**Task Report Cos30018 Option B**

**B.7: Extension**

**Name: Le Bao Nguyen**

**Student Id: 104169837**

1. Install pytrends library:

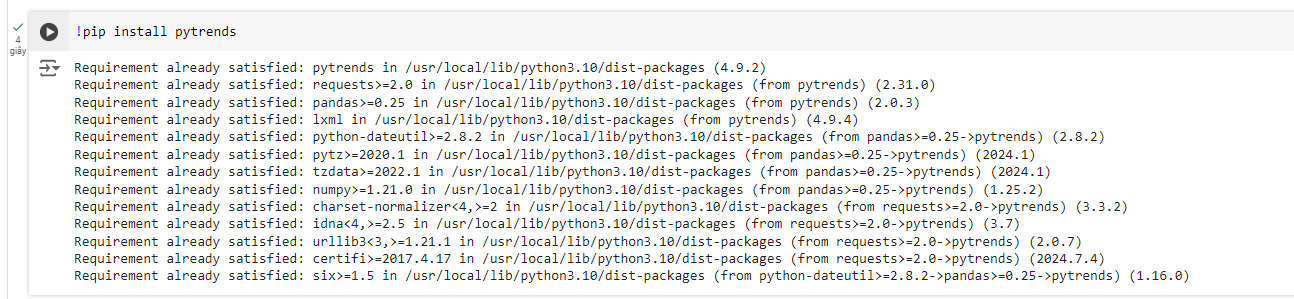


Figure 1: Downloading libraries to run the code.

1. Importing libraries (From the task B.2 + new libraries):

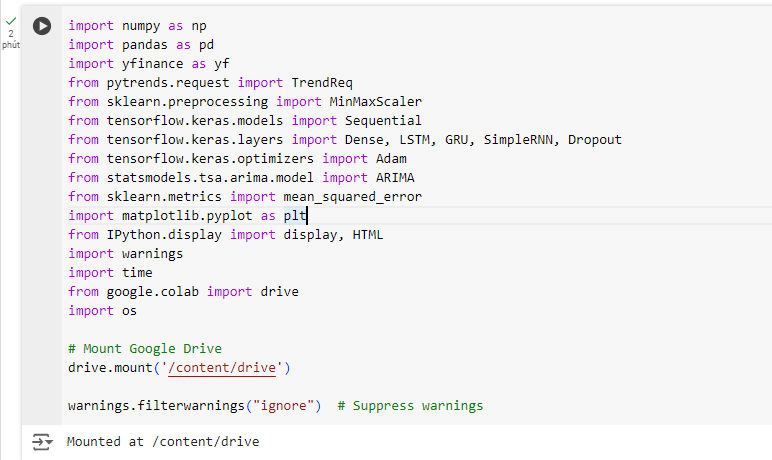


Figure 2: Importing libraries to run the code.

* The script imports previous libraries and new libraries:
* "pytrends:": A Google Trends API that lets you get trending search data straight from the source.
* "time": Provides various time-related functions, including delays.
* "os": A module that gives users access to operating system-dependent features like file system reading and writing.

1. Fetch google trends data:

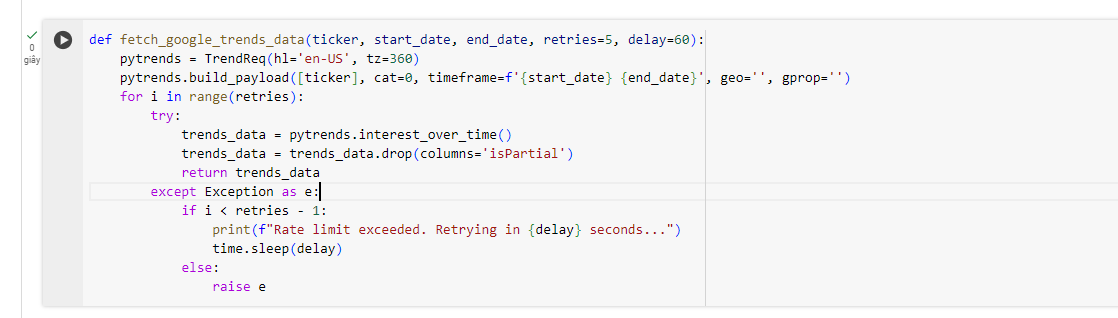


Figure 3: Fetching google trends data.

