Evolution of Time Series Analytics

A Wholistic approach towards Stock Market Prediction

| 1970 | 1974 | 1986 | 1997 | 2014 | 2018 |

HOLT's WINTER

DEEP LEARNING INTRODUCED

GRU INTRODUCED

ARIMA INTRODUCED

LSTM INTRODUCED

FB RPOPHET
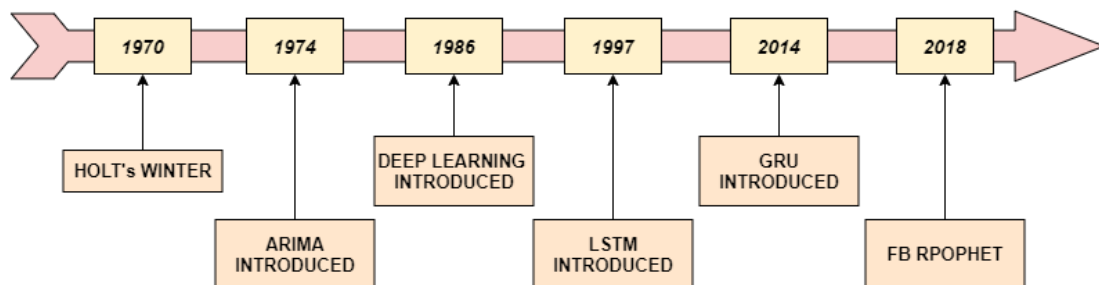
Contents

Introduction about Time Series Analytics

Abstract

This project conceptualizes the evolution of time series from 1980 to today's world, from a time where forecasts were done using pen and paper to using A.I. Time Series Evolution potrays how statisticians and researchers came all the way from performing forecasting using linear regression to using state of the art deep learning models. This paper has a brief introduction about the essence of time series analytics, its components, classical statistical smoothing methods such as Holt's Winter & SES, regression based models such as linear & quadratic trend, general forecasting models such as ARIMA & SARIMA to Advanced models such as LSTM's and FB Prophet.
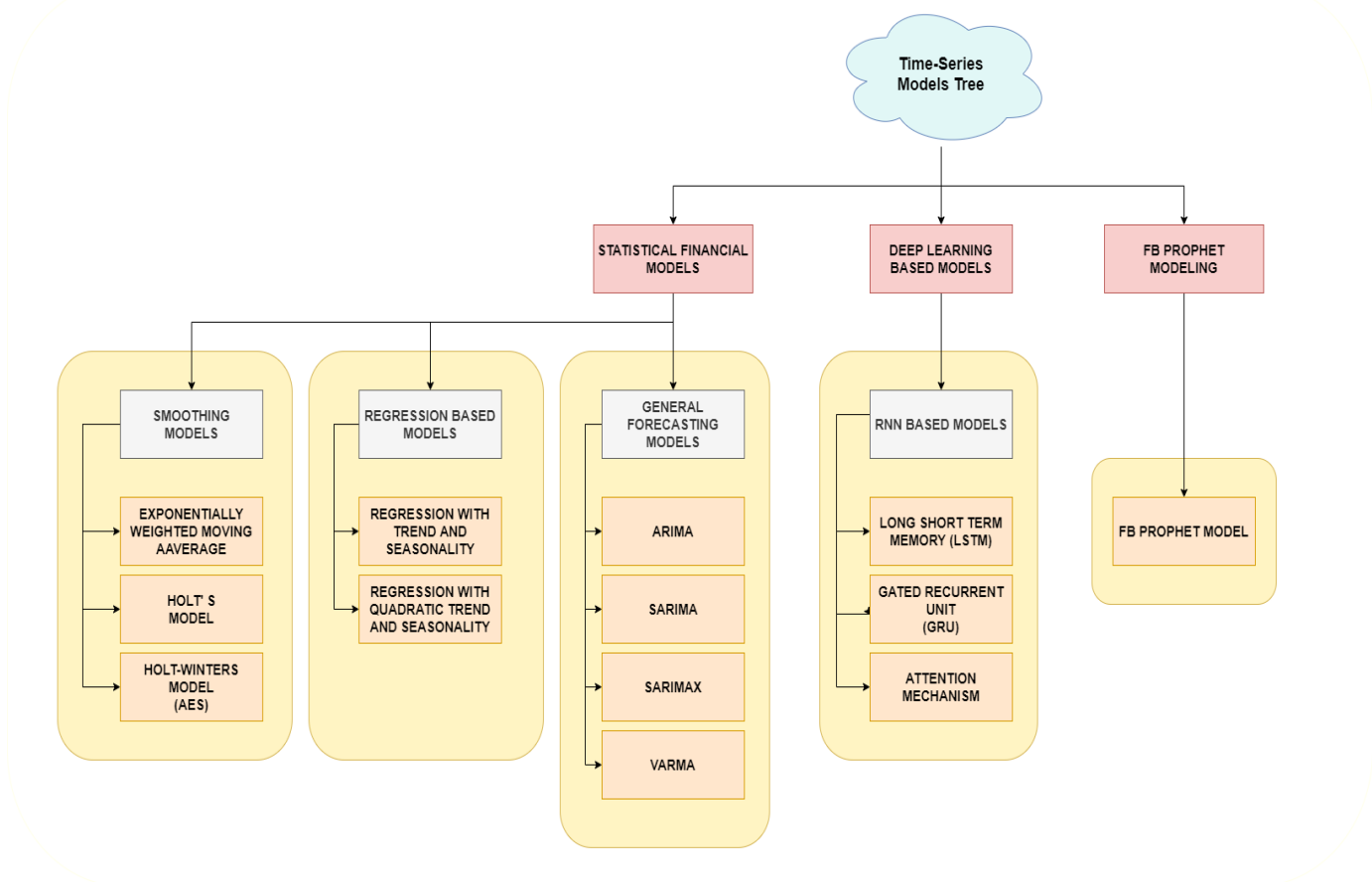
What is Time Series?

Time Series is a 1-Dimensional data stored with respect to Time. It is a consecutive sequence of data that is collected throughout series of days/months/years. For example, Weather data that is collected form last ten years or Monthly sales data for Starbucks Cafe. Over here the granularity of weather data is daily while the Starbucks data is monthly. Hence granularity plays a vital role as it shapes a level at which our data is stored.

What is Time Series Analytics?

Now let us say we have got sequence data. So, we wonder what all analysis could be performed with it. Time series data can be used mainly to plot trends, seasonality. Trends and seasonality help us to visualize whether the past sales have grown or receded also which month of the year are we expecting spike in sales or such. This gives an important concept of forecasting. Forecasting means collecting past data values, performing some

mathematical computations and provide some forecasts just like what a prophet does.Another world wide usage of time series analytics is Anomaly detection. We can predict when a next anomaly is going to occur, thereby saving companies thousands of dollars.

What is Successful Time Series Forecasting?
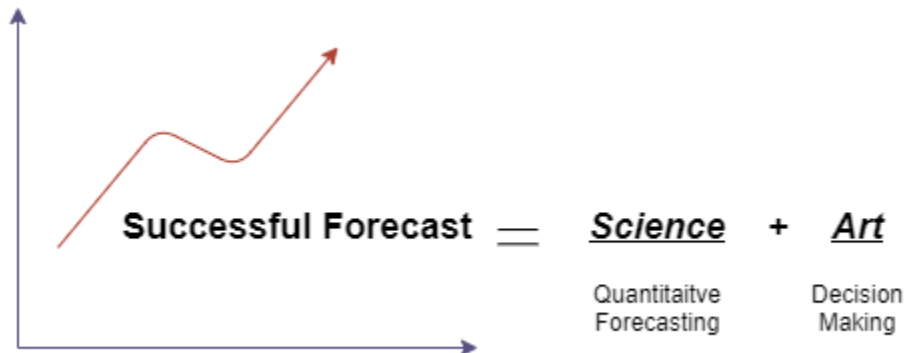
We often come across job designations such as financial analyst, quantitative analyst. Both job titles are a high paying and experience-based positions. The question arises what makes these jobs so demanding as well as a pivotal part of a corporate infrastructure. The reason is that successful forecast's is a combination of solid mathematical skills as well as the art of decision making which is developed after years and years of industry knowledge and experience. Hence these professionals are SME's of time series analytics.

Time Series Components
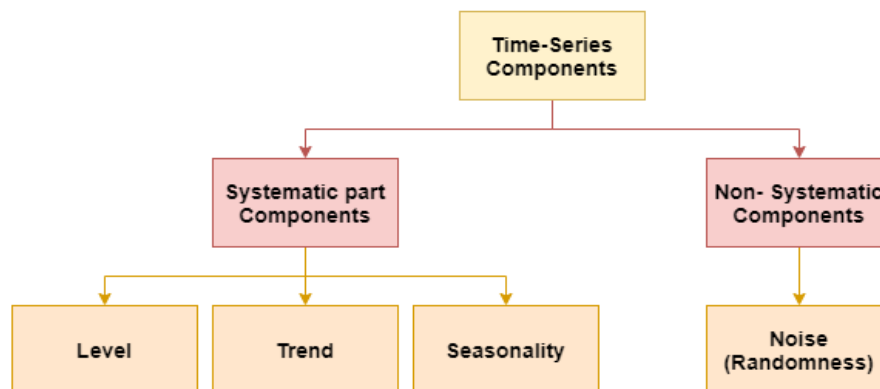
There are 4 main components and they can be implemented across 2 major divisions
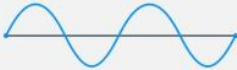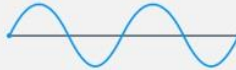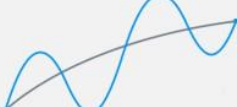
- **Level**: The average value of the series.
- **Trend**: Change in the values. Ex Upward trend (Positive) | downward trend (Negative)

- **Seasonality**: Short Term cyclical behavior observed in data. Ex if you have an ice cream shop, it is expected that you will have higher sales during the summer quarter than the winter quarter and this pattern will follow every year.
- **Noise**: Random variations that are not accounted generally. Ex if you have a goods manufacturing company, suppose for 2-3 days you have power outages. Then these days are called noise as the power outages recurring every week is less. It can be thought of as outliers in data.

Additive Components: Level + Seasonality + Trend + Noise

Multiplicative components: Level X Seasonality X Trend X Noise

Forecasting Methods Foundation: Forecasting Methods try to isolate the systematic part or noise.

	NONSEASONAL	ADDITIVE SEASONAL	MULTIPLICATIVE SEASONAL
Constant Level	(Simple) NN	NA	NM
Linear Trend	(HOLT) LN	LA	(WINTERS) LM
Damped Trend (0.95)	DN	DA	DM
Exponential Trend (1.05)	EN	EA	EM

Source : https://medium.com/@devtodev/how-to-find-and-manage-seasonality-of-your-game-project-77577d4ae7f8

The above image plots various combinations of components and is displays in a graphical format.

Introduction about U.S. Stock Market

What is bear and bull market?
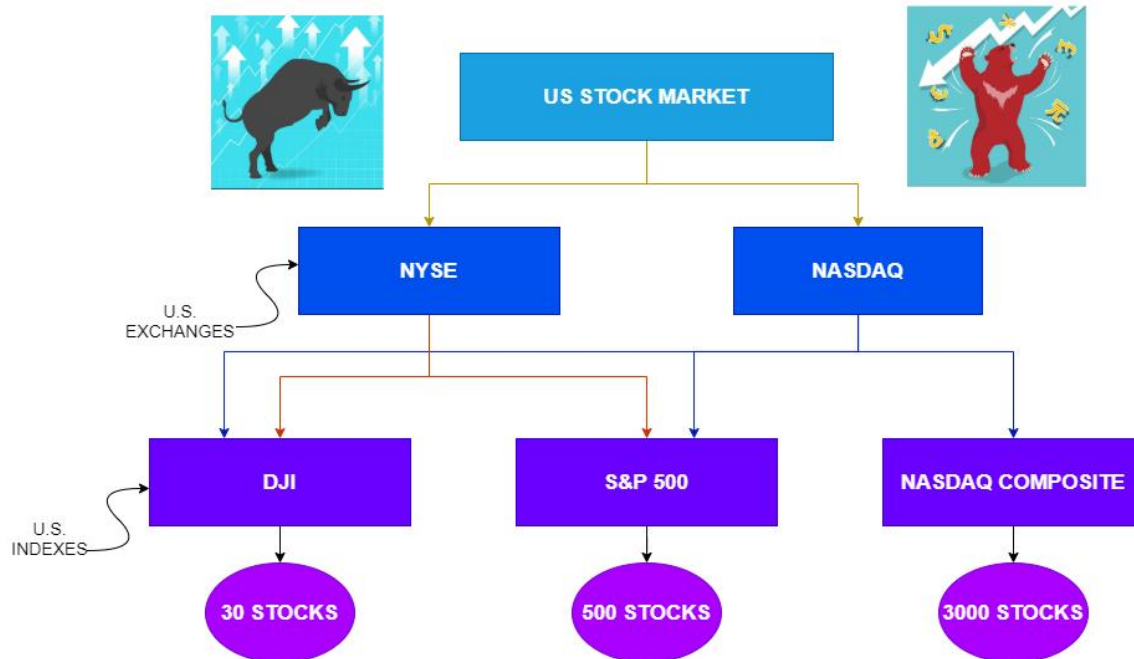
US Stock Market: A stock market, equity market or capital market is the collection of buyers and stock sellers (also referred to as shares) containing ownership claims on companies. They can include securities listed on a public stock exchange, as well as private-traded stocks such as private-sector shares offered to investors through equity crowdfunding platforms. Investment in the stock market is most frequently carried out through stock brokerages and platforms for electronic commerce.

