Indian Trade Data Analysis and Forecasting

By

Akarsh Somani (39/CSE/16005/0000162)

Gaurav Misra (39/CSE/16015/0000172)

Bachelor Thesis submitted to

Indian Institute of Information Technology Kalyani

For the partial fulfillment of the degree of

Bachelor of Technology

In

Computer Science and Engineering

June, 2020

Certificate

This is to certify that the thesis entitled “Indian Trade Data Analysis and Forecasting” being submitted by Akarsh Somani and Gaurav Misra, undergraduate students, Reg. No 162 and 172, Roll No. 39/CSE/16005 and 39/CSE/16015, respectively, in the Department of Computer Science and Engineering, Indian Institute of Information Technology Kalyani, West Bengal 741235, India, for the award of Bachelors of Technology in Computer Science and Engineering is an original research work carried by them under my supervision and guidance. The thesis has fulfilled all the requirements as per the regulations of Indian Institute of Information Technology Kalyani and in my opinion, has reached the standards needed for submission. The work, techniques and the results presented have not been submitted to any other University or Institute for the award of any other degree or diploma.

Dalia Nandi

Assistant Professor

Electronics and Communication Engineering Department

Indian Institute of Information Technology Kalyani

Declaration

I hereby declare that the work being presented in this thesis entitled, “Indian Trade Data Analysis and Forecasting”, submitted to Indian Institute of Information Technology Kalyani in partial fulfillment for the award of the degree of **Bachelor of Technology** in Computer Science and Engineering during the period from July, 2019 to June, 2020 under the supervision of Dr. Dalia Nandi, Department of Electronics and Communication Engineering, Indian Institute of Information Technology Kalyani, West Bengal - 741235, India, does not contain any classified information.

Akarsh Somani and Gaurav Misra

Reg No/Roll No: 39/CSE/16005/162 and 39/CSE/16015/172

Department: Computer Science and Engineering

Institute Name: Indian Institute of Information Technology Kalyani

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

.....

Dr. Dalia Nandi

Position: Assistant Professor

Departmental address: Electronics and Communication Engineering

Place: Kalyani

Date: June 18, 2020

Acknowledgments

Firstly, we would like to thank our supervisor Dr. Dalia Nandi for her support and guidance to complete this project work without whom we would have not been able to make out this far. We would also like to thank our friends who supported us greatly and were always willing to help us. We are very grateful to Department of Computer Science and Engineering, Indian Institute of Information Technology Kalyani, West Bengal - 741235, India, for providing us this wonderful opportunity. Lastly, we would like to thank our Parents a God for their never-ending grace.

Akarsh Somani and Gaurav Misra

Reg No/Roll No: 39/CSE/16005/162 and 39/CSE/16015/172

Department: Computer Science and Engineering

Institute Name: Indian Institute of Information Technology Kalyani

Place: Kalyani

Date: June 18, 2020

Abstract

Trade is an economic concept which involves Buying and Selling of the commodities, or exchanging goods and services between needy people. Trade is important in a way that it increases competition and decreases overall world wise cost of a product.

India is only second largest market place (after China) in terms of human resource. Hence many foreign countries exploit this fact, especially China, to sell their products. Hence our Import Trade matters a lot from economic point of view.

On the other hand, we, being second largest population, also produce a vast range of products, all thanks to diversity present in India, while in North India, we have huge production of almost all types of grains and also fruits; in South India, we produce every kind of spices; in East India, we have a huge amount of tea-production. All of this is just a chunk of what we Export to other countries.

So to balance our economy with feasible amount of Import and Export trade, we need to analyze which area of production needs more attention for Exports and which product needs to be encouraged for production to decrease the amount of Imports. So our Data Analysis part points at this area of our thesis.

Import and Export both are very dynamic in nature since every year we evolve in what we want as an Import and what we produce in order to Export. If we can forecast the trade amount we might know the areas, which are putting our economy into deficit and also reinforce those areas where we're improving in terms of trade profit.

Our Motivations for this thesis are listed as following -

1. Which HS Code involves the most import/export?
2. From which country we import/export the most?
3. Forecasting the Indian import/export data to predict what should be our country's next move?
4. How our trades are evolving across the years?
5. Getting Rid of Trade Deficit (Case Study)?

Contents

Chapter 1

Introduction

In this thesis, we will be dealing with the **Indian trade related Data Analysis and Forecasting** with respect to other countries and try to interpret the impact of the trade. Trade is one of the important factors for the economy of any country and that underlines the purpose of this thesis.

1.1 Significance of Trade Analysis

Whenever we try to solve any problem using Data Science, then at first we try to answer the *Data* part of Data Science. And after getting data, we more often than not, analyze it. That's the Analysis part we're going to discuss here. Actually the analysis part helps with solving every kind of problem involving Data Science. We just can't get into modeling without analyzing the data we are dealing with. So we also start traditionally by performing data analysis on available trade data HS code wise and also country wise. Analysis part gives us advantage in data preprocessing, feature selection, better model selection and how to infer our forecasting.

While analyzing trade data, we found more interesting and valuable insights than we thought prior to analysis. Although we know that all of our insights may not carry weightage in real life still some of them can bring revolutionary change in how our trade affects economy.

1.2 Significance of Trade Forecasting

Just think about the following – What can we achieve if we know the next month's total import or export amount? The answer depends on how much accuracy can be achieved while forecasting future data points. Assuming we get fairly well accuracy, we can surely say that there're a lot of use cases to get returns associated with trade amount forecasting.

We purchase things regularly out of our need. Sometimes more and sometimes less, but if we somehow know that next month's import is going to be lower, we can conclude that prices are also going to get cheaper. The reason behind lowering of price is that either demand is low or we are producing the concerned product on a larger scale in our country, hence the decrement in import.

The most specific use-case appear to be for industries, since they buy very precious machineries for functionality and production purpose, hence if they get to know that imports are going to decrease and hence prices of machines can be cheaper they can plan to buy machines earlier than usual and same for delaying the purchase in case of increment in imports and decrement in exports.

To highlight the advantages of trade forecasting for Government, let's first talk about the custom duty (taxes enforced upon the goods which foreign countries export to us). So each good which comes from other countries to our country generates some kind of revenue source. It forms a good portion of what government controls. In this way government controls the market prices for imported products. So, if government feels that homemade products are not able to compete with foreign products and next month's import is going to be even higher than it can simply increase the custom duty on that product which will reflect on the cost associated, hence domestic products and companies gets a chance.

Government can also test its effectiveness in storing agricultural products, like cold storages etc. If we see a downward trend in exports of some agricultural product and next month's forecast also depicts the same, then it can be deduced that something is causing difficulty in producing the corresponding product, it can be either the middlemen or our distribution system or it can even be natural.

These are just few ways of seeing things regarding trade forecasting, there're many other practical scenarios which can be influenced by forecasting in a positive manner.

Chapter 2

Literature Survey

2.1 Forecasting Models

- **Exponential Smoothing** (Sandip Roy, Sankar Prasad Biswas, Subhajyoti Mahata, Rajesh Bose 2018)**:** It is the technique for smoothing(Averaging) the timeseries using exponential window function. In simple moving average the past observation are equally weighted where as here it exponentially decreases over the time.
- **Auto Regressive Model (AR)** (Nimisha Tomar, Durga Patel, Akshat Jain 2020)**:** It is used when a value from the time series has dependency on previous values e.g. $X_t = f(X_{t-1})$. The order of an auto-regression is the number of immediately preceding values in the series that are used to forecast the current value e.g. order of 2 denotes that the value X_t is dependent on X_{t-1} and X_{t-2}.
- **Moving Average Model** (S. Vemuri, R. Balasubramanian, E.F. Hill 1974)**:** In time series analysis, the Moving-Average model (MA model) specifies that the output variable depends linearly on the current and various past values of a stochastic (imperfectly predictable) term.
- **Holt-Winters Model** (Chatfield 1978)**:** It is an extension over the simple exponential smoothing method. Here we use triple smoothing with the factor - seasonal period, trend type and seasonal type. Here seasonal and trend type means *Multiplicative* or *Additive*.
- **ARIMA Model** (Lyashenko 2020)**:** Auto Regressive Integrated Moving Average (ARIMA) models are applied in some cases where data show evidence of non-stationarity, where an initial differencing step (corresponding to the "integrated" part of the model) can be applied one or more times to eliminate the non-stationarity. We'll discuss Additive ARIMA, Multiplicative ARIMA and Seasonal ARIMA.
- **CNN Model:** In deep learning, a Convolutional Neural Network (CNN) is a class of deep neural networks, most commonly applied to analyzing visual imagery.
- **RNN Model** (Yi-Ting Tsai, Yu-Ren Zeng, Yue-Shan Chang 2018)**:** A Recurrent Neural Network (RNN) is a class of artificial neural networks where connections between nodes form a directed graph along a temporal sequence. This allows it to exhibit temporal dynamic behavior.
- **LSTM Model:** Long Short-Term Memory (LSTM) is an artificial recurrent neural network (RNN) architecture used in the field of deep learning. Unlike standard feedforward neural networks, LSTM has feedback connections.