* Purpose: Gets Google Trends data for a ticker within a particular time frame.
* Details:
* TrendReq: Initializes a request object for Google Trends.
* build\_payload: Builds the API request payload using the given parameters.
* Retries loop: Attempts to fetch the data multiple times in case of rate limit issues, with a delay between retries.
* interest\_over\_time: Retrieves statistics on interest over time for the given search phrase.
* drop(columns='isPartial'): Removes the 'isPartial' column, which indicates if the data is incomplete for the current period.

1. Data loading and processing (From the task B.2 + add pytrends):

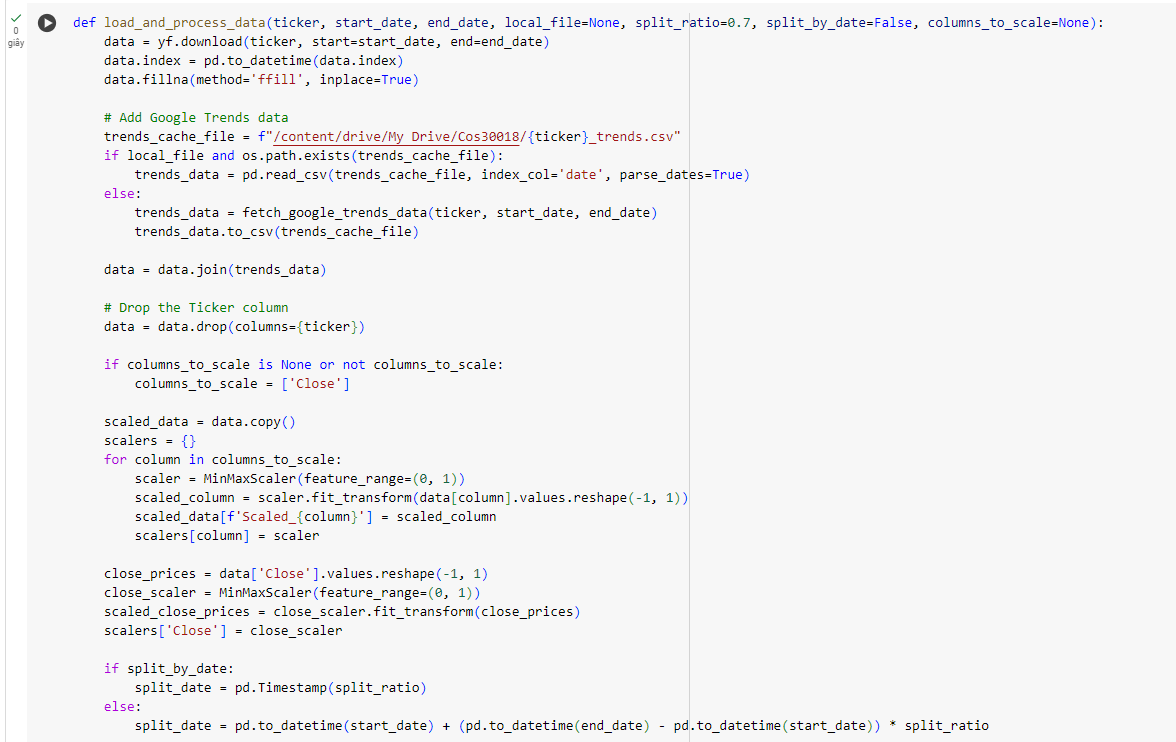


Figure 4: Loading and processing data (1).



Figure 5: Loading and processing data (2).

* Purpose: Loads stock data, adds Google Trends data, scales selected columns, and splits data into training and testing sets.
* Details:
* yf.download: Downloads historical stock data from Yahoo Finance.
* fillna(method='ffill'): Fills any missing values using forward fill method.
* fetch\_google\_trends\_data: Gets data from Google Trends and saves it locally to avoid repeated downloads.
* Join: Combines Google Trends data with stock data.
* MinMaxScaler: Scales selected columns to a range between 0 and 1.
* split\_by\_date: Splits the data into testing and training sets using a ratio or an assigned date.

