**Time Series Stock Analysis**

**Francis Odo**

**Background**

Trading stocks in the stock market is a decision-based activity that relies heavily on current performance information and the prediction or forecast of future stock performance. Data is compiled daily for all active trading sessions Monday through Friday, monthly and yearly. Collected data consists of Opening Price, Lowest Price, Highest Price, Closing Price, Trading Volume and Stock Name. The information can be used for many reasons depending on the interest of participants in the stock market or anyone interested.

This exercise is only meant to demonstrate the application of machine learning techniques and not a real stock price prediction.

**Objective**

The focus of this analysis is on using statistical data and charts to identify trends in the performance of a selected stock in the dataset. The objective is to demonstrate the application of key aspects of data analysis techniques working with Time Series Stock Data collected over a period of five years. The analysis provides answers to key performance index questions using multiple algorithms such as Linear Regression and Neural Network.

**Question to which answers are offered**

* Trend - Is there an upward or downward trend?
* Pattern - Is there a pattern over time?
* Seasonality or Stationarity – If mean and variance are constant or joint distribution remain the same?

**Development Environment** Python Pandas Matplotlib Jupyter Notebook TensorFlow Data file (aal\_stocks\_5yr.csv) Database – PostgresSQL

**Exploratory and Preprocessing**

The first step in this analysis is to inspect the data to be analyzed, in other words, get familiar with the structure and characteristics of the data. Having a good understanding of the data is crucial to determining how to proceed with the analysis of the data.

Our Time Series Stock Market Data consists of seven columns and sixty thousand rows. Due to processing resource challenges the data is significantly reduced to a three thousand and seventy-seven rows.

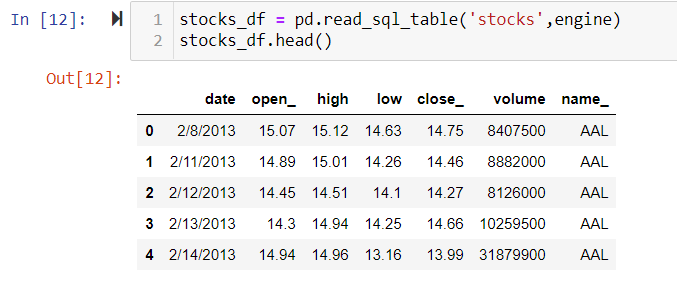
Every raw data consists of incomplete or null information, which does not make any sense to the computer in terms of analysis and computation. In the process, the data will have to be cleaned. Following that cleaning procedure, the NAN, NAs and Nulls are dropped from the data. Further, the date format was changed and set as index.

Depending on the algorithm of choice, there are factors that will need to be considered and addressed when working with Time Series data. These factors often present themselves as you extract and transform the data. Python Pandas provides a wide choice of functions and methods to remove, rearrange as necessary.

Column objects or features of this data are: Date – Date of trading Open – Opening trading stock price for the day High – The highest trading stock price of the day Low – The lowest trading stock price of the day Close – The closing stock price of the day Volume – The total stock trading volume of the day Name – The Ticker symbol for the company stock

In order to ensure our data sample is not overly or extremely skewed, it needs to be normalized and scaled. This step in the process converts all input data to what the computer can interpret, zeros and ones. Furthermore, the data is standardized using the Python Pandas StandardScaler.

The closing stock price of the day “close\_” is used as Target in this analysis. All other variable may be used as input feature as needed.



**Database Preparation**

Prepare the RDS system to store data. The PostgresSQL database is used in this exercise. AWS will work just as well. However, the setup procedure is different, but the principal logic is similar. The steps highlighted are common to either choice. 1. Set up the connection in Jupyter Notebook (Protocol, User, Password, Location, Port and db name). Also, load the libraries and dependencies as in “ with engine.connect() as connection”.