2.2 Which one to use

- Exponential/Moving Average models may perform moderately in short term forecasting but are not reliable since there's no weightage of past values taken.
- Autoregressive models forecast only on past information; they implicitly assume that the fundamental forces, that influenced the past prices, will not change over time. This can lead to surprising and inaccurate predictions.
- ARIMA method is appropriate only for a time series that is stationary with signs of trend and seasonality, whereas our data contains some randomness.
- CNNs are inappropriate for sequential data as they don't use past values to make predictions.
- RNNs are used for processing sequential data but perform badly in case of long-term dependency, this happens because of vanishing gradient problem.
- LSTMs store past information and know what to forget and what not to forget as a modification over RNNs.

Now we'll try each model one by one and See which one provides us with the best forecasting results.

Chapter 3

Dataset Preparation

3.1 Data Source

We scraped the trade dataset from Department of Commerce, Govt. of India website. Data used for Analysis and Forecasting is monthly. Monthly data is available from January, 2006 to January, 2020. We have total trade amount (Import/Export) for each month which is expressed in million US dollars. For Analysis we break it into HS code wise and Country wise data, while for Forecasting we take the total amount of trade occurred.

HS Code - Harmonized System (HS) of tariff nomenclature is an internationally standardized system of names and numbers to classify traded products.

Sample Data:

Following is a snippet of monthly Import/Export dataset -

Date	Import	Export
2006-01-01	12519.71	9143.66
2006-02-01	11479.69	8993.29
2006-03-01	14314.02	11560.97
2006-04-01	12924.18	8624.66
2006-05-01	15105.67	10109.30
...	...	...
2019-09-01	37693.81	26007.97
2019-10-01	37241.53	26213.39
2019-11-01	38101.94	25630.87
2019-12-01	38577.34	27142.63
2020-01-01	41146.86	25882.90

169 rows × 2 columns

Table 3-1: Sample of monthly trade data

Below is the sample from monthly Import/Export Data HS code wise and Country wise -

Date	HSCode	Commodity	Import_Value	Export_Value
2007-01-01	1	LIVE ANIMALS.	0.02	0.78
2007-01-01	2	MEAT AND EDIBLE MEAT OFFAL.	0.01	51.44
2007-01-01	3	FISH AND CRUSTACEANS, MOLLUSCS AND OTHER AQUAT...	1.12	116.92
2007-01-01	4	DAIRY PRODUCE; BIRDS' EGGS; NATURAL HONEY; EDI...	1.88	33.96
2007-01-01	5	PRODUCTS OF ANIMAL ORIGIN, NOT ELSEWHERE SPECI...	1.47	3.01
...	...	...	...	...
2020-01-01	95	TOYS, GAMES AND SPORTS REQUISITES; PARTS AND A...	46.51	28.64
2020-01-01	96	MISCELLANEOUS MANUFACTURED ARTICLES.	62.46	49.80
2020-01-01	97	WORKS OF ART COLLECTORS' PIECES AND ANTIQUES.	4.68	3.23
2020-01-01	98	PROJECT GOODS; SOME SPECIAL USES.	429.85	1.02
2020-01-01	99	MISCELLANEOUS GOODS.	3.80	7.99

15288 rows × 4 columns

Table 3-2: Sample of monthly trade data by HS code

Date	Country	Import_Value	Export_Value
2007-01-01	AFGHANISTAN TIS	0.48	22.22
2007-01-01	ALBANIA	NaN	0.34
2007-01-01	ALGERIA	73.86	29.94
2007-01-01	AMERI SAMOA	0.09	0.02
2007-01-01	ANGOLA	0.81	13.07
...	...	...	...
2020-01-01	VIETNAM SOC REP	466.37	373.05
2020-01-01	VIRGIN IS US	0.37	0.56
2020-01-01	YEMEN REPUBLC	1.07	65.07
2020-01-01	ZAMBIA	80.33	20.58
2020-01-01	ZIMBABWE	0.55	9.45

34107 rows × 3 columns

Table 3-3: Sample of monthly trade data by Country

3.2 Preprocessing of Dataset

We have total 169 data points for both kinds of trade Import and Export. Now we break the dataset into two sets – training and validation such that training set includes 156 data points and validation set includes 13 data points to forecast upon.

We have no missing values hence we don't need to impute any data point. We have all numerical continuous columns hence we don't need to one hot encode as well.

For LSTM, we normalize the monthly data before fitting our model i.e. the input will be in between 0 and 1. Also we used 12 previous month's data to forecast on validation data.

3.3 Data Visualization

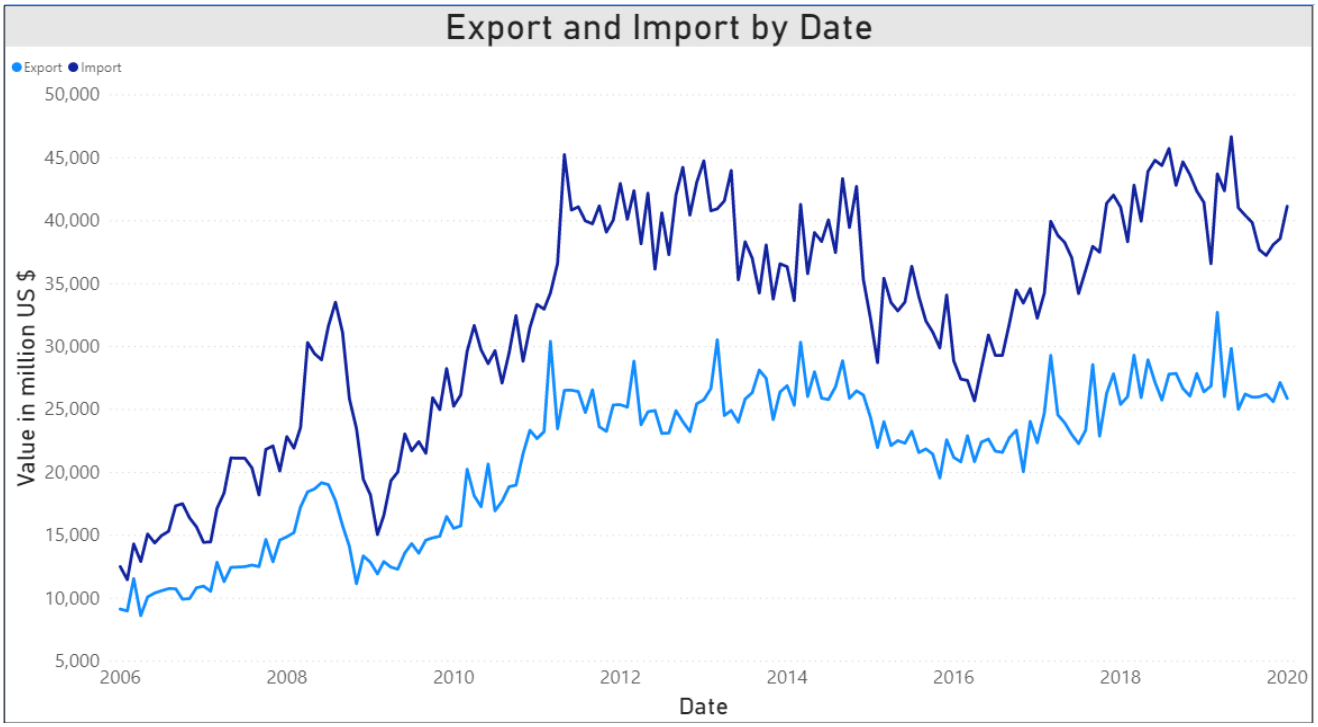

Figure 3-1: Visualization of monthly Import and Export Data

3.4 Insights about Data

We analyzed the countries and products which India trades the most. China tops the list with around $4379 million Import followed by United Arab Emirates, Saudi Arab and USA. While USA is top most country to which India exports the most $3013 million followed by UAE, China, Hong Kong and Singapore. As we know that Trade is the key to normalize costs across the world. India is a developing country with rich resources and population thus attracts the business from all over the world.