1. Displaying data in a custom table (From the task B.2):

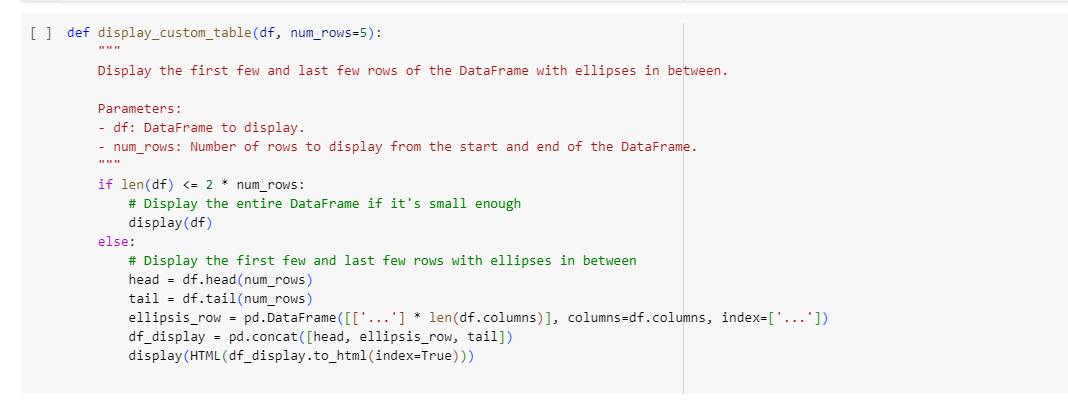


Figure 6: Displaying the data from csv file.

* We still use the same displaying data function just like B.2.

1. Displaying trend data:

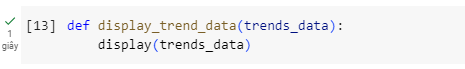


Figure 7: Displaying the trend data from csv file.

* Displays the DataFrame with Google Trends data using IPython's "display" method.

1. Model Creation (From the task B.4):



Figure 8: Code to create the model.

* We still use the same displaying data function just like B.4.

1. Experimentation with Different Configurations (From the task B.4):



Figure 9: Code to experiment with model (1).



Figure 10: Code to experiment with model (2).

* We still use the same displaying data function just like B.4.

1. Arima Predictions:

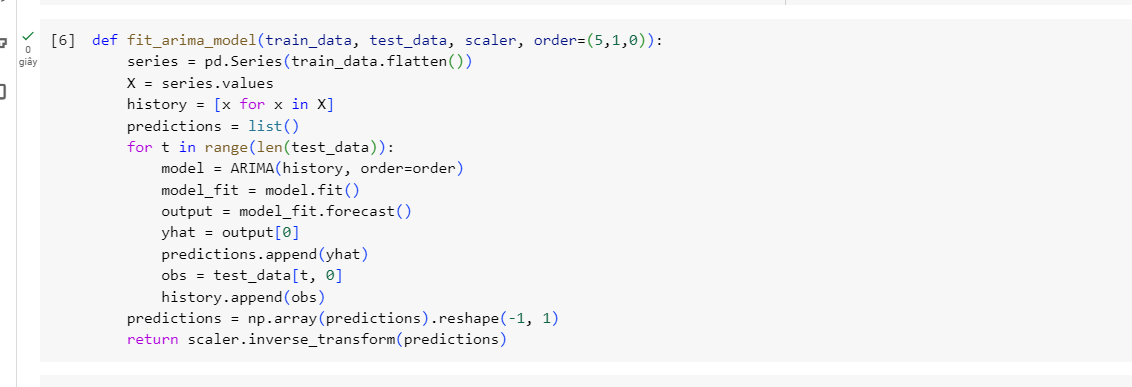


Figure 11: Code to fit arima model.

* The ARIMA (AutoRegressive Integrated Moving Average) model is a popular time series forecasting technique that makes predictions about future points in a series using historical data.
* Order parameters:
* P: The number of lag observations included in the model (autoregressive part).
* D: The number of difference analyses (integrated part) performed on the raw observations.
* Q: The size of the moving average window (moving average part).
* Implementation:
* Convert the training data into a pandas series.
* Flatten the series values to make it a 1D array.
* Initialize the values from the training data into a list called "history".
* Make an empty list called "predictions" that contains the expected values.
* Fit the "history" data to an ARIMA model.
* Applying the fitted model, predict the next value.
* Add the predicted value (yhat) to the list of predictions.
* For the following iteration, add the real observation (obs) from the test data to "history".
* Restructure and convert the prediction list to a numpy array.
* To inversely transform the predictions back to the original scale, use the scaler.