NASDAQ: The NASDAQ Composite is a common stock market index and related securities listed on the Nasdaq stock exchange. This is one of the three most tracked indexes in US capital markets along with the Dow Jones Industrial Average and S&P 500. The composition of the NASDAQ composite is heavily weighted towards IT companies.

NYSE: The New York Stock Exchange is an American stock exchange headquartered at New York. As of February 2018, it is by far the world's largest stock exchange by market capitalization of its listed companies at US$ 30.1 trillion. The average daily trading volume in 2013 was around US$ 169 billion.

DJI: The Dow Jones Industrial Average Dow Jones is a capital market index tracking the investment performance of 30 major corporations listed on US stock exchanges. The index value is the amount of the price of one stock share for each component company separated by a which factor for the index.

S&P 500: The S&P 500, or simply the S&P is a stock market index that measures the stock performance of 500 major companies listed on US stock exchanges. It is one of the most widely tracked share indices and it is considered by many to be one of the best US stock market representations. The index's average annual total return, including dividends, has been 9.8 percent since its establishment in 1926. however, there have been many years in which the index has fallen by more than 30 percent.

Let us Consider a DJI listed company to perform times series analytics on

I have selected American Express as my main focus. The American Express Company, commonly known as Amex, is an American multinational corporation of financial services with headquarters in New York City, New York. The company was founded in 1850 and is one of the Dow Jones Industrial Average's 30 major players. The company is best known for its charging card, credit card and traveler's cheque service.

In 2016, credit cards using the American Express network (AXP) summed up for up to 22.9 % of the United States' overall number of credit card transactions. The organization has 114.4 million cards in operation as of December 31, 2019, with 54.7 million cards in use in the United States, each with an estimated annual cost of 19.972 dollars. In 2017, Forbes listed American Express as the 23rd most valuable company in the world (and the highest in financial services), estimating that the brand is worth US$ 24.5 billion. In 2020, Fortune magazine rated American Express as number 9 in its Top 100 Companies to Work for Fortune list in 2020 based on a satisfaction survey of its employees.

Source: https://www.americanexpress.com/

Source: https://www.americanexpress.com/

Environment Selection | Data Partition | EDA

Deciding Testing | Validation | Forecasting split
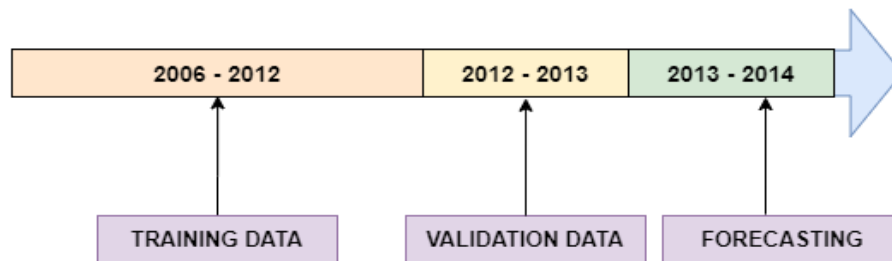
I have considered data from 2006 to 2013. The train | validation split is given above.

Language and Environment Used to Create forecasting

I have used R as well as Python to perform time series analysis. R was used to perform statistical analysis models while Python for Deep Learning and FB Prophet models. R performs better for statistical analysis as it was created for that purpose only, while Python outperforms R in Deep Learning functionalities. For R, R studio was used, for Python, Google GPU was used in Colab environment.

Image Source : https://medium.com/@senthilnathangautham/colab-gcp-compute-how-to-link-them-together-98747e8d940e

Data Loading and EDA

Date	Open	High	Low	Close	Volume	Name
1/3/2006	51.7	52.58	51.05	52.58	7825700	AXP
1/4/2006	52.44	52.57	51.81	51.95	5729400	AXP
1/5/2006	51.9	52.51	51.9	52.5	3926000	AXP
1/6/2006	52.64	52.87	52.28	52.68	4137800	AXP
1/7/2006	52.82	53.99	52.82	53.99	6093300	AXP

Above given image is a snippet of the AXP data in Excel.

Data has 7 rows:

- **Open**: Opening price of AXP Stock
- **High**: Highest value of AXP Stock throughput the day.

- **Low**: Lowest value of AXP Stock throughput the day.
- **Volume**: Number of AXP Stocks traded that day.
- **Close**: Closing value of AXP stock for a day.
- **Date**: Date
- **Name**: Abbreviation of American Express stock.

View of the Entire Data Set

American Express Time Series Component

Above given is a plot of high column for AXP throughout the years. We see that there is a positive trend after the great recession from 2007 – 2009. But towards the end of 2012 we see the stock starts to recede again. We know that stock market is highly volatile as well as a stock price depend on various factors such as companies' image, their new launches, existing revenue cycles, external factors etc. But what we can predict is a trend an interval (95%) of which the values will lie in. Upon pre analyzing we can say there are various rises and declines. So, question arises how we are going to predict it. Like I have said earlier forecasting is a combination of statistical models and experience. Below given is a brief explanation about which models have been used in past and the forecasting capabilities they bring along with them. Towards the end I have given more weightage on how state of the art FB Prophet and Deep Learning technologies are perceiving time series data and how do they have an upper hand when compared to traditional statistical models.

Data Visualization & Data Pre-processing

OHLC Charts

Open-high-low-close charts (or OHLC charts) are used as a trading method for tracking and evaluating price shifts for shares, currencies, stocks, bonds, commodities, etc. over time. OHLC charts are useful to interpret the market's day-to-day feeling and to forecast any potential price changes through the trends produced.
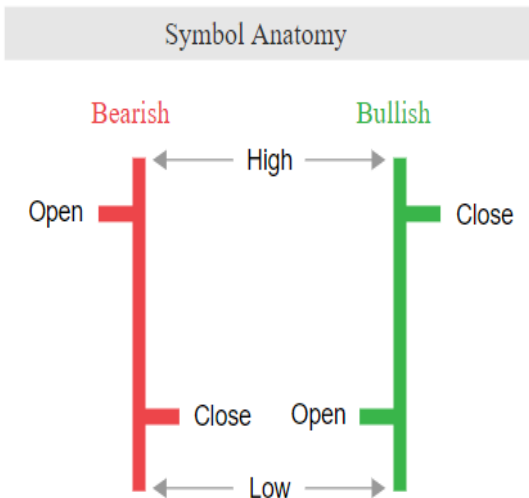
Source : https://datavizcatalogue.com/methods/OHLC_chart.html
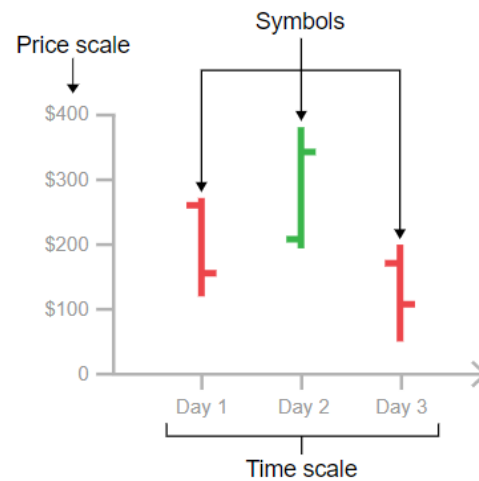
Source : https://datavizcatalogue.com/methods/OHLC_chart.html
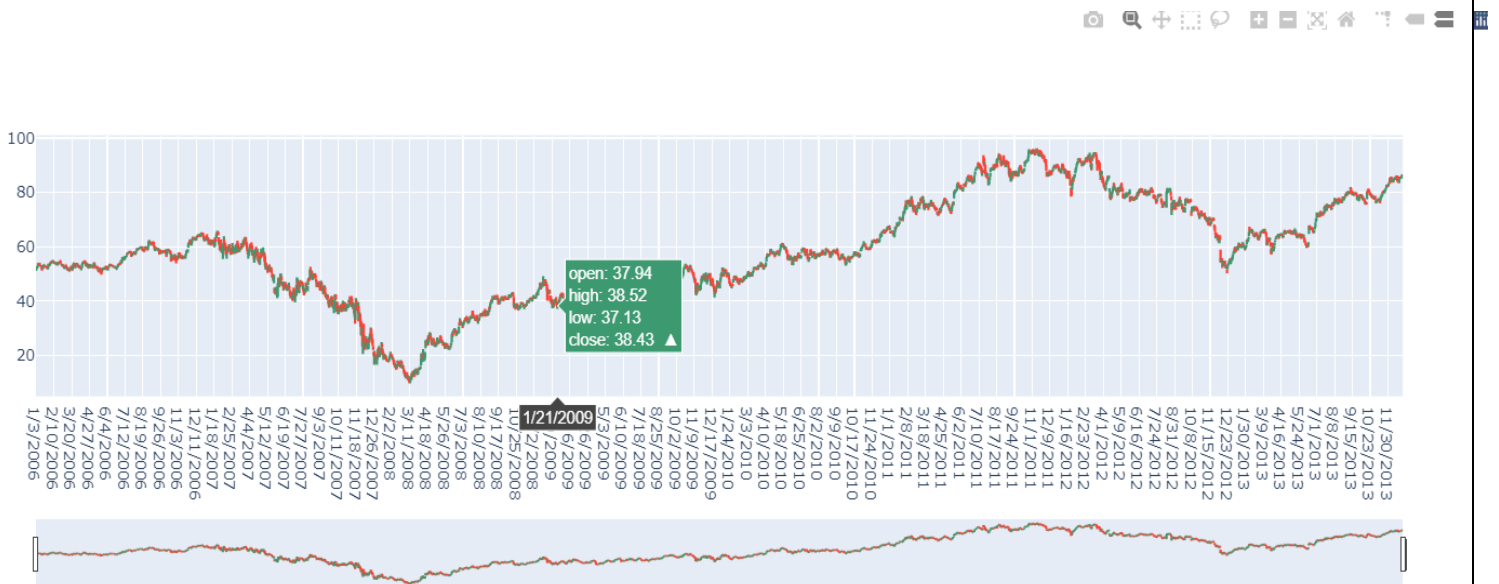
OHLC Chart of entire AXP Stock. Above given image is an interactive plot which can be visualized in Python.

Check for Stationarity

Stationarity means that the data has a constant mean and variance throughout. It signifies that it will be difficult for the statistical models to isolate trend and seasonality.