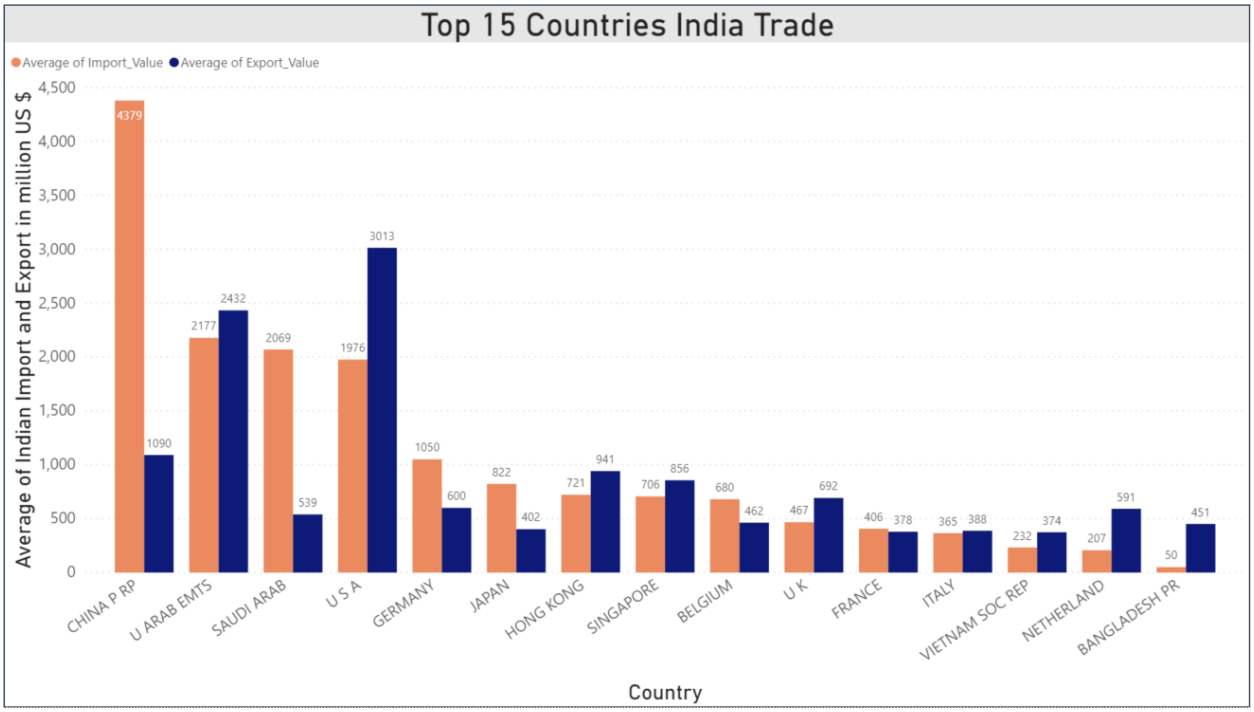

Figure 3-2: Top 15 Countries by average trade

Products corresponding to HS code 27 are the highest contributor for Imports made by India, which includes Minerals Fuels and Mineral Oils specially. India is a country of 130 Crore people, it explains this consumption. For every kind of fuel we're dependent on other countries especially Arab countries, from where we get fuels like Petrol, Diesel, Kerosene etc. If we are to get rid of this vulnerability, Indian government should start investing in identification of new fuel resources and encourage the citizens to use it e.g. electric vehicles, solar panels, windmill etc.

It’s quite surprising to see the HS code 27 on top of list in Exports too. The major reason for that is exchange deals with various countries especially Singapore, to whom India does most export of Mineral and Crude Oils.

Also being one the biggest population who likes to wear jewelries, the Import and Export is second most for HS code 71.

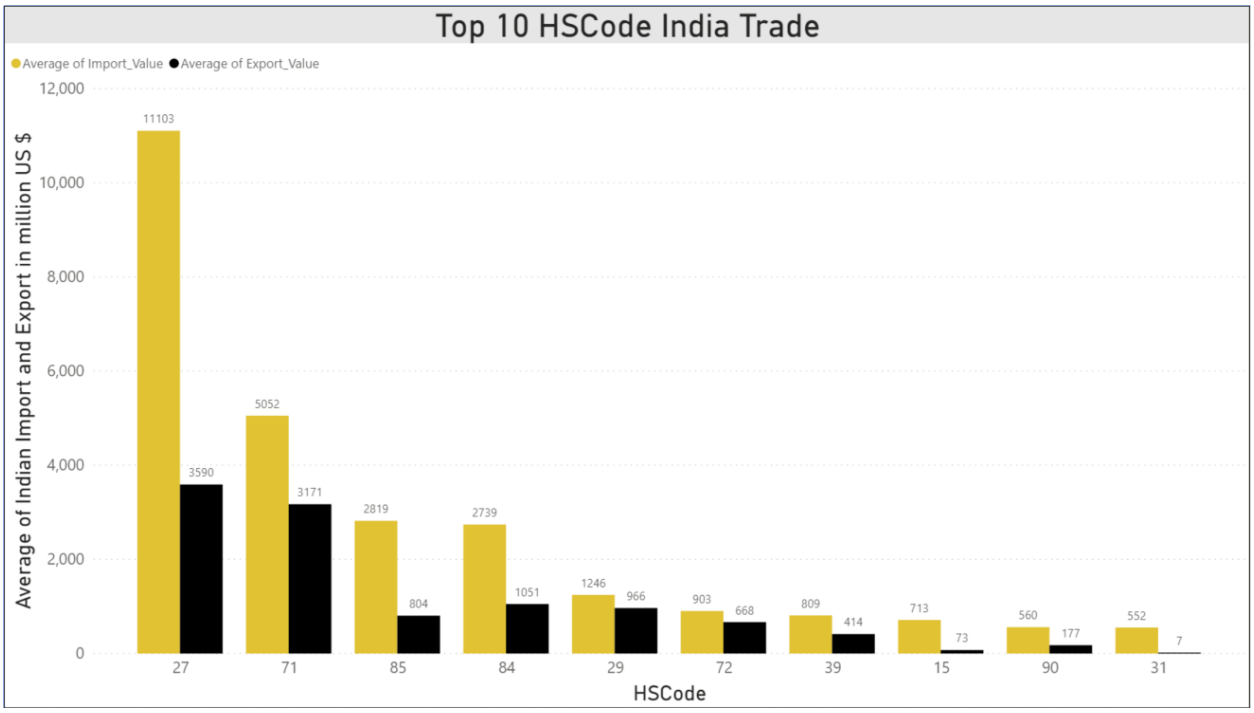

Figure 3-3: Top 10 HS code or products that India trades

The Description of the mentioned HS codes in the graph is as follows -

HSCode	Commodity
15	ANIMAL OR VEGETABLE FATS AND OILS AND THEIR CLEAVAGE PRODUCTS; PRE. EDIBLE FATS; ANIMAL OR VEGETABLE WAXEX.
27	MINERAL FUELS, MINERAL OILS AND PRODUCTS OF THEIR DISTILLATION; BITUMINOUS SUBSTANCES; MINERAL WAXES.
29	ORGANIC CHEMICALS
31	FERTILISERS.
39	PLASTIC AND ARTICLES THEREOF.
71	NATURAL OR CULTURED PEARLS,PRECIOUS OR SEMIPRECIOUS STONES,PRE.METALS,CLAD WITH PRE.METAL AND ARTCLS THEREOF;IMIT.JEWLRY;COIN.
72	IRON AND STEEL
84	NUCLEAR REACTORS, BOILERS, MACHINERY AND MECHANICAL APPLIANCES; PARTS THEREOF.
85	ELECTRICAL MACHINERY AND EQUIPMENT AND PARTS THEREOF; SOUND RECORDERS AND REPRODUCERS, TELEVISION IMA SOUND RECORDERS AND REPRODUCERS,AND PARTS.
90	OPTICAL, PHOTOGRAPHIC CINEMATOGRAPHIC MEASURING, CHECKING PRECISION, MEDICAL OR SURGICAL INST. AND APPA PARTS AND ACCESSORIES THEREOF;