1. Ensemble Predictions (Combine with pytrends):

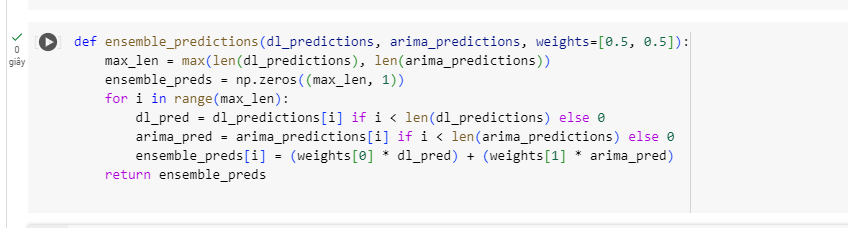


Figure 12: Code to create ensemble predictions (with pytrends).

* Return the combined predictions combining predictions from several models to enhance performance as a whole. This method makes use of the advantages of several models to provide a forecast that is more reliable and accurate.
* In order to record public interest and market attitude, which may be a good indicator of changes in stock prices, additional external data was included, such as Google Trends.

1. Main script run:



Figure 13: The script to run the code and the prediction (1).



Figure 14: The script to run the code and the prediction (2).

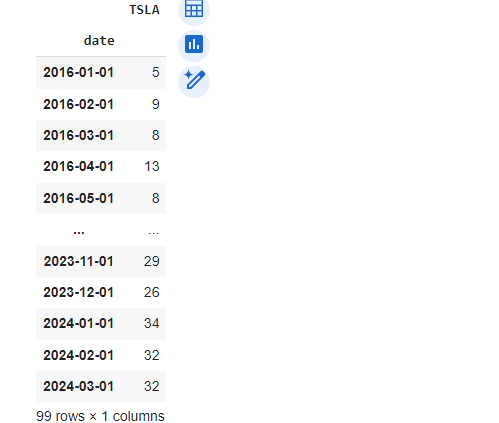


Figure 15: The pytrends data table.

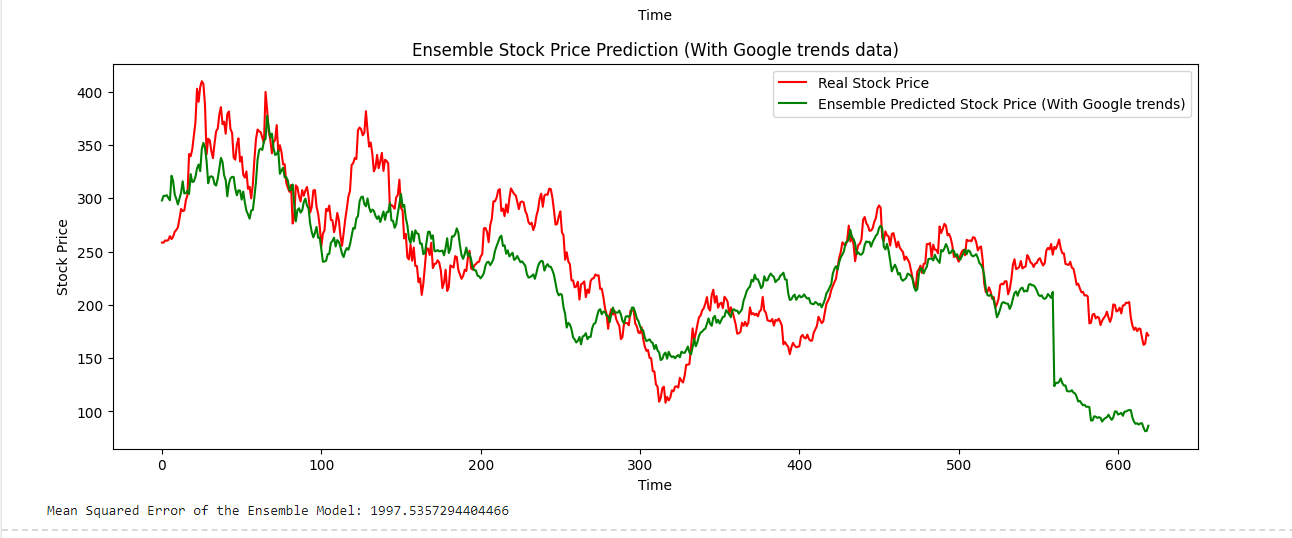


Figure 16: The ensemble predictions (with Google trends data).

1. References:

ADEP, V. (2021). *Google Trends using Python*. <https://www.kaggle.com/code/adepvenugopal/google-trends-using-python>

GUTIÉRREZ, J. L. R. (2022). Get Google Trends Data using Pytrends. <https://www.kaggle.com/code/luisresendiz/get-google-trends-data-using-pytrends>