Source : https://towardsdatascience.com/stationarity-in-time-series-analysis-90c94f27322
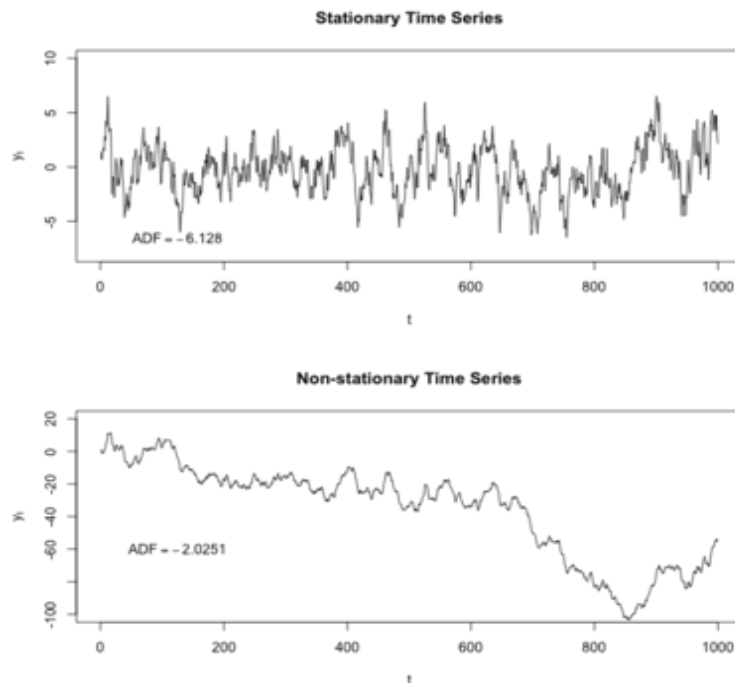
We use Augmented Dickey – Fuller Test to check for stationarity of Data

Augmented Dickey – Fuller Test (ADF) checks the null hypothesis in statistics and econometrics that a unit root is present in a sample of a time series..

Null Hypothesis: (p < 0.50) The data is Stationary

Alternative Hypothesis: (p > 0.50) The data is Non - Stationary

```
print('Augmented Dickey-Fuller Test on AMX Data')
testresult = pd.Series(dftest[0:4],index=['ADF test statistic','p-value','# lags used','# observations'])
for key,val in dftest[4].items():
    testresult[f'critical value ({key})']=val
print(testresult)
```

```
Augmented Dickey-Fuller Test on AMX Data
ADF test statistic        -0.836116
p-value                    0.808282
# lags used                2.000000
# observations          2917.000000
critical value (1%)       -3.432594
critical value (5%)       -2.862531
critical value (10%)      -2.567298
dtype: float64
```

As we can see the p value >0.05 -> The data is not stationary.

Data Pre-Processing

- **Splitting Train and test**
    - Train -> 2006 – 2012
    - Validation -> 2012 – 2013
- **Checking for Null Values**
- **Scaling values from 0 to 1 only for training dataset**

```
sc = MinMaxScaler(feature_range=(0,1))
training_set_scaled = sc.fit_transform(training_set)
sc

MinMaxScaler(copy=True, feature_range=(0, 1))
```

```
[13] dataset.isnull()
```

Date	Open	High	Low	Close	Volume	Name
2006-01-03	False	False	False	False	False	False
2006-01-04	False	False	False	False	False	False
2006-01-05	False	False	False	False	False	False
2006-01-06	False	False	False	False	False	False
2006-01-07	False	False	False	False	False	False

ETS Decomposition

ETS Decomposition breaks the time serie into 4 components to have a better visualization about the Time series Components.
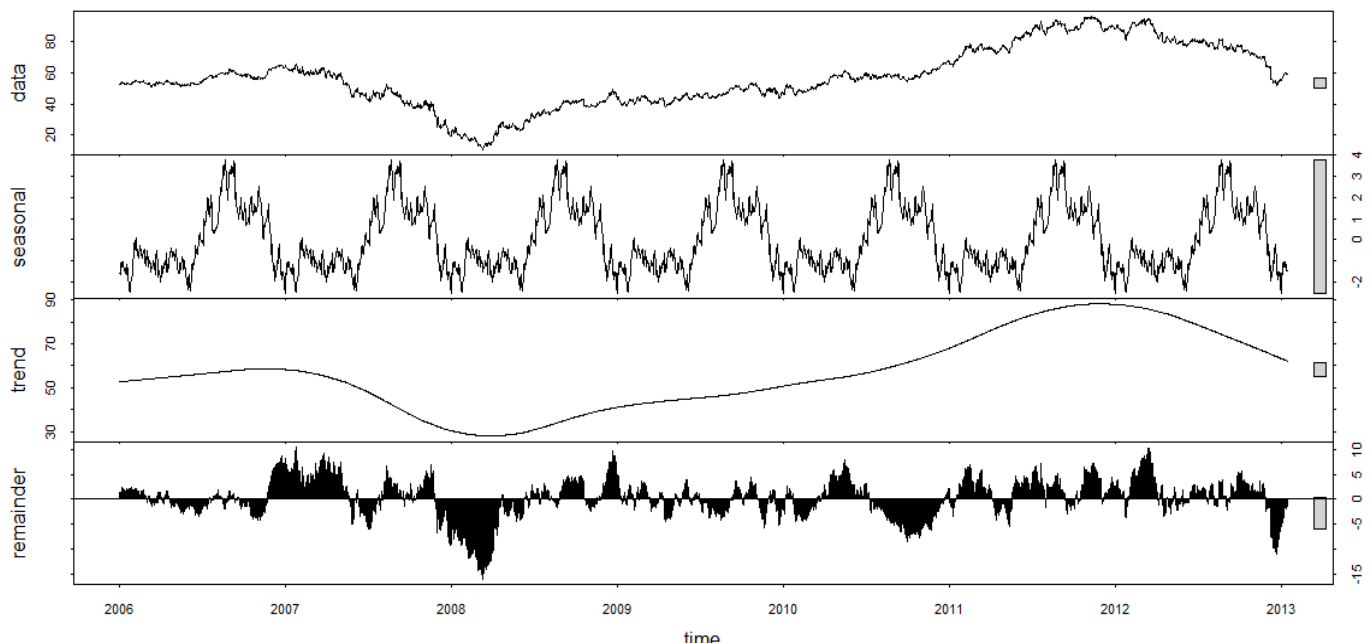
- **Data**: The actual data across time.
- **Seasonal**: Plots the seasonal component in data.
- **Trend**: The upward or downward motion of Time series.
- **Remainder**: Represents the residuals(errors) in the data.

We see that the data has upward as well as downward trend and there is some seasonal component present. To have a better understanding about seasonality we plot the autocorrelation chart.

Autocorrelation

Autocorrelation is one of the most important factor of Time series. It means subtracting the current data point from its yesterday's value. It tells if todays and yesterday's values are correlated or not. If we subtract form yesterday's time stamp it is called as lag=1, if we subtract it from previous 2 days value data it is called as lag 2. Ex. If you have monthly sales data and you plot it for 12 lags and while plotting you see that there is high auto correlation in 12th lag. It means that sales of last year same month are highly auto correlated. It is natural, suppose you sell ice cream, it is likely that for every summer the sales will be high.

Autocorrelation forAmerican Express
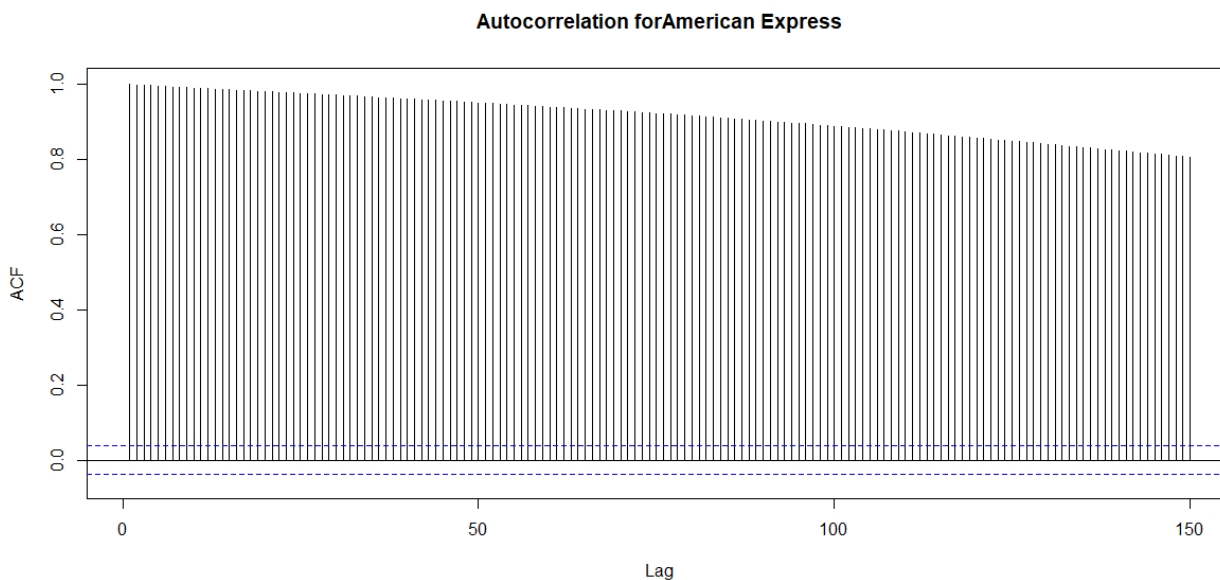
Current Days stock price is high related on yesterdays closing price. This can also be seen on the above ACF Plot. Earlier lags prove that the data has trend while the lags towards the end provide seasonanlity.

Accuracy Measures

- **RMSE**: Root Mean Square Error (RMSE) is the standard deviation of the residuals (prediction errors).
- **MAE**: Mean absolute error (MAE) is a measure of errors between paired observations expressing the same phenomenon.
- **MPE**: In statistics, the mean percentage error (MPE) is the average computed of percentage of errors by which forecasts of a differ.
- **MAPE**: It measures the overall accuracy as a percentage, and it can be evaluated as the average absolute percent error for each time period divided by actual values.

Base Model
Naïve Model & Seasonal Naïve Model

Naïve Model calculates the mean of the last point of the training set and displays the result throughout while the seasonal naïve considers the seasonal component as well. This model is considered as the base model as it considers last point of the training data as in time series the most recent points contribute most during forecasting than the earlier data points.

```
df.naive.pred <- naive(train.ts, h = nvalid)
df.snaive.pred <- snaive(train.ts, h = nvalid)
```
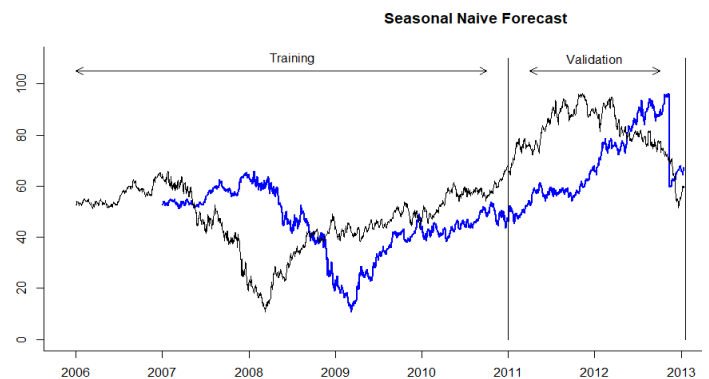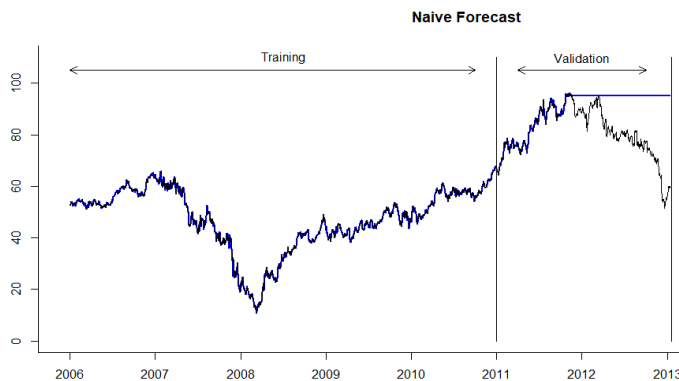


```
> round(accuracy(df.naive.pred$mean, valid.ts), 3)
              ME    RMSE     MAE     MPE    MAPE  ACF1 Theil's U
Test set -15.451 18.537 15.457 -21.748 21.754 0.988    22.951
> round(accuracy(df.snaive.pred$mean, valid.ts), 3)
             ME    RMSE     MAE    MPE   MAPE  ACF1 Theil's U
Test set 1.778 15.367 13.369 0.476 16.632 0.984    15.097
>
```

As we can see both the models have high RMSE value 18.53 & 13.36 respectively. Seasonal naive is doing better than base naïve. Naïve accuracy can be set as a baseline accuracy. It means that this is the simplest model possible and it acts as a reference to other models to see if they are performing better than the baseline RMSE accuracy.