Table 3-4: HS code and corresponding Products/Commodity

3.5 Technologies Used

Python3 – One of the best open source languages for Data Analysis and Visualization

Libraries used –

- Numpy, Matplotlib and Pandas for Basic Data Analysis
- Tensorflow, Keras and Scikit-learn for LSTM Implementation
- Scikit-learn and Statmodels for other statistical model implementation e.g. ARIMA and Holt winters method.

Jupyter Notebook (IDE for Data Analysis and Visualization)

Chapter 4

Implementation of Forecasting

4.1 Types of Timeseries

There are basic two kind of time-series – Multiplicative and Additive. Time series data with varying amplitude is called Multiplicative (Figure 4-1) otherwise it's called Additive (Figure 4-2).

Multiplicative Time Series = Trend * Seasonality * Randomness

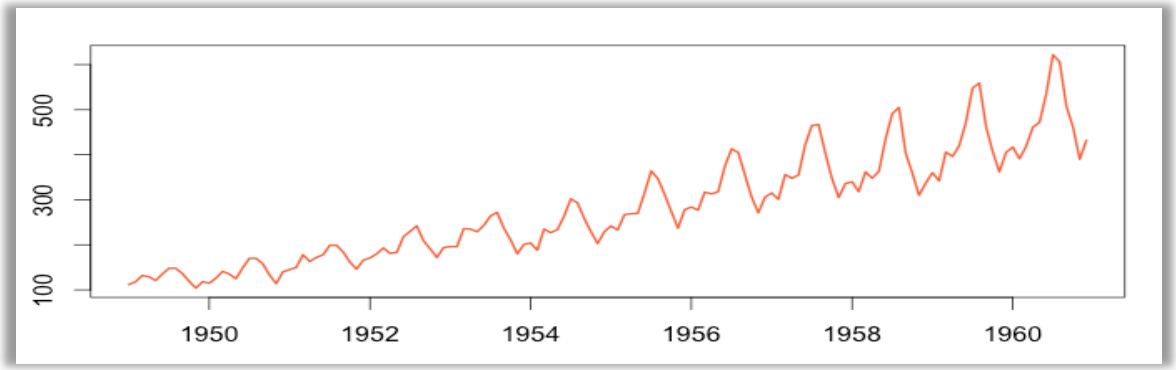

Figure 4-1: Multiplicative Time Series

Additive Time Series = Trend + Seasonality + Randomness

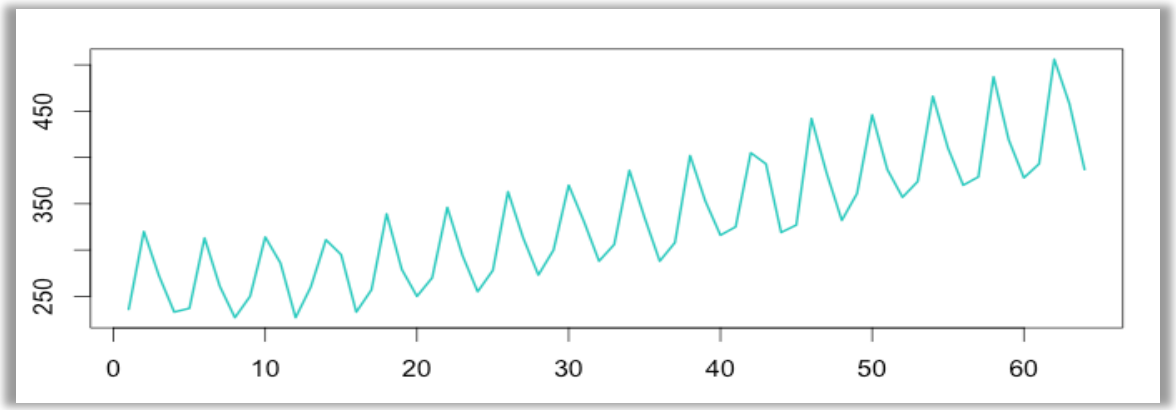

Figure 4-2: Additive Time Series

4.2 Exponential Smoothing(Exponential Averaging)

It is the technique for smoothing(Averaging) the timeseries using exponential window function. In simple moving average the past observation are equally weighted where as here it exponentially decreases over the time. In this method we use

$$s_0 = x_0$$
$$s_t = \alpha x_t + (1-\alpha)s_{t-1},\ t > 0$$

where α is the *smoothing factor*, and $0 < \alpha < 1$.

Where S_0 is the value at time t = 0 and the forecast at time t is given as S_t

The term smoothing factor applied to α here is something of a misnomer, as larger values of α actually reduce the level of smoothing, and in the limiting case with α = 1 the output series is just the current observation.

As mentioned, it uses the exponential window function; we substitute the value of the above equation back to itself.

$$\begin{aligned} s_t &= \alpha x_t + (1-\alpha)s_{t-1} \\ &= \alpha x_t + \alpha(1-\alpha)x_{t-1} + (1-\alpha)^2 s_{t-2} \\ &= \alpha\left[x_t + (1-\alpha)x_{t-1} + (1-\alpha)^2 x_{t-2} + (1-\alpha)^3 x_{t-3} + \cdots + (1-\alpha)^{t-1}x_1\right] + (1-\alpha)^t x_0. \end{aligned}$$

In other words, it forms a Geometric Progression (GP) which is a discrete version of an exponential function.

Code Snippet:

```
from statsmodels.tsa.holtwinters import SimpleExpSmoothing
model_import =
SimpleExpSmoothing(df_train['Import']).fit(smoothing_level=0.6,optimized=False)
yhat_import = model_import.forecast(test_count)
model_export =
SimpleExpSmoothing(df_train['Export']).fit(smoothing_level=0.6,optimized=False)
```

Plot:

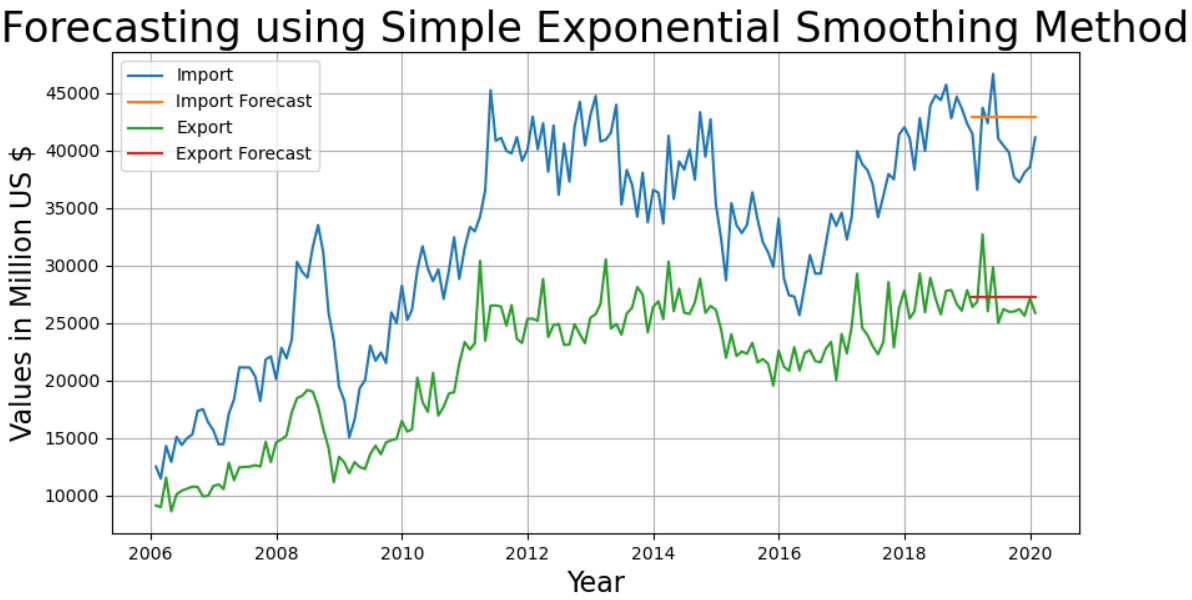

Figure 4-3: Depiction of Import/Export Forecasting using Exponential Smoothing

Result:

RMSE Import/Export - 3771.36, 2042.23

4.3 Auto Regressive Model

It is used when a value from the time series has dependency on previous values e.g. $X_t = f(X_{t-1})$. The order of an auto-regression is the number of immediately preceding values in the series that are used to forecast the current value e.g. order of 2 denotes that the value X_t is dependent on X_{t-1} and X_{t-2}.

Terminologies used – **Auto Correlation Function (ACF) and Partial Autocorrelation Function (PAF)**

ACF = CORR (Y_t , Y_{t-k}) ,

PACF can be calculated as

- Remove the linear dependency from the timeseries
- Calculate the correlation.

PACF is used to find the order of the autoregressive model.

Code Snippet:

```
from statsmodels.tsa.ar_model import AR
from random import random
# Import
# fit model
model = AR(df_train['Import'])
model_fit = model.fit()
# make prediction
yhat_import = model_fit.predict(len(df_train), len(df_train)+test_count-1)
# Export
# fit model
model2 = AR(df_train['Export'])
model2_fit = model2.fit()
# make prediction
yhat_export = model2_fit.predict(len(df_train), len(df_train)+test_count-1)
```

Plot:

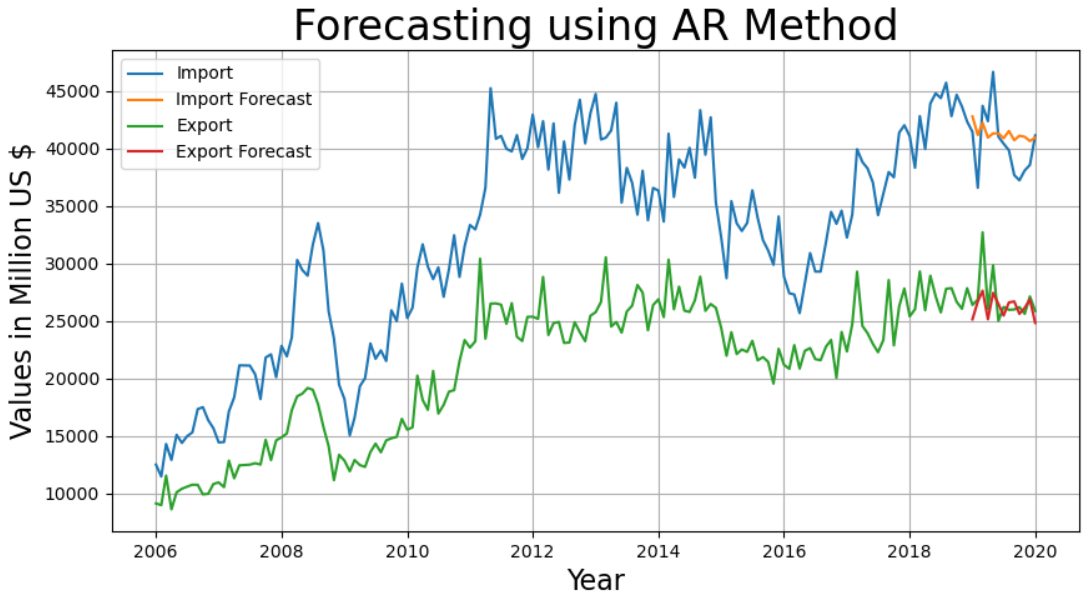

Figure 4-4: Depiction of Import/Export Forecasting using Auto Regressive Model

Result:

RMSE Import/Export - 2718.07, 1738.07

4.4 Moving Average Model

Moving Average is a technique that calculates the overall trend in the dataset. As the name suggests that we take the Average over a fixed rolling size window. The MA_t is calculated by taking the simple mean of the previous window size, which is 4 here (Figure 4-5).

The moving average is thus calculated as –

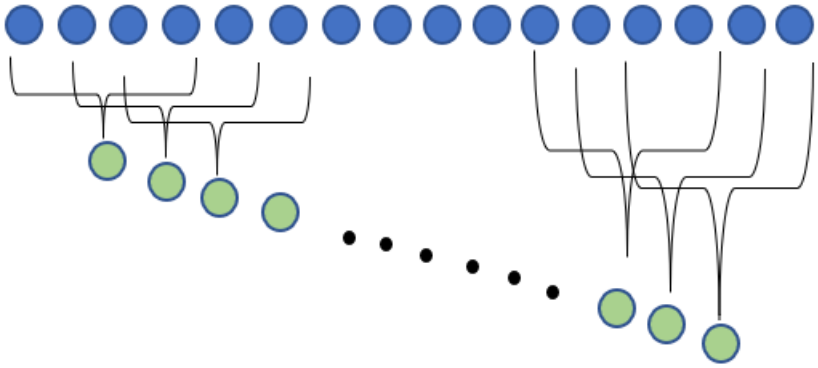

Figure 4-5: Depiction of Moving Average Model working

$MA_4 = X_4 + X_3 + X_2 + X_1$

And the successive value can be calculated as –

$MA_n = MA_{n-1} + (X_n - X_{n\text{-}window\_size})$

We have chosen the window of 12 previous months while forecasting the current month's total trade amount.

Code Snippet:

```
from statsmodels.tsa.arima_model import ARMA
from random import random
# Import
# fit model import
model = ARMA(df_train['Import'], order=(0, 10))
model_fit = model.fit(disp=False)
# make prediction
yhat_import = model_fit.predict(len(df_train), len(df_train)+ test_size-1)
# Export
# fit model export
model2 = ARMA(df_train['Export'], order=(0, 12))
model2_fit = model2.fit(disp=False)
# make prediction
```

Plot:

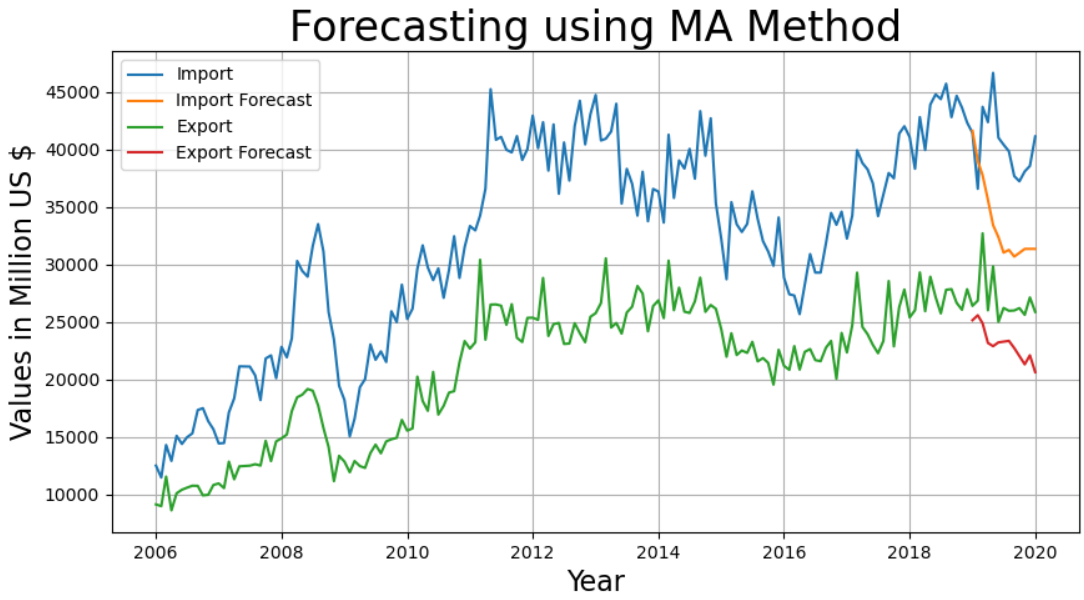

Figure 4-6: Depiction of Import/Export Forecasting using Moving Average Model

Result:

RMSE Import/Export - 7743.73, 4284.66

4.5 Holt-Winters Method

It is the extension over the simple exponential smoothing method. Here we use triple smoothing with the factor - seasonal period, trend type and seasonal type. Here seasonal and trend type means *Multiplicative* or *Additive*.

s_t represents the smoothed value of the constant part for time t. b_t represents the sequence of best estimates of the linear trend that are superimposed on the seasonal changes. c_t is the sequence of seasonal correction factors. c_t is the expected proportion of the predicted trend at any time t mod L in the cycle that the observations take on. As a rule of thumb, a minimum of two full seasons (or 2L periods) of historical data is needed to initialize a set of seasonal factors.

The output of the algorithm is again written as F_{t+m}, an estimate of the value of x at time t+m, m>0 based on the raw data up to time t. Triple exponential smoothing with multiplicative seasonality is given by the formulas

$$
\begin{aligned}
s_0 &= x_0 \\
s_t &= \alpha \frac{x_t}{c_{t-L}} + (1-\alpha)(s_{t-1} + b_{t-1}) \\
b_t &= \beta(s_t - s_{t-1}) + (1-\beta)b_{t-1} \\
c_t &= \gamma \frac{x_t}{s_t} + (1-\gamma)c_{t-L} \\
F_{t+m} &= (s_t + mb_t)c_{t-L+1+(m-1) \mod L},
\end{aligned}
$$

Where α is the data smoothing factor, $0 < \alpha < 1$, β is the trend smoothing factor, $0 < \beta < 1$, and γ is the seasonal change smoothing factor, $0 < \gamma < 1$. Reference

In our model we used seasonal period = 12 and trend type and seasonal type as multiplicative.

Code Snippet:

```
from statsmodels.tsa.holtwinters import ExponentialSmoothing
model_import = ExponentialSmoothing(df_train['Import'] ,seasonal_periods=12 ,trend='mul',
seasonal='mul',).fit()
yhat_import = model_import.forecast(test_size)
model_export = ExponentialSmoothing(df_train['Export'] ,seasonal_periods=12 ,trend='mul',
seasonal='mul',).fit()
yhat_export = model_export.forecast(test_size)
```

Plot:

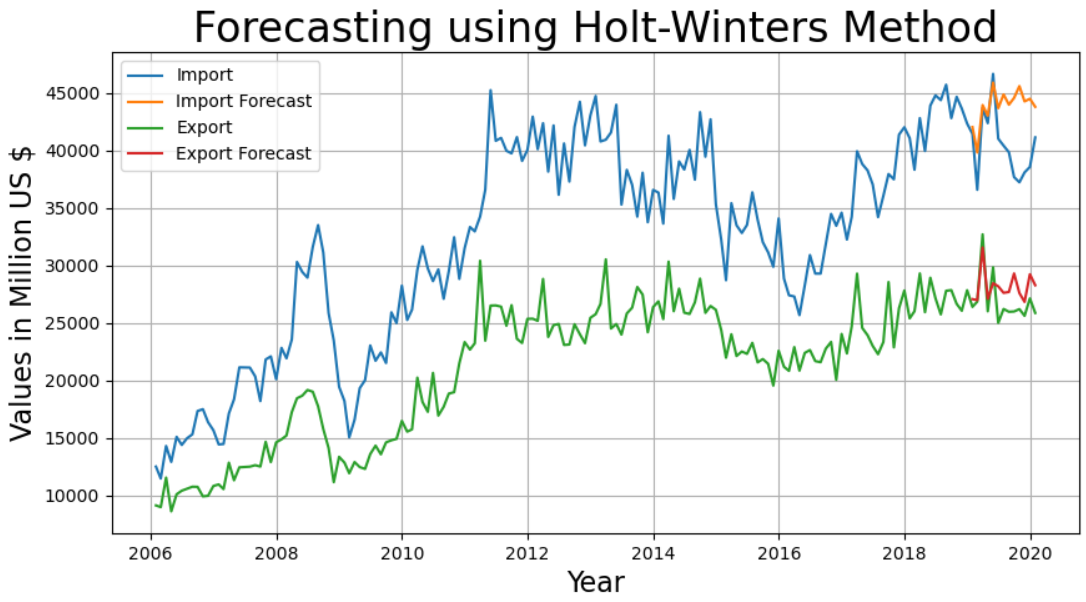

Figure 4-7: Depiction of Import/Export Forecasting using Holt Winter Model

Result:

RMSE Import/Export - 4417.84, 1850.49

4.6 ARIMA Multiplicative

ARIMA stands for *Autoregressive Integrated Moving Average* which is a combination of three terms –

The AR part of ARIMA indicates the evolving variable of interest is regressed on its own lagged (i.e., prior) values. The MA part indicates that the regression error is actually a linear combination of error terms whose values occurred contemporaneously and at various times in the past. The I, which stands for *Integrated*, indicates that the data values have been replaced with the difference between their values and the previous values (and this differencing process may have been performed more than once). The purpose of each of these features is to make the model fit the data as well as possible.

Non-seasonal ARIMA models are generally denoted ARIMA(p, d, q) where parameters p, d, and q are non-negative integers, p is the order (number of time lags) of the autoregressive

model, d is the degree of differencing (the number of times the data have had past values subtracted), and q is the order of the moving-average model.

We have already seen the Autoregressive and Moving average individually.

Multiplicative Time Series = Trend * Seasonality * Randomness, As explained previously.

For predicting the p and q we use partial autocorrelation and autocorrelation function. We used plots to find the optimal value of p and q graphically.

Here we are taking the (p, d, q) = (1,0,5). d as 0 because we do not require the differencing as our randomness is stationary.

Code Snippet:

Import:

```
from statsmodels.tsa.arima_model import ARIMA
# fit model
model = ARIMA(df['Randomness_Import'][:-test_size], order=(1, 0, 5))
model_fit = model.fit(disp=False)
# make prediction
yhat = model_fit.predict(len(df[:-test_size]), len(df[:-test_size]) + test_size-1)
df['ARMA_import_forecast'] = yhat
df['ARMA_import_forecast'] = df['ARMA_import_forecast'] + df['Seasonal_Import'] +
df['MA_Import']
```

Export:

```
from statsmodels.tsa.arima_model import ARIMA
# fit model
model = ARIMA(df['Randomness_Export'][:-test_size], order=(1, 0, 5))
model_fit = model.fit(disp=False)
# make prediction
yhat = model_fit.predict(len(df[:-test_size]), len(df[:-test_size]) + test_size-1)
df['ARMA_export_forecast'] = yhat
df['ARMA_export_forecast'] = df['ARMA_export_forecast'] + df['Seasonal_Export'] +
df['MA_Export']
```

Plot:

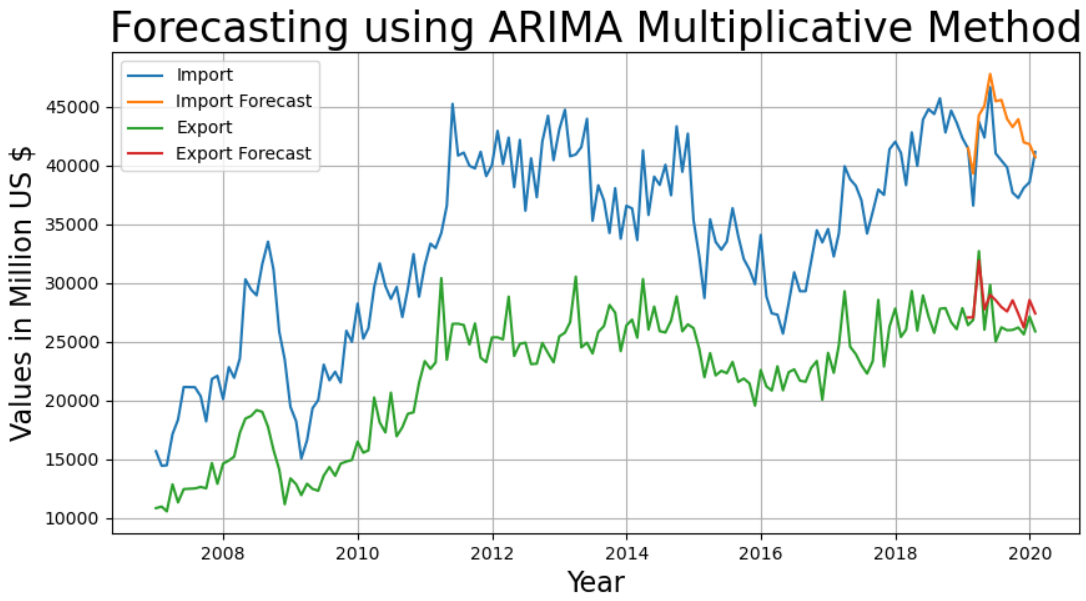

Figure 4-8: Depiction of Import/Export Forecasting using ARIMA Multiplicative Model

Result:

RMSE Import/Export - 3286.15, 2026.4

4.7 ARIMA Additive

This is similar to the above method except for the fact that we have considered our time series as Additive (just for comparison).

Additive Time Series = Trend + Seasonality + Randomness

Here we are taking the (p, d, q) = (1,0,5). d as 0 because we do not require the differencing as our randomness is stationary.

We can see that the forecast does not fit right.

Code Snippet:

Import:

```
from statsmodels.tsa.arima_model import ARIMA
# fit model
model = ARIMA(df['Randomness_Import'][:-test_size], order=(2, 0, 5))
model_fit = model.fit(disp=False)
# make prediction
yhat = model_fit.predict(len(df[:-test_size]), len(df[:-test_size]) + test_size-1)
df['ARMA_import_forecast'] = yhat
df['ARMA_import_forecast'] = df['ARMA_import_forecast'] * df['Seasonal_Import'] *
df['MA_Import']
```

Export:

```
from statsmodels.tsa.arima_model import ARIMA
# fit model
model = ARIMA(df['Randomness_Export'][:-test_size], order=(1, 0, 5))
model_fit = model.fit(disp=False)
# make prediction
yhat = model_fit.predict(len(df[:-test_size]), len(df[:-test_size]) + test_size-1)
df['ARMA_export_forecast'] = yhat
df['ARMA_export_forecast'] = df['ARMA_export_forecast'] * df['Seasonal_Export'] *
df['MA_Export']
```

Plot:

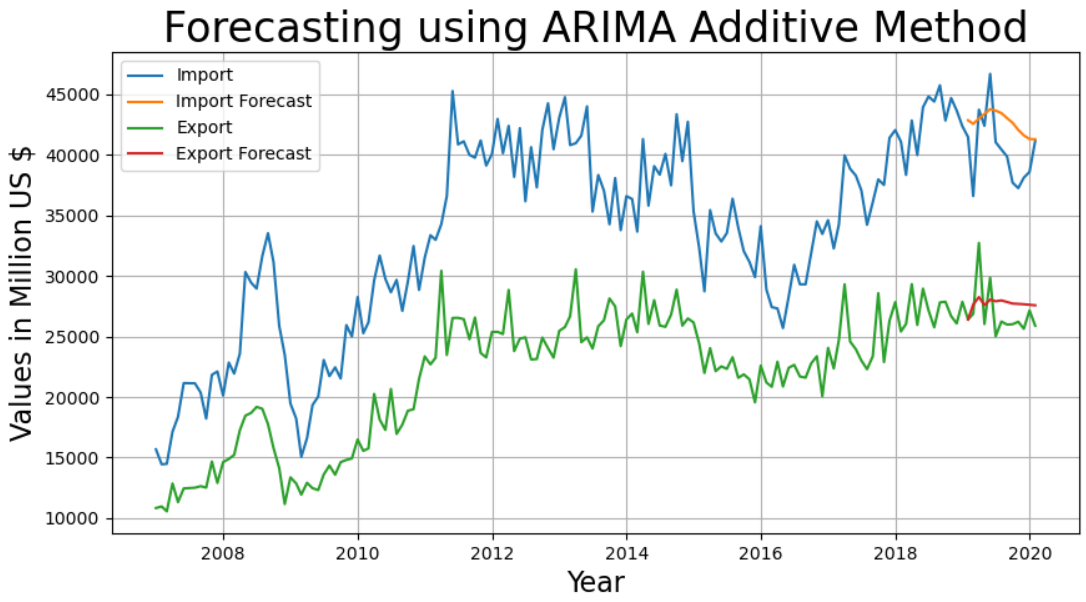

Figure 4-9: Depiction of Import/Export Forecasting using ARIMA Additive Model

Result:

RMSE Import/Export - 3739.46, 1646.06

4.8 Seasonal ARIMA

Seasonal ARIMA here is ARIMA Multiplicative method with a seasonality factor m. Here we have chosen the multiplicative factor of 12 as ours is a monthly data.

Also, we used brute force to find the best p, d, q values which reduces the RMSE. Import (1,0,0) and Export (1,0,2). This model has performed quite well and thus below is the plot.

Code Snippet:

```
import statsmodels.api as sm
model_import = sm.tsa.statespace.SARIMAX(df_train['Import'], order=(1, 0,
0),seasonal_order=(0,1,1,12), trend='n').fit()
yhat_import = model_import.forecast(test_size)
model_export = sm.tsa.statespace.SARIMAX(df_train['Export'], order=(1, 0,
2),seasonal_order=(0,1,1,12), trend='n').fit()
yhat_export = model_export.forecast(test_size)
```

Plot:

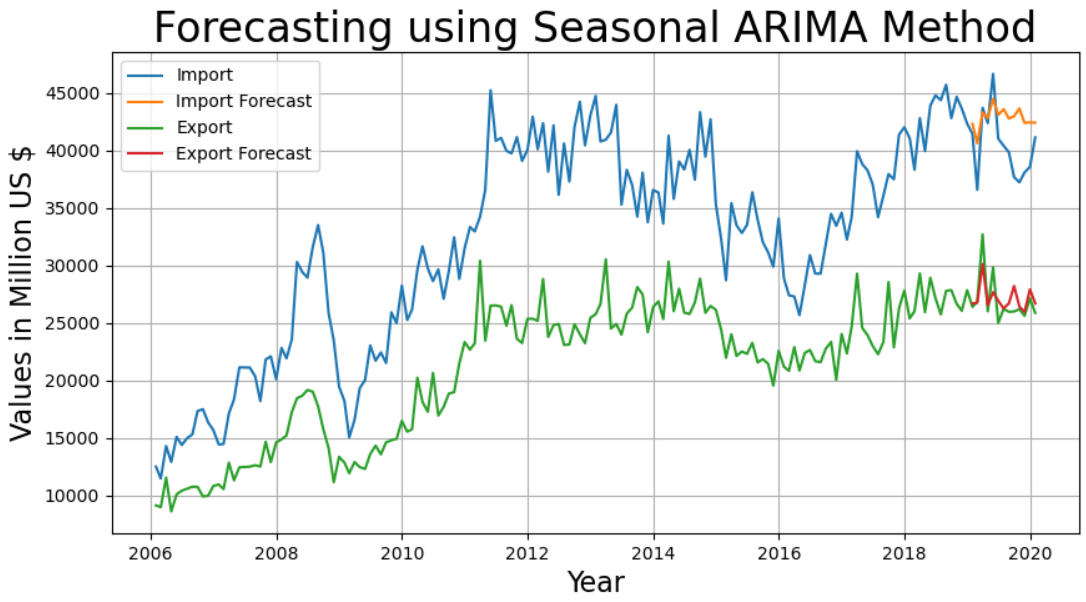

Figure 4-10: Depiction of Import/Export Forecasting using Seasonal ARIMA Model

Result:

RMSE Import/Export - 3391.32, 1310.58

4.9 RNN (Recurrent Neural Networks)

Humans don't start their thinking from scratch every second. As we read this paragraph, we understand each word based on your understanding of previous words. We don't throw everything away and start thinking from scratch again. Our thoughts have persistence.

