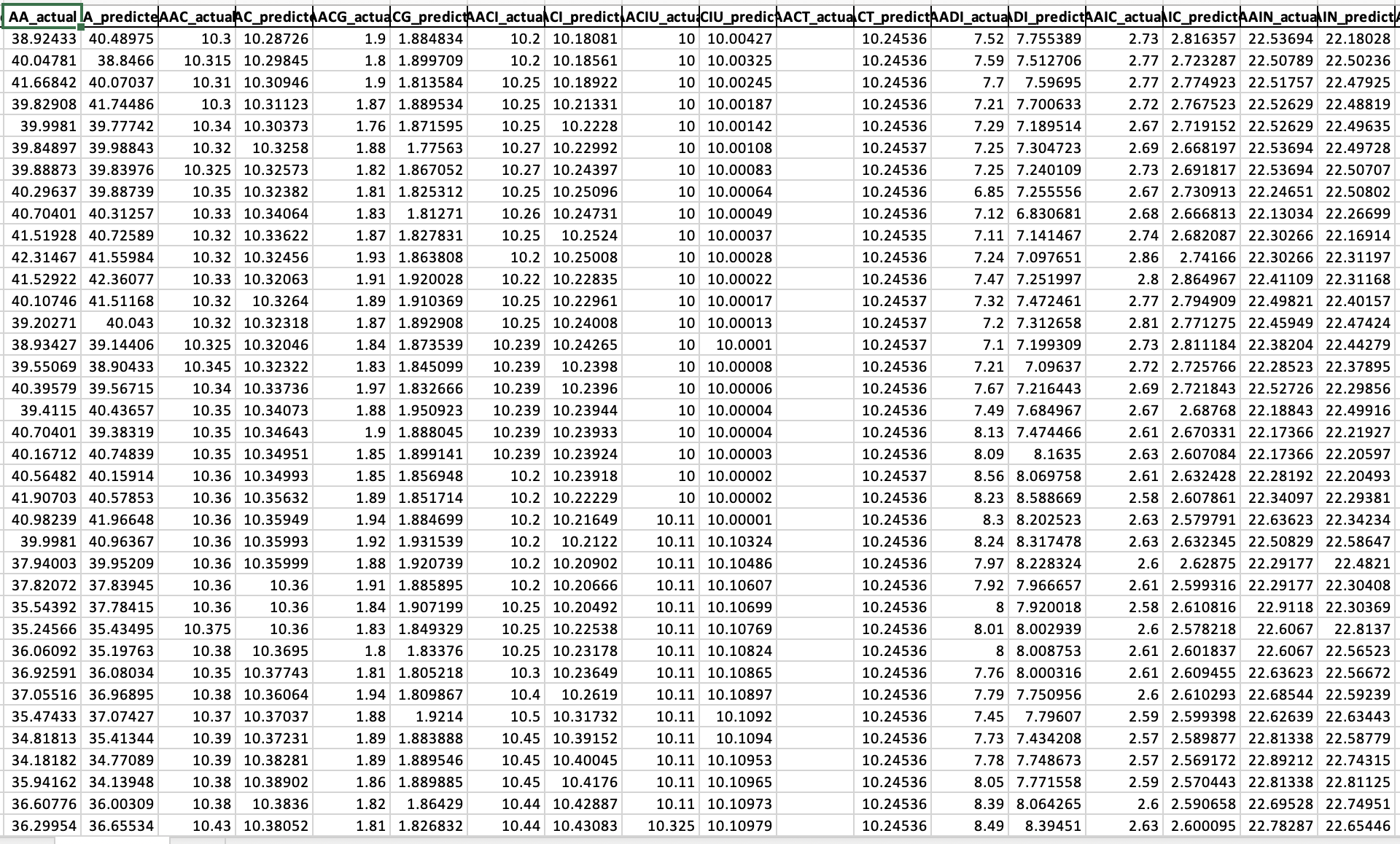
Model Training and Prediction: For each data point in the test set, the ARIMA model is trained on all preceding data points, and a forecast is made for the next point. This approach is often referred to as a rolling-forecast origin because the 'start point' from which the forecast is made rolls forward in time.

Configuration: By default, the ARIMA model is set to parameters (1,1,1). However, you might need to adjust these based on the characteristics of your specific time series data.

Multiple Series: If you have a dataframe (df) with multiple time series columns, the code will loop through each one, make predictions, and store the actual and predicted values in a results dictionary.

To use the code, ensure you have the pandas and statsmodels libraries installed, import the necessary modules, and call the run\_arima function with your time series data.

Here is our predict data



2.LSTM

We hope to predict the future trends of the stock market using LSTM in order to achieve forecasting of future stock price movements. The following content outlines our code process and data preprocessing steps.

(1) Data preprocess

This code defines a Python function called `split\_time\_series\_data` that is used to split time series data into training and testing sets. Here's an explanation of the code. Import necessary libraries. Define Function and the function takes four parameters.

- `time\_series\_data`: This is the input time series data that you want to split.

- `train\_ratio`: A float (between 0 and 1) representing the ratio of the data to be used for training. If provided, it will be used to determine the size of the training set.

- `test\_ratio`: A float (between 0 and 1) representing the ratio of the data to be used for testing. If provided, it will be used to determine the size of the testing set.

- `split\_index`: An integer representing the index at which to split the data into training and testing sets. If provided, it will be used as the split point.

The function first checks which method of splitting the data is specified (either `train\_ratio`, `test\_ratio`, or `split\_index`). If none of them is specified or if more than one is specified, it raises a `ValueError` with the message "ERROR."