Smoothing Models

```
                    ┌─────────────────┐
                    │   Exponential   │
                    │   Smoothing     │
                    └────────┬────────┘
              ┌──────────────┴──────────────┐
              ▼                              ▼
    ┌─────────────────┐          ┌─────────────────┐
    │     Simple      │          │    Advanced     │
    │   Exponential   │          │   Exponential   │
    │   Smoothing     │          │   Smoothing     │
    └─────────────────┘          └────────┬────────┘
                                 ┌─────────┴─────────┐
                                 ▼                   ▼
                         ┌──────────────┐    ┌──────────────┐
                         │ Holt's Model │    │ Holt's Winter│
                         │              │    │    Model     │
                         └──────────────┘    └──────────────┘
```
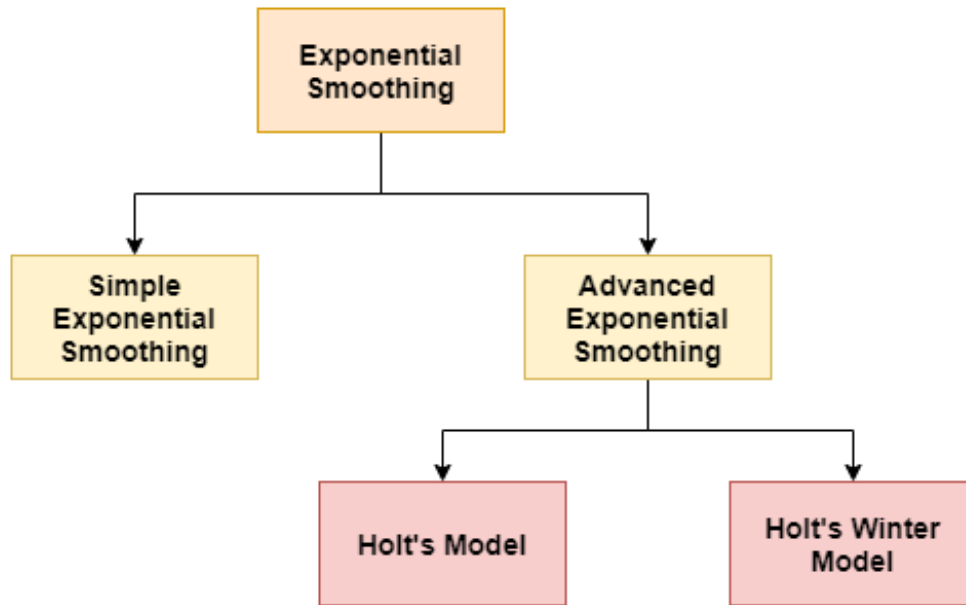
SES**:** Take a weighted average of all past historical data periods such that weights decrease exponentially in the past. Often more weight is given to recent historical data periods, but do not neglect older historical data periods entirely. "a" is a smoothing constant.

- *"a" closer to 1 signifies more influence on recent values*
- *"a" closer to 0 signifies more influence on past values*

Choice of alpha depends on the statistical choice looking at the data and the type of data, for example stock data has more weightage to the recent data points as they influence the values most.

SES Additive Model with Optimal Smoothing Parameters
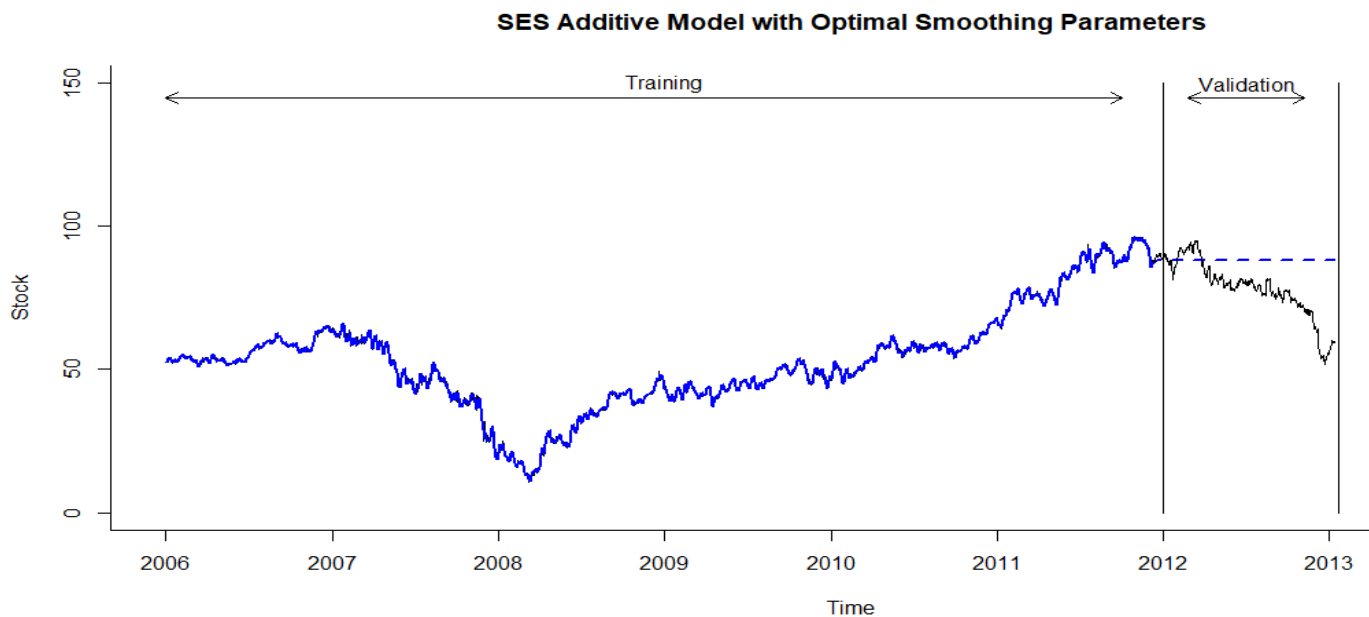
Holt's Model: For time series containing trend, Holt's (double-exponential) smoothing is used. Idea is to increase simple exponential smoothing (SES) by capturing a trend variable.

Here there are 2 parameters

- α = smoothing constant for exponential smoothing ($0 \le a \le 1$)
- β = smoothing constant for trend estimate ($0 \le b \le 1$)

Holts has 2 models

- *Additive model: Forecast = Level (L_t) + Trend (T_t)*
- *Multiplicative model: Forecast = Level (L_t) × Trend (T_t)*

Holt's Multiplicative Model with Optimal Smoothing Parameters
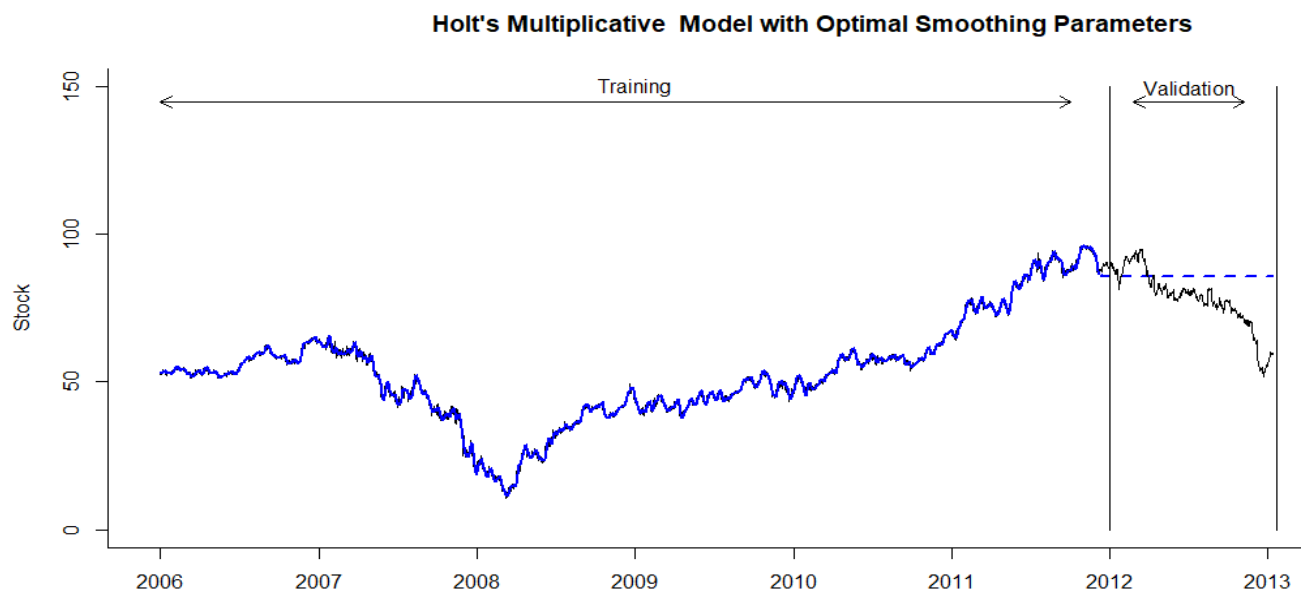
Holt's winter Model: Winter's (Holt-Winter's) model is used for time series with trend and seasonality. Idea is to increase the model of Holt by capturing a seasonal portion as well as the trend components.

- α = smoothing constant for exponential smoothing ($0 \leq a \leq 1$)
- β = smoothing constant for trend ($0 \leq b \leq 1$)
- γ = smoothing constant for seasonality ($0 \leq \gamma \leq 1$)
- *Winter's multiplicative model: Forecast = [Level (Lt) + Trend(Tt)]×Seasonal Component (S_t)*

Holt's Winter Multiplicative Model with Optimal Smoothing Parameters
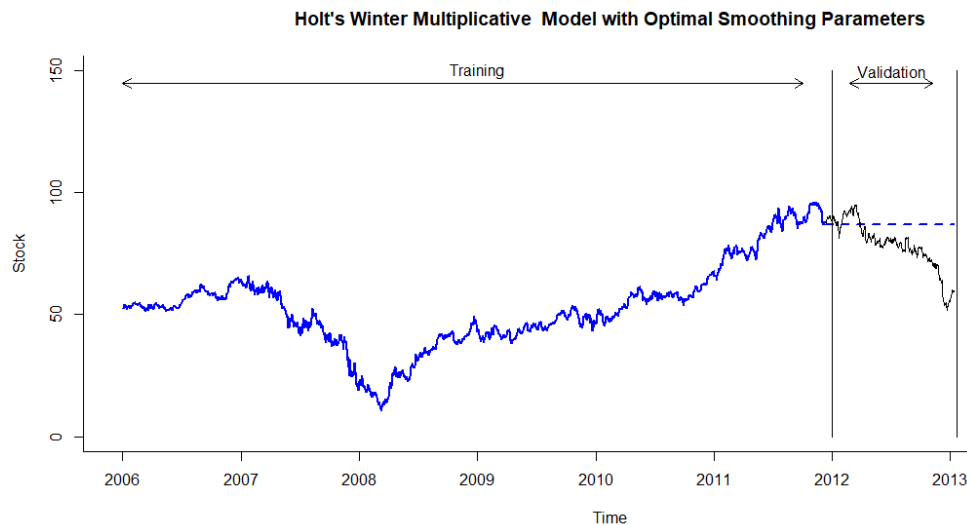
```
> round(accuracy(ses.orig.pred, valid.ts), 3)
                ME    RMSE    MAE     MPE    MAPE  MASE  ACF1 Theil's U
Training set  0.016  1.110  0.828   0.003  1.857 0.047 0.622        NA
Test set     -9.395 13.768 10.640 -14.062 15.411 0.605 0.990    17.582
> round(accuracy(h.AAN.opt.pred, valid.ts), 3)
                ME    RMSE    MAE     MPE    MAPE  MASE  ACF1 Theil's U
Training set  0.015  0.812  0.579   0.002  1.298 0.033 0.072        NA
Test set     -8.223 12.985  9.984 -12.543 14.460 0.568 0.990    16.683
> round(accuracy(hw.ZZZ.pred, valid.ts), 3)
                ME    RMSE    MAE     MPE    MAPE  MASE  ACF1 Theil's U
Training set  0.016  0.812  0.579   0.001  1.298 0.033 0.081        NA
Test set     -8.290 13.028 10.020 -12.631 14.513 0.570 0.990    16.733
```

We see RMSE marginally differs in the three smoothing models. Also, the best smoothing model is holt's multiplicative (RMSE: 12.985) which is better than the Naive model.

Regression Based Models

Regression analysis is a type of predictive modelling technique that explores the relationship between a dependent variable(x) and an independent variable (y). This technique is used to forecast, model time series and evaluate the causal effect relation between the variables. These models are effective to handle seasonal models as well.