2. Define the connection (path and connection engine)

3. Create a database named “Time\_Series” in PostgresSQL DB

4. Create a table for the data from the csv file

5. Import or load the data into Postgres (Note: Data was originally downloaded from Kaggle.com)

6. Install “!pip install ipython-sql”, load the dependencies for sql “from sqlalchemy import create\_engine”

**Code Plan**

1. Read in the data in Jupyter Notebook using pd.read\_sql\_table('aal\_stocks',engine)
2. Format date aal\_stocks\_df["date\_"] = (pd.to\_datetime(aal\_stocks\_df['date\_'].values))
3. Check to see if there are null values or missing data
4. Generate categorical variable list
5. Check the number of unique values in each column
6. Create a OneHotEncoder instance
7. Merge one-hot encoded features and drop the originals
8. Drop Nas and nulls
9. Standardize data with StandardScaler
10. Normalise the data to be processed
11. Split into features and target arrays (X and Y, train and test)
12. Create a StandardScaler instance
13. Scale the data
14. Create a Linear Regression model
15. Fit the model
16. Predict "y"
17. Print the coefficient of determination, Intercept, Slope and the predicted output

**Visualization**

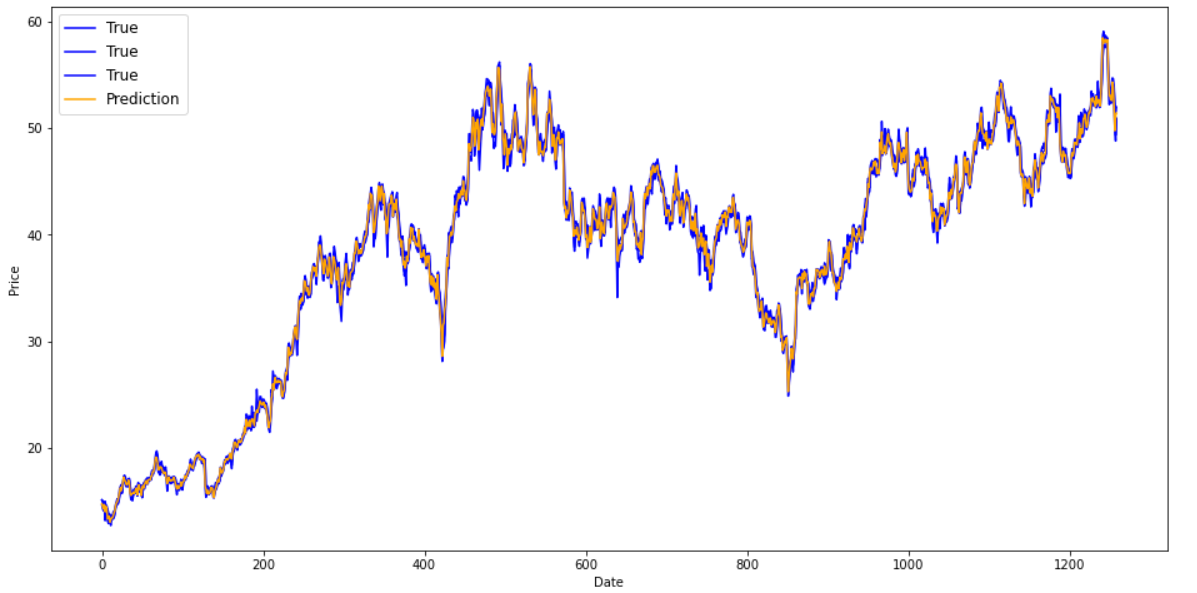


Fig. *Graphical representation of the input features and the predicted values*.

**Summary**

Regression models describes the relationship between all sample variables by fitting a line to the data. Linear Regression uses a straight line. Regression allows the estimation of how dependent variable(s) change as the independent variable(s) changes. Linear Regression estimates the relationship between the two quantitative variables. Linear Regression model algorithm as applied in this exercise assumes:

* Data is in the form of a time series, which indicates that data exists over a continuous time interval with equal spacing between measurements.
* Variables with normal distribution were compared by means of parametric tests i.e. value of parameters.
* The relationship between the independent and dependent variable is linear.
* The size of the error in our prediction doesn’t change significantly across the values of the independent variable.
* There are no hidden relationships among observations.

**Risks**

The prediction in this exercise is not a real-life prediction. It is a demonstration of how machine learning can be applied to time series data analysis. There are other few things needed for this to be complete. For example, the application needs to allow:

1. Choice of stock symbol to be predicted at the user interface
2. Timeframe for which test and prediction covers
3. Mobile application
4. Most recent and current time series data