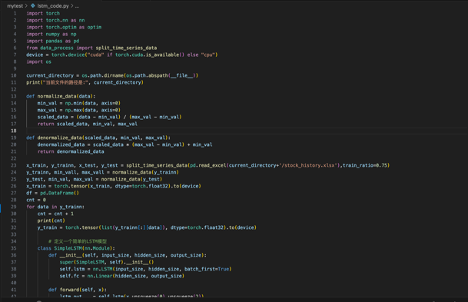
- `x\_train` is created as an array of integers representing the indices of the training data.

- `x\_test` is created as an array of integers representing the indices of the testing data. The indices for the testing data start from where the training data ends.

- `y\_train` is set to the training data.

- `y\_test` is set to the testing data.

Finally, the function returns four variables: `x\_train`, `y\_train`, `x\_test`, and `y\_test`, which can be used for further analysis or modeling.



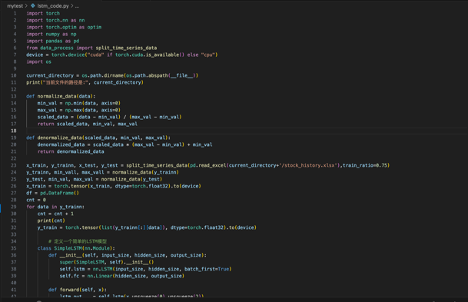
(2) LSTM model Implementation

Predict time series data using a LSTM (Long Short-Term Memory) neural network. Here's an explanation of the code step by step:

Fisrtly, we need to import Libraries. The script starts by importing various libraries, including `torch` (PyTorch), `numpy`, `pandas`, and the module `data\_process` to get the processed data. Additionally, it checks whether a CUDA-compatible GPU is available and sets the device accordingly. And transformer the model and data to GPU for faster training.

Setting the Current Directory for read and save data. The current directory of the script is determined using `os.path.dirname(os.path.abspath(\_\_file\_\_))`, which is the directory where the script is located. This directory path is then printed.

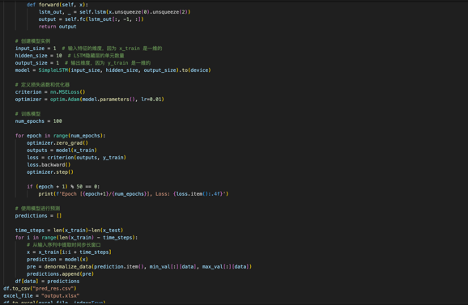
Implement Data Normalization Functions. `normalize\_data` and `denormalize\_data` functions are defined to normalize and denormalize data using the minimum and maximum values. The data with normalization functions can help model better train and easy to predict.

Implement Data Splitting. The script uses the `split\_time\_series\_data` function from the `data\_process` module to split time series data into training and testing sets. It reads data from an Excel file named 'stock\_history.xlsx' located in the current directory and splits it into `x\_train`, `y\_trainn`, `x\_test`, and `y\_test` datasets.

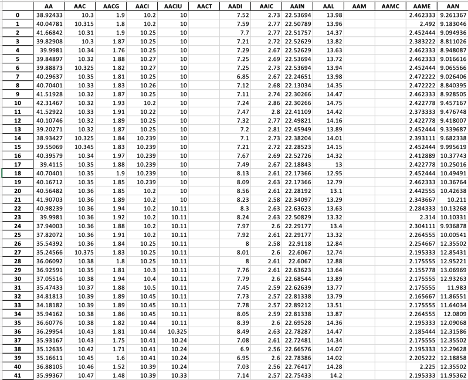
Train a Simple LSTM Model. The script enters a loop where it iterates over each column in the `y\_trainn` dataset (presumably representing different stock price time series). For each column. Simple LSTM model is defined using the `SimpleLSTM` class. This model takes input sequences of length 1 and predicts a single value. The model is moved to the GPU. The loss function (Mean Squared Error) and optimizer (Adam) are defined.

Inside the training loop for each stock's time series data. The optimizer gradients are zeroed using optimizer.zero\_grad(). Model predictions are computed for the entire x\_train dataset using model(x\_train). The Mean Squared Error (MSE) loss is calculated between the predicted values (outputs) and the corresponding y\_train values. The gradients are computed and updated using backpropagation with loss.backward() and optimizer.step(). Every 50 epochs, the current loss is printed.

After training the LSTM model for each stock's time series data, the script enters a loop to make predictions. For each stock, it iterates through the x\_train dataset (presumably a time window) to make predictions using the trained model. Predictions are denormalized using the denormalize\_data function and stored in the df DataFrame with columns corresponding to different stocks. Save Predictions to CSV and Excel.

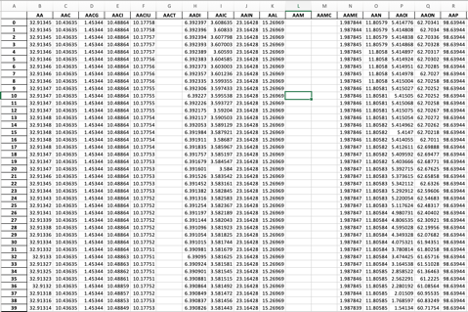


The actual value



The predicted value

.xlsx format



.csv format

2. Mock Trading Environment

Our system allows users to pick their stocks for the portfolio and will implement trading strategies for them, using the file “trading.py”. We used the two models we generated based on their mean square error as weights and the Relative Strength Index (RSI) for buy/sell signal detection. Our model shows the amount for each stock and the total value of the portfolio over time.

The output of our model is as below: (the file used 'arima\_predictions\_all\_columns\_7000.csv' and 'pred\_res.csv', so please include those files when running the code)