There are 4 models performed

- **Regression Model with Linear Trend |** $y_t = b_0 + b_1 t + e$
- **Regression Model with Quadratic (Polynomial) Trend |** $y_t = b_0 + b_1 t + b_2 t^2 + e$
- **Regression Model with Seasonality |** $y_t = b_0 + b_1 D_2 + b_2 D_3 + ... + b_{11} D_{12} + e$
- **Regression Model with Quadratic Trend and Seasonality|** $y_t = b_0 + b_1 D_2 + b_2 D_3 + ... + b_{11} D_{12} + e$
  (Where D is day or month)
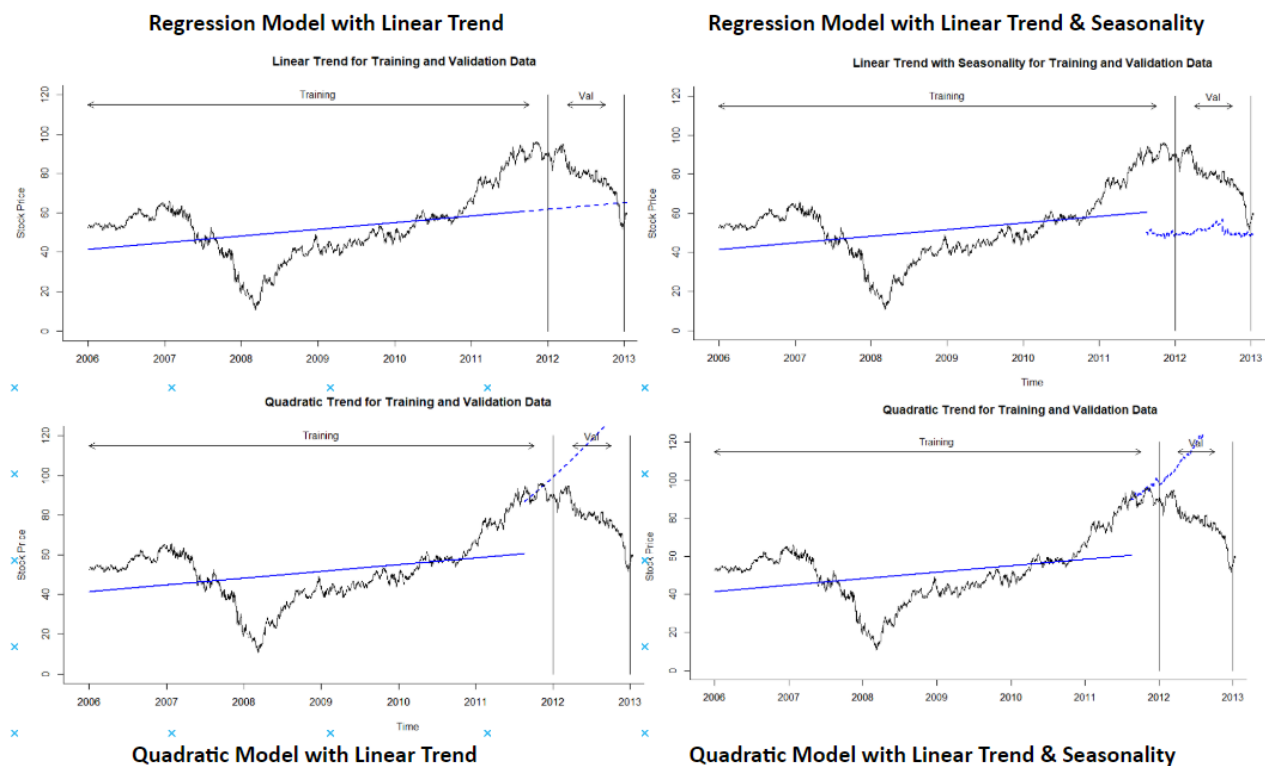

```
> round(accuracy(train.lin.pred, valid.ts),3)
                ME    RMSE    MAE     MPE   MAPE  MASE  ACF1 Theil's U
Training set  0.000 14.298 11.708 -12.653 30.705 0.707 0.997        NA
Test set     18.728 22.055 20.071  21.239 23.660 1.212 0.991    21.003
> round(accuracy(train.quad.pred, valid.ts),3)
                 ME    RMSE    MAE     MPE   MAPE  MASE  ACF1 Theil's U
Training set   0.000  8.142  6.049  -5.903 16.770 0.365 0.994        NA
Test set     -30.525 39.693 31.112 -42.604 43.234 1.879 0.993    49.475
> round(accuracy(train.season.pred, valid.ts),3)
                ME    RMSE    MAE     MPE  MAPE  MASE  ACF1 Theil's U
Training set  0.000 15.181 11.595 -13.606 30.5 0.700 0.997        NA
Test set     31.166 32.958 31.166  37.000 37.0 1.882 0.990    31.845
> round(accuracy(train.trend.season.pred, valid.ts),3)
                 ME    RMSE    MAE     MPE   MAPE  MASE  ACF1 Theil's U
Training set   0.000  7.919  5.953  -5.431 16.080 0.360 0.995        NA
Test set     -31.766 40.958 31.933 -44.236 44.413 1.929 0.994    50.941
```

We see all the regression models are under fitted. Moreover, they are performing far worse than the baseline model.

ARIMA Models
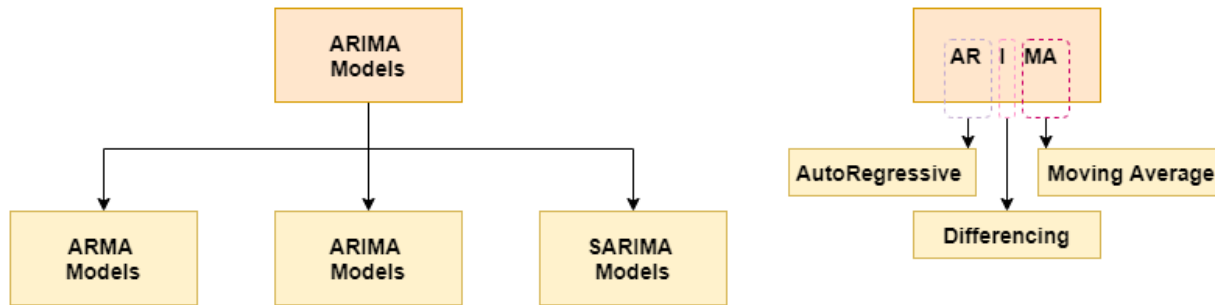
Autoregressive Integrated Moving Average (ARIMA) are popular models in time series forecasting. They are also called as Box-Jenkins methodology. They can easily plot – level (stationary), trend, and seasonality – or a combination of these components. It may include up to 3 components & 6 parameters depending on the nature of our data.

There are 6 sub-components , made up by combining Trend and Seasonality with 3 main components.

- **Autoregressive (p | P):** $y_t = b_0 + b_1 y_{t-1} + b_2 y_{t-2} \ldots + b_p y_{t-p} + e_t$
  It identifies the relationship autoregressive between lags, p (p = 0, 1, 2, …) gives which is the best number of lags, it is just like ACF function discussed earlier.
- **Moving Average (q | Q):** $y_t = c + e_t + q_1 e_{t-1} + q_2 e_{t-2} \ldots + q_q e_{t-q}$
  [*c = means & e = error*] It models past residuals and errors in a model using autocorrelation lags
- **Integrated (d | D):** It is called as differencing part. it means how many times we must difference the model to remove the trend. Our goal is to make nonstationary data stationary.

There are 3 Models

- **ARMA** (p,0,q)
- **ARIMA** (p,d,q)
- **SARIMA** (p,d,q)(P,D,Q)m : S stands for Seasonal ARIMA . Over here (P,D,Q)m are same like ARIMA's (p,d,q) but they deal with seasonal part of the data while (p,d,q) with trend, m is no of seasons

Let's consider an example

ARIMA(3,1,2)(0,0,2)[12] -> This model suggest that Trend has an AR of 3, MA of 2 and it needs to be differenced 1 to remove the trend of the data. Coming to seasonality, it only has MA part and does not need to be differenced.

All being said, lets execute it on our AXP data.

```
Forecast method: ARIMA(5,2,0)

Model Information:
Series: train.ts
ARIMA(5,2,0)

Coefficients:
          ar1      ar2      ar3      ar4      ar5
       -0.7329  -0.6353  -0.4877  -0.3085  -0.1723
s.e.    0.0218   0.0263   0.0278   0.0263   0.0218

sigma^2 estimated as 0.757:  log likelihood=-2623.93
AIC=5259.86    AICc=5259.9    BIC=5293.62

Error measures:
                        ME        RMSE       MAE          MPE      MAPE       MASE         ACF1
Training set -0.000200252 0.8685451 0.6313014 0.002807671 1.456048 0.03812883 -0.02306801
```
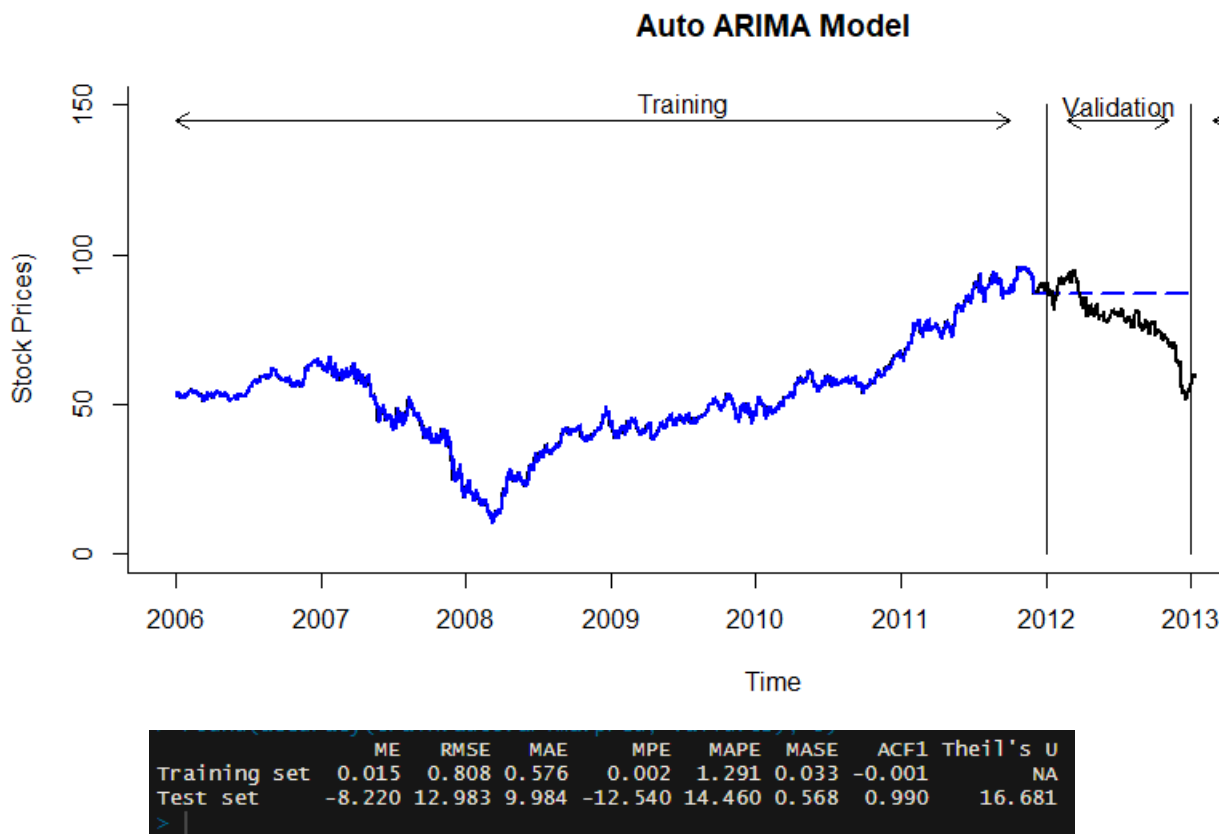
We use auto arima () function. R calculates the best function components for our data. Here we see that our data has autocorrelation of 5 and it needs to be differenced twice to remove trend and make it stationary.

Auto ARIMA Model


```
                    ME    RMSE     MAE      MPE    MAPE   MASE     ACF1  Theil's U
Training set     0.015   0.808   0.576    0.002   1.291  0.033   -0.001        NA
Test set        -8.220  12.983   9.984  -12.540  14.460  0.568    0.990    16.681
>
```

We see that ARIMA model if highly underfitted also it is performing marginally better than the base model.

Overall, statistical models are not performing well with the testing set also they are performing closely to the base model RMSE accuracy.

FBPROPHET

SUPPLY & DEMAND OF FORECASTS
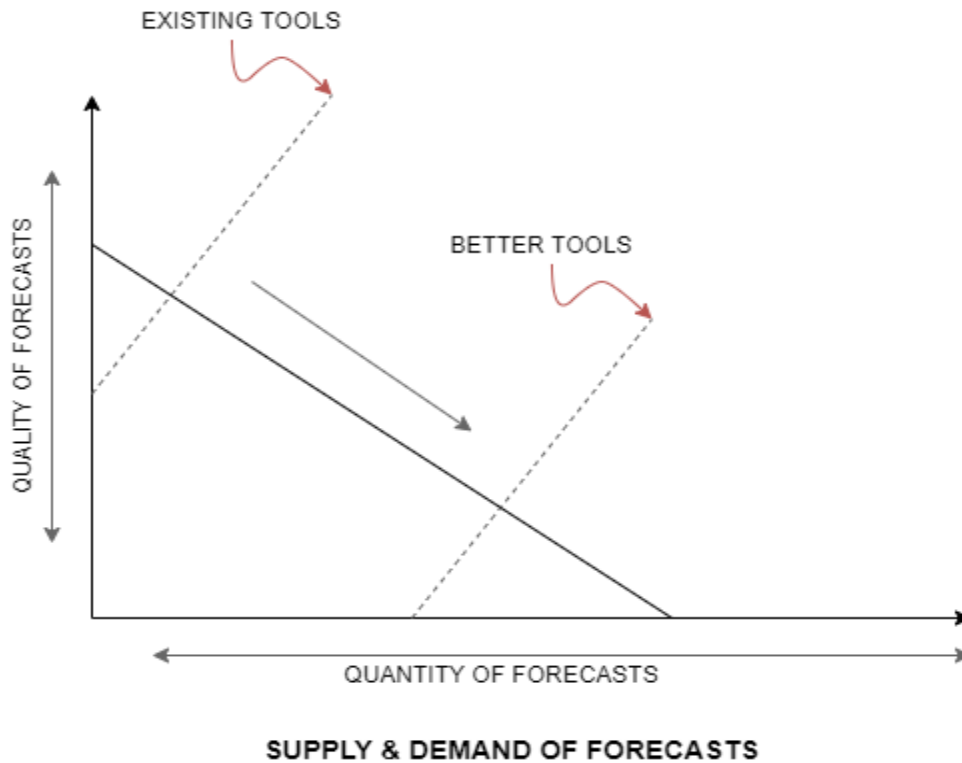
Supply and Demand graph shows the relationship that if forecasts made forecasting easier how many people will do forecasts. Usually forecasting can be a bit trickier to generic population hence people find it tough to forecast. The solution to this problem was that to make such model that can automate or semi-automate forecasting.

Keeping this is mind the state of the art **FBPROPHET** was created at Facebook by Sean J. Taylor, Benjamin Letham.

About Prophet**:** Prophet is a procedure for forecasting time series data based on an additive model where non-linear trends are fit with yearly, weekly, and daily seasonality, plus holiday effects. It works best with time series that have strong seasonal effects and several seasons of historical data. Prophet is robust to missing data and shifts in the trend, and typically handles outliers well.
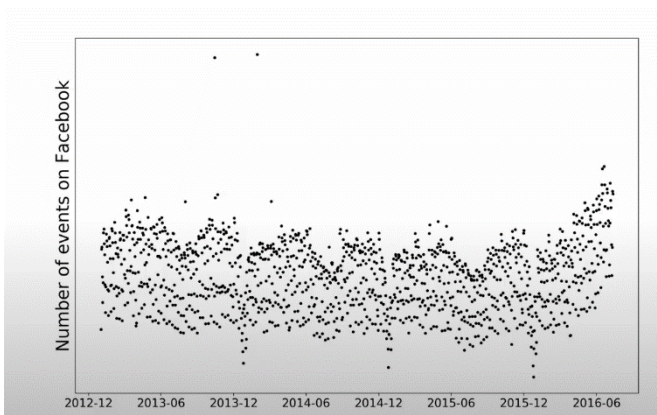
(Actual Data Points at FB)
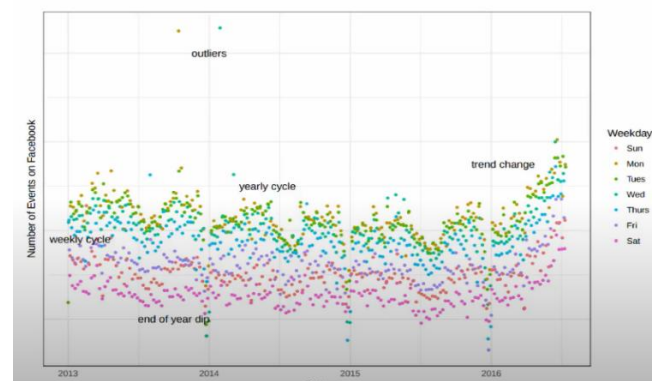
(Data Visualization)

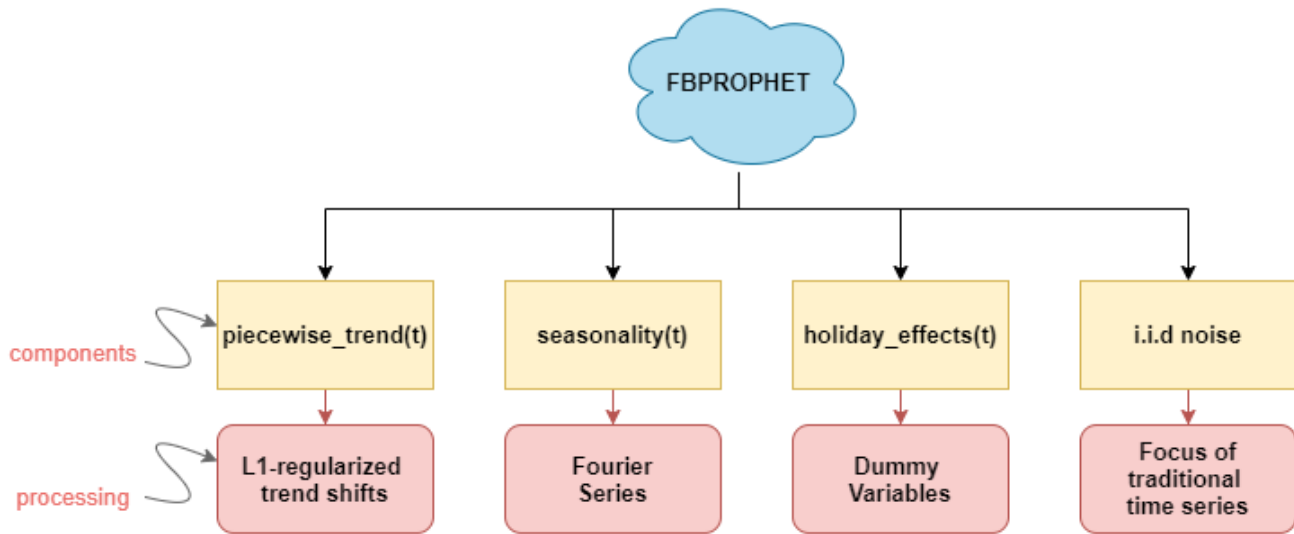## Breaking down FB Prophet

The FBPROPHET consists of 4 major components

- **Piecwiese_trend(t)**
- **Seasonality(t)**
- **Holiday_effects(t)**
- **Iid | Noise**

Piecwiese_trend(t) – The trend component is based on L1 regularized trend shifts.

Seasonality(t)- Fourier series adapts the seasonality very well.

Holiday effects(t) – You can pass a list of holidays to the model stating that there can be irregular patterns during these days and the days following it hence these dates will be given less importance. For example, if you own a restaurant you can expect a customer surge during the public holidays. This feature also helps to target certain outliers.

Iid noise- An iid noise refers to a case when errors follow a distribution with unique variance (as in white noise), but not necessarily with a zero expectation. If you have an iid noise, then you have at least one explanatory variable that is captured in the residuals.

Prophet Time Series Decomposition

Now let's Run the model

Black points are the actual data point and blue line is predicted line via FB prophet. The blue region signifies confidence of 95%. It means that the model is 95% confident that the prediction will be under the blue shaded region.
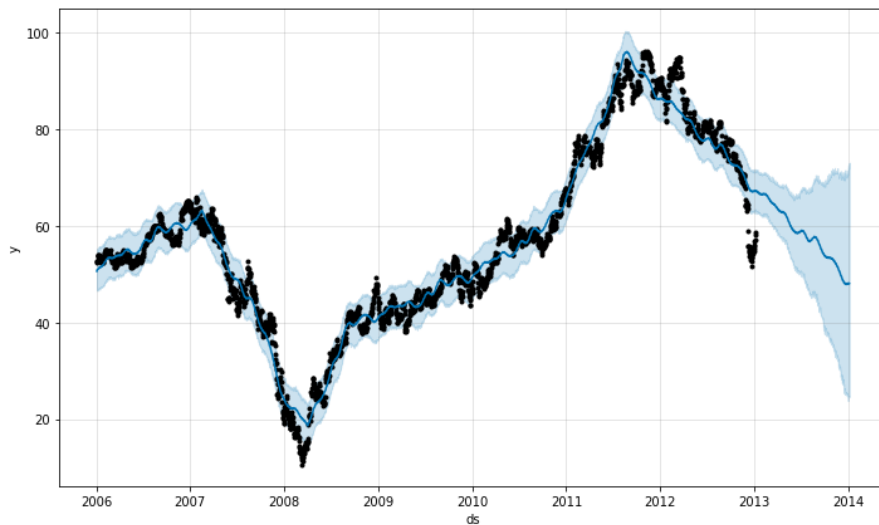
Another beautiful feature of Fb prophet is that is points/marks the major changes in trend and seasonality. In the image given below the red line shows major changes. Stock market is Highly Volatile. which can be proven below.


```
rmse(predictions,test['y'])
```

```
18.997844250300645
```

FB Prophet is supposed to the best model in the market, but it is not performing good on AXP Data. But when run on other datasets its giver very high performance.

Fb Prophet Advantages over Other Models

- Priors on seasonality parameters
- Prior on how often we expect changepoints
- Functional form for growth (linear/logistic)
- Covariates and holidays
- Custom seasonality's
- Automate / Semi automated Forecasting

Deep Learning for Time Series

| 1986 | 1997 | 2014 | 2015 | 2019 |

1986 — Deep Learning was introduced by Rina Dechter

1997 — LSTM was proposed in 1997 by Sepp Hochreiter and Jürgen Schmidhuber

2014 — Kyunghyun Cho et al. put forward a simplified variant called Gated recurrent unit (GRU)

2015 — Researchers started to use LSTM & GRU's for Time-Series

2019 — People have started to use Transformers for Time-Series
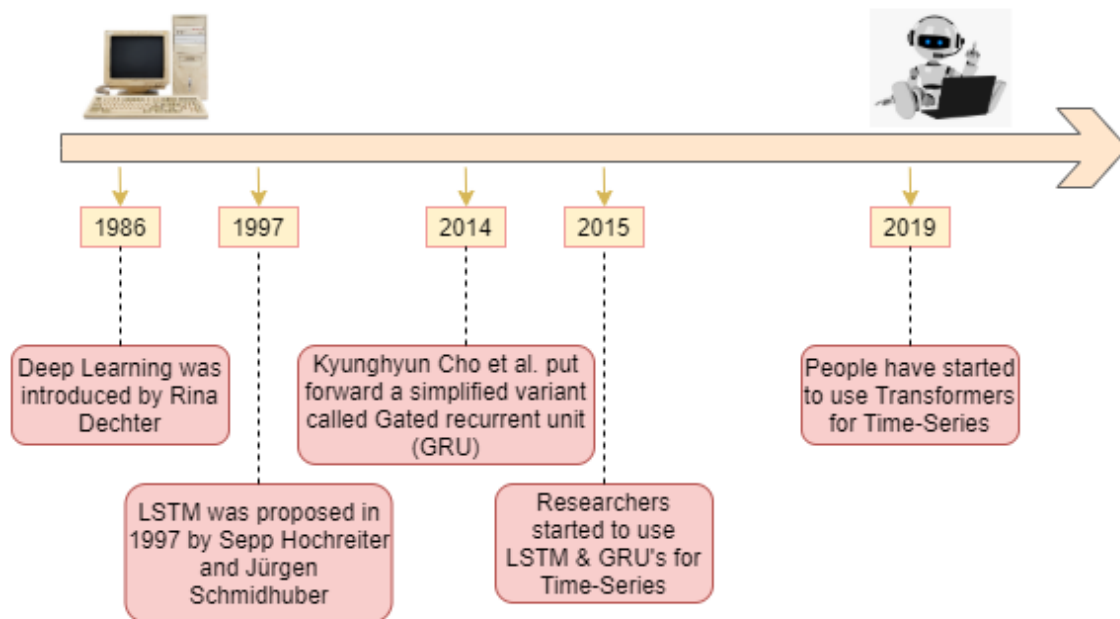
Introduction about Deep Learning

What is this so-called term called as Deep Learning which has gained so much popularity in today's world?

A broader or extended family of Machine learning, which is itself a subset of artificial intelligence, deep learning employs powerful machines, large data sets, "supervised" (trained) neural networks and an algorithm called back-propagation to identify objects by imitating neuronal layers in the neocortex of a human brain. Artificial neural networks (ANNs) is inspired by the transmission of information and the propagation of coordination nodes within biological systems. ANN's have different biological brain variations. Neural networks tend to be static and symbolic, while the biological brain is fluid (plastic) and similar to most living organisms.

Neural Network Architecture | ANN NEURON | RNN NEURON
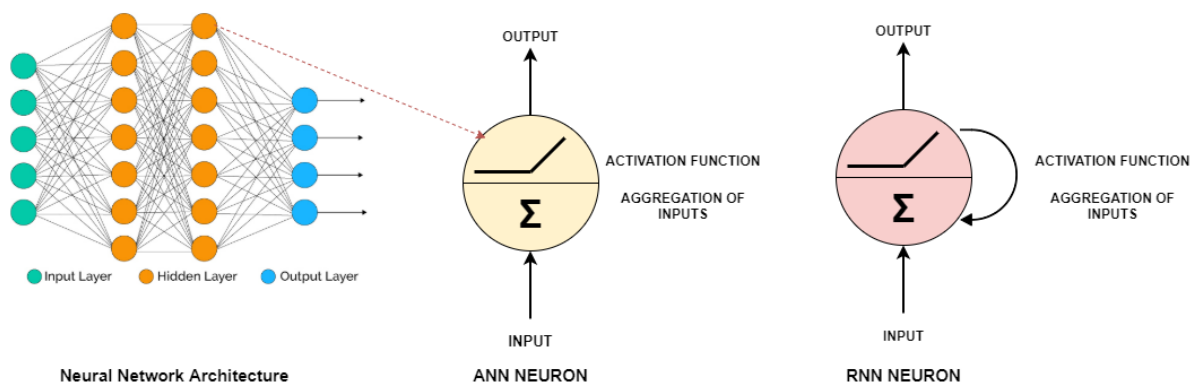
What is RNN?

Recurrent Neural Network is an implicitly memorized generalization of feedforward neural network. RNN is repetitive in nature, because it performs the same task for each data input while the output of the current input depends on the previous one. It is copied and sent back into the recurrent network after the output has been generated. It considers the current input and the feedback it has accumulated from the previous input for deciding. Unlike feedforward neural networks (dense network), RNNs use their internal state to process input sequences. This makes them applicable to tasks such as text recognition, speech recognition etc. Both inputs are distinct one from another in other neural networks. In RNN, however, all inputs are connected to one another. Above given image is of a RNN and ANN cell. We can see how these o/p of activation output is fed in neuron itself as input.
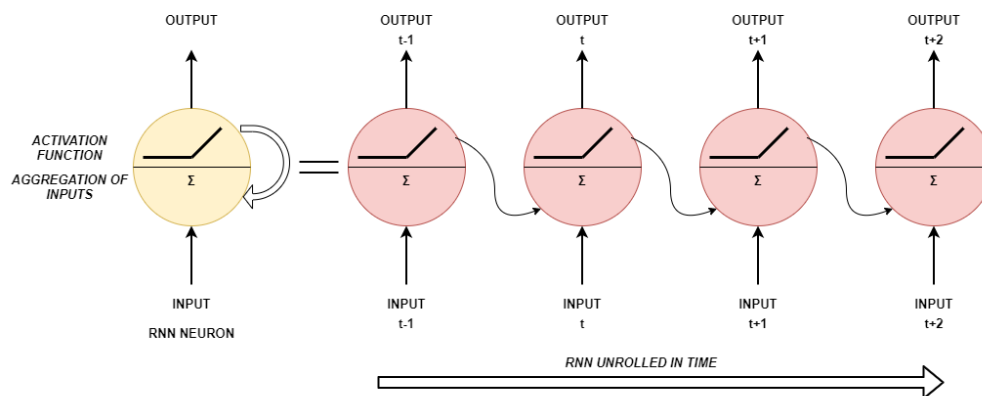
When Different RNN cells are unrolled in time a RNN network sequence is formed. A RNN cell has 2 inputs, the actual data and the input from an earlier cell
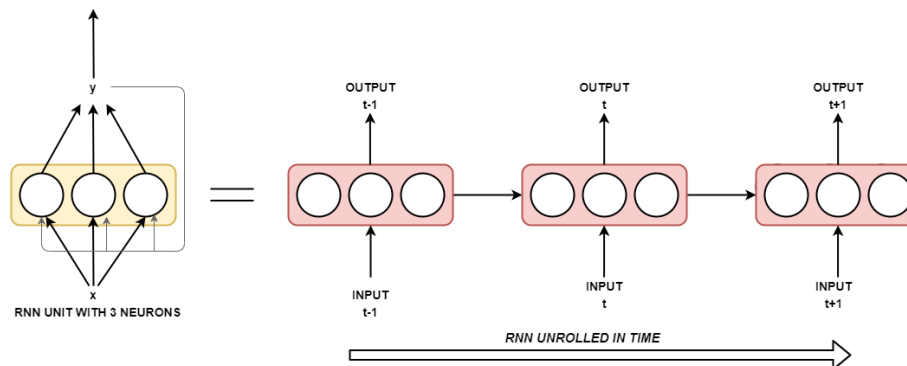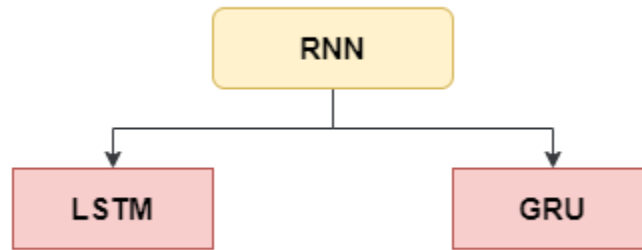
A single RNN cell can have multiple neurons, we can set it according our need and computational capacity.

Drawbacks associated with RNN

- It cannot remember long sequences
- Vanishing Gradient
- Exploding Gradient

Hence LSTM's and GRU's were created to address the issues of RNN

LSTM

LSTM are usually used for chatbot, wherein long sequence of data is used. For chatbots we need to remember the starting training values as well because let's say that a gender was trained during start. It should be remembered till the end of the training. This concept can be used in time series data as it is a sequence data. Long Short-Term Memory (LSTM) networks are a revamped variant of recurrent neural networks, allowing memory retrieval of past data simpler. The RNN problem of the vanishing gradient is solved here.
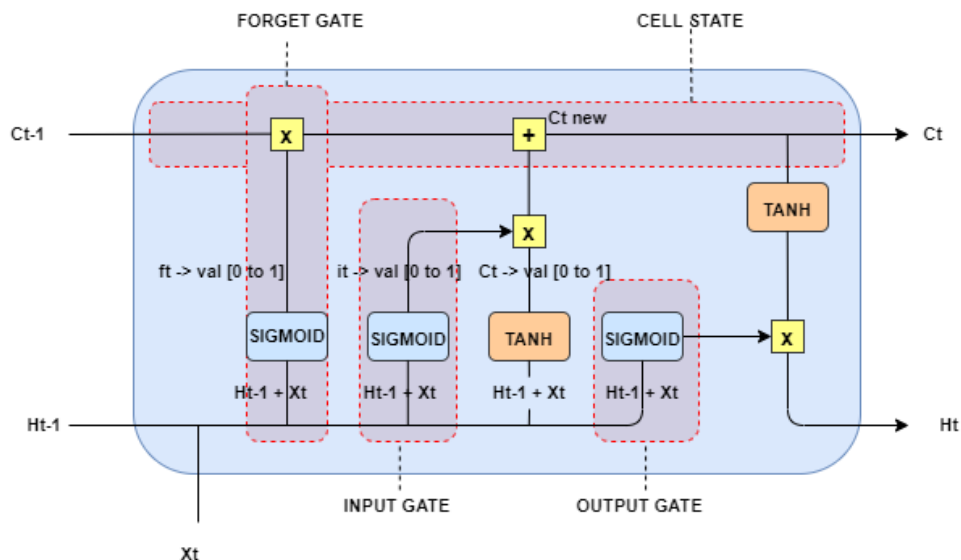
States are information carriers throughout the call and the network

- **Ht**: Previous hidden state (info about previous inputs also used for prediction)
- **Xt**: Current input
- **Ct**: Previous cell state
- **Cell state**: Acts as a highway throughout the sequence chain. Just like memory of the network. Carries memory of earlier cells thereby reducing the vanishing gradient problem

Activation: The activation resamples the values of its input into a range of values depending upon the activation function. Sigmoid and Tanh function are used in LSTMS and GRU.

- **Sigmoid Activation** – Squishes value between 0 and 1. Helpful to update or forget the data as any number getting multiplied 0 will be null thereby forgetting the value. Any number multiplied by 1 is kept.
- **Tanh Activation** – Squishes the value between -1 and +1. This helps to regulate the network
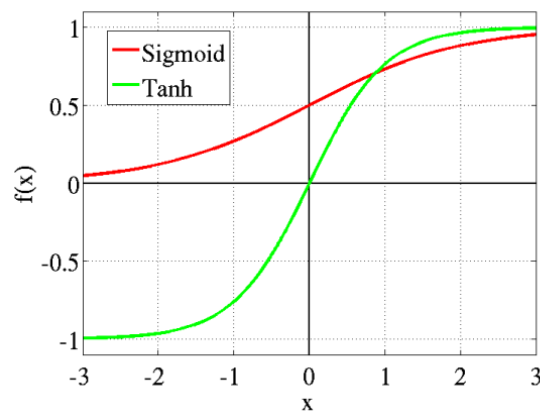
Gates: Information gets added or removed using gates.

- **Forget Gate** – Which information to be kept or forgotten
- **Input Gate** – To update the cell state
- **Output gate** - Decides what information to be passed on to the next hidden state
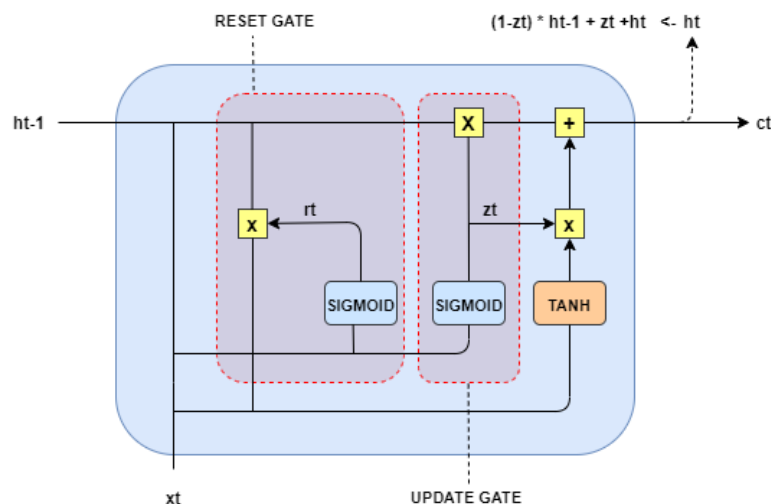
GRU

The Difference between LSTM and GRU is that the GRU got rid of the cell state and used hidden state to transfer information. It has merged forget and input gate of LSTM into a gate called as Update Gate.

Reset Gate: How much past information to forget

Update Gate: Similar to forget and Input gate of LSTM. Decides which information to throw away and which information to add/update.

Benefit of GRU

- Less tensor operations and hence speeder to train than LSTM

How does LSTM & GRU work

Step 1: Load Dense, LSTM, Dropout, GRU, Bidirectional Keras Packages

Step 2: Split the Data into Training and Testing

Step 3: Scale the Training Data set

```
[13] scaler = MinMaxScaler(feature_range=(0,1))
     training_set_scaled = scaler.fit_transform(training_set)
     scaler

 [→  MinMaxScaler(copy=True, feature_range=(0, 1))
```

Step 4: Select the number of past data points that influence the current data point

```
[14] X_train = []
     y_train = []
     for i in range(100,2496):
         X_train.append(training_set_scaled[i-100:i,0])
         y_train.append(training_set_scaled[i,0])
     X_train, y_train = np.array(X_train), np.array(y_train)
```

We see that for a selected data point i we select past 100 data points also we save i as y variable

Step 5: Create a Sequential LSTM or GRU Model

- We tune following hyperparameters
    - **Unit**: # neurons in each LSTM layer
    - **Optimizer**: Optimizers are algorithms or methods used to change the attributes of your neural network such as weights and learning rate in order to reduce the losses.
    - **Epochs**: Epoch refers to one cycle through the full training dataset
    - **Dropout**: Neurons which we randomly unselect while training
    - **Batch Size**: The batch size defines the number of samples that propagate through the network.
    - **Return Sequence**: Layer that will provide access to the hidden state output and the cell state.

```
[16]  regressor = Sequential()

      regressor.add(LSTM(units=50, return_sequences=True, input_shape=(X_train.shape[1],1)))
      regressor.add(Dropout(0.2))

      regressor.add(LSTM(units=50, return_sequences=True))
      regressor.add(Dropout(0.2))

      regressor.add(LSTM(units=50, return_sequences=True))
      regressor.add(Dropout(0.2))

      regressor.add(LSTM(units=50, return_sequences=True))
      regressor.add(Dropout(0.2))

      regressor.add(LSTM(units=50, return_sequences=True))
      regressor.add(Dropout(0.2))

      regressor.add(LSTM(units=50))
      regressor.add(Dropout(0.2))

      regressor.add(Dense(units=1))


      regressor.compile(optimizer='rmsprop',loss='mean_squared_error')

      regressor.fit(X_train,y_train,epochs=50,batch_size=32)
```
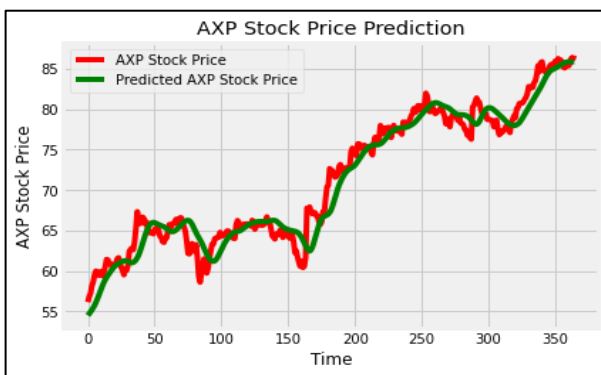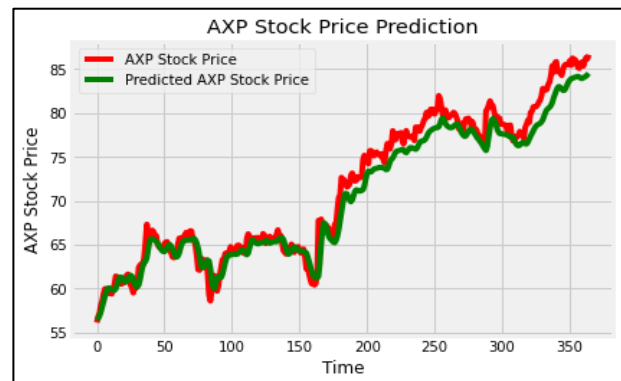
Step 6: Apply inverse transform on data and plot the entire data set with prediction

Step 7: Calculate the RMSE

Output after running LSTMS | GRU on Colab


LSTM | RMSE = 1.64

GRU | RMSE = 1.84

Hence, we can see that both the models outperform every model performed I this report for AXP dataset.

Tensor board Visualization

As we know that Neural networks is a black box methodology. Hence, TensorBoard was made to provide some spotlight to the behind the scene functionalities of black box. It basically visualizes accuracy as well as losses vs the epochs. It shows connection graphs which clarifies how the connections are made during layer stacking. Also, it provides various histograms of weights, biases, or other tensors as they change over time. Over here we have connected our LSTM models output to TensorBoard callback like given below.

```
regressor.add(Dense(units=1))
log_dir = os.path.join("/logs",datetime.datetime.now().strftime("%Y%m%d-%H%M%S"))
#log_dir = "logs/fit/" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
tensorboard_callback = TensorBoard(log_dir=log_dir, histogram_freq=1, batch_size=32, write_graph=False  , write_grads=True ,update_freq='epoch')
regressor.compile(optimizer='adam',loss='mean_squared_error')
regressor.fit(X_train,y_train,epochs=50,batch_size=32, callbacks=[tensorboard_callback])
```

Below given image is the screen shot of the output screen of Tensorboard Scalar.
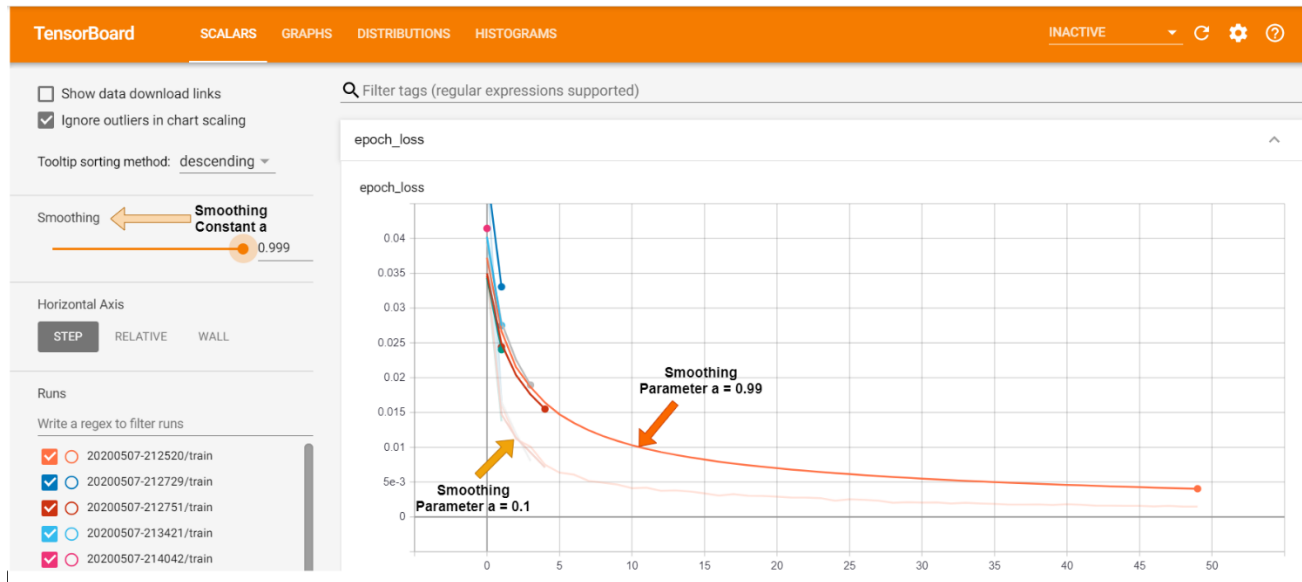
Smoothing constant is a, it decides the weightage to which part of data do you want to focus on.

- **a closer to 1 -> Weightage given more to recent data.**
- **a closer to 0 -> Weightage given more to past data.**

Model Optimization

LSTM	Optimizer	Rmsprop		Adam	
	Batch Size	32	64	32	64
Epochs	10	6.78	6.04	5.84	5.75
	20	3.54	3.02	2.82	2.05
	50	2.98	2.54	1.64	1.06

GRU	Optimizer	Adam		SGD	
	Batch Size	32	64	32	64
Epochs	10	6.8	6.51	5.96	5.85
	20	3.7	3.12	2.9	2.43
	50	3.12	2.65	1.84	1.12

I tried using various combinations of **Batch size | Optimizer | Epochs** out of which I found **LSTM** with Adam optimizer, 50 epochs to be most accurate yet I have selected with 32 batch size has it had marginal difference from Adam with batch size of 64, Hence our goal should be to have lowest RMSE with lowest capacity, same goes with GRU the best optimizer was SGD.

Model Comparison

Model Domain	Model Type	RMSE
Base Model	Naïve Model	18.537
	Naïve Seasonal Model	15.367
Smoothing Models	SES	13.768
	Holt's Model	12.985
	Holt's Winter Model	13.028
Regression Models	Linear Trend	22.05
	Linear Trend and Seasonality	31.66
	Quadratic Trend	39.63
	Quadratic Trend and Seasonality	40.958
ARIMA Models	ARIMA Models	19.983
FB Prophet	FB Prophet	18.997
Deep Learning	LSTM	1.64
	GRU	1.84

Conclusions

We can see that Deep Learning models are outperforming other models by considerably higher margins. It is because they have higher computational functionalities as well as they offer better hyper parameter configurations to select upon. Hence, they seem to fit the training as well as testing data far better than traditional statistical models. Talking about statistical models, ARIMA is supposed to be the best model as it was created to fit Trend and seasonality effortlessly with the help of its 6 parameters combinations. When we consider forecasting globally, Facebook Prophet is often regarded as the best model out there in the market. Over here we see that it is underperforming thoroughly, the reason is because stock is influenced by hundreds of other factors, be it some supply chain data as such, Prophet would have outperformed other models even LSTM's as well. The credit goes to its Fourier transforms used to track seasonality. Coming toward our last model. Like the name says Deep Learning. The models use the latest advancements in fields of A.I, though it has been only for 4 years in market, and it is not frequently used as the ARIMA. It has a tremendous future scope.

Future Scope

In 2017 Transformer Architecture was introduced. It basically processes all sequence data parallelly while traditional LSTM take one input at a time. For example. If you type "How are you?", the LSTM chat bot encoder will parse one word at a time, while the transformer chat bot will, parse all the 3 words parallel in one go. Researchers have started to use this, mechanism in Time series sequence data and is supposed to be better than LSTM. Hence, we can expect 2020 to be the year of transformers for Time series.

References

Websites

https://www.itl.nist.gov/div898/handbook/pmc/section4/pmc4.html

https://www.statisticssolutions.com/time-series-analysis/

https://www.aptech.com/blog/introduction-to-the-fundamentals-of-time-series-data-and-analysis/

http://userwww.sfsu.edu/efc/classes/biol710/timeseries/timeseries1.html

https://www.researchoptimus.com/article/what-is-time-series-analysis.php

https://towardsdatascience.com/understanding-gru-networks-2ef37df6c9be

https://d2l.ai/chapter_recurrent-modern/gru.html

Research Papers

F. Karim, S. Majumdar, H. Darabi and S. Chen, "LSTM Fully Convolutional Networks for Time Series Classification," in *IEEE Access*, vol. 6, pp. 1662-1669, 2018, doi: 10.1109/ACCESS.2017.2779939.

X. Zhao, X. Han, W. Su and Z. Yan, "Time series prediction method based on Convolutional Autoencoder and LSTM," *2019 Chinese Automation Congress (CAC)*, Hangzhou, China, 2019, pp. 5790-5793, doi: 10.1109/CAC48633.2019.8996842.

J. Ji and J. Hou, "Forecast on Bus Trip Demand Based on ARIMA Models and Gated Recurrent Unit Neural Networks," *2017 International Conference on Computer Systems, Electronics and Control (ICCSEC)*, Dalian, 2017, pp. 105-108, doi: 10.1109/ICCSEC.2017.8446813.

N. K. Chikkakrishna, C. Hardik, K. Deepika and N. Sparsha, "Short-Term Traffic Prediction Using Sarima and FbPROPHET," 2019 IEEE 16th India Council International Conference (INDICON), Rajkot, India, 2019, pp. 1-4, doi: 10.1109/INDICON47234.2019.9028937.

R. Fu, Z. Zhang and L. Li, "Using LSTM and GRU neural network methods for traffic flow prediction," 2016 31st Youth Academic Annual Conference of Chinese Association of Automation (YAC), Wuhan, 2016, pp. 324-328, doi: 10.1109/YAC.2016.7804912.