Traditional neural networks can't do this, and it seems like a major shortcoming.

Recurrent neural networks address this issue. They are networks with loops in them, allowing information to persist (Figure 4-11).

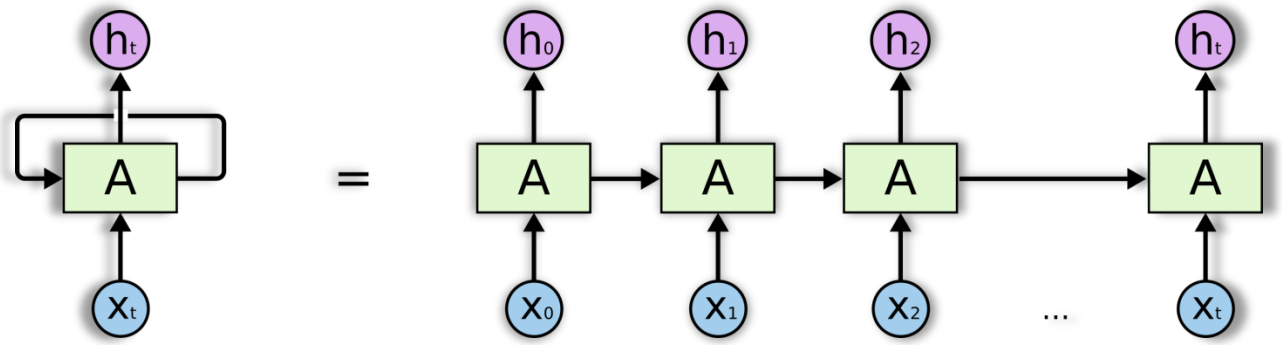

Figure 4-11: Structure of RNN Models

Sometimes, we only need to look at recent information to perform the present task. In such cases, where the gap between the relevant information and the place where prediction is needed, is small, RNNs can learn to use the past information.

But there are also cases where we need more contexts. Consider trying to predict the last word in the text "I grew up in France… I speak fluent French." It's entirely possible for the gap between the relevant information and the point where it is needed to become very large. Unfortunately, as that gap grows, RNNs become unable to learn to connect the information.

Thankfully, LSTMs Solves this problem!

4.9.1 LSTM Networks

Long Short Term Memory networks – usually just called "LSTMs" – are a special kind of RNN, capable of learning long-term dependencies. They were introduced by Hochreiter & Schmidhuber (1997), and were refined and popularized by many people in following work.

LSTMs are explicitly designed to avoid the long-term dependency problem. Remembering information for long periods of time is practically their default behavior, not something they struggle to learn!

All recurrent neural networks have the form of a chain of repeating modules of neural network. In standard RNNs, this repeating module will have a very simple structure, such as a single tanh layer (Figure 4-12).

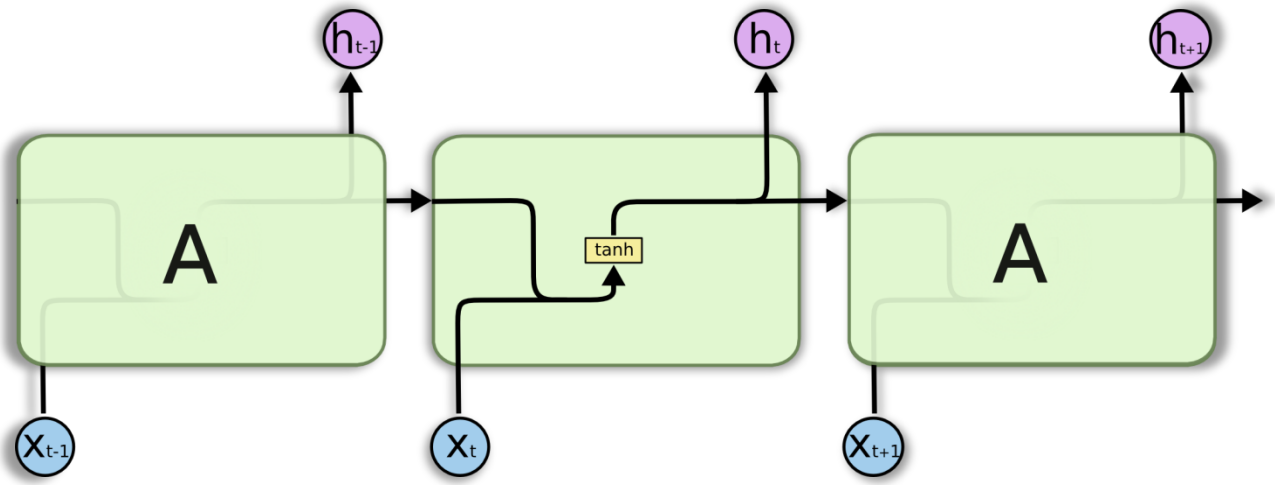

Figure 4-12: Micro level structure of RNN Models

LSTMs also have this chain like structure, but the repeating module has a different structure. Instead of having a single neural network layer, there are four, interacting in a very special way (figure 4-13).

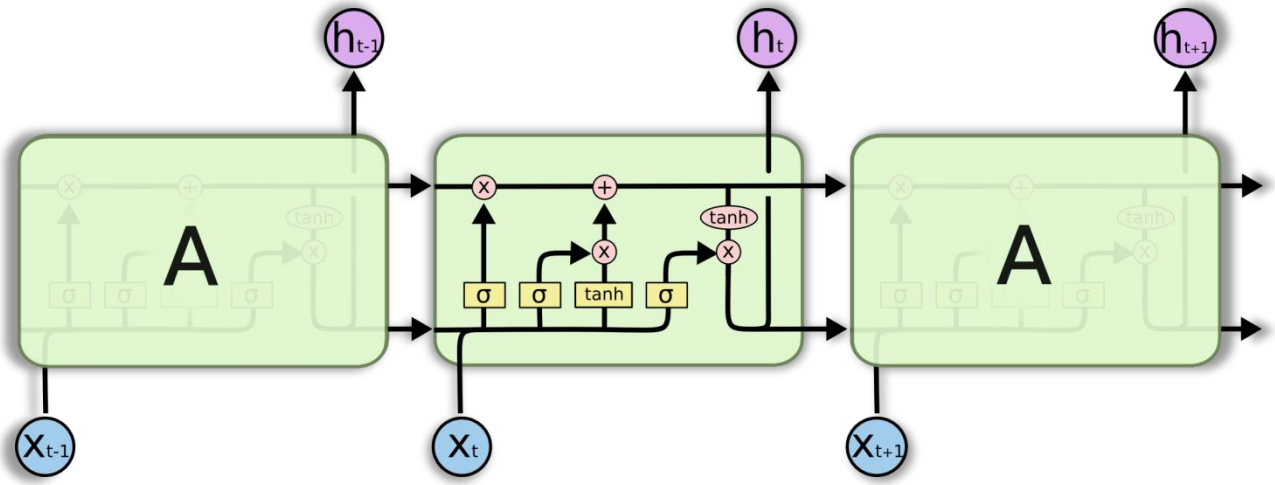

Figure 4-13: Micro level structure of LSTM Models

The key to LSTMs is the cell state, the horizontal line running through the top of the diagram.

The cell state is kind of like a conveyor belt. It runs straight down the entire chain, with only some minor linear interactions. It's very easy for information to just flow along it unchanged (figure 4-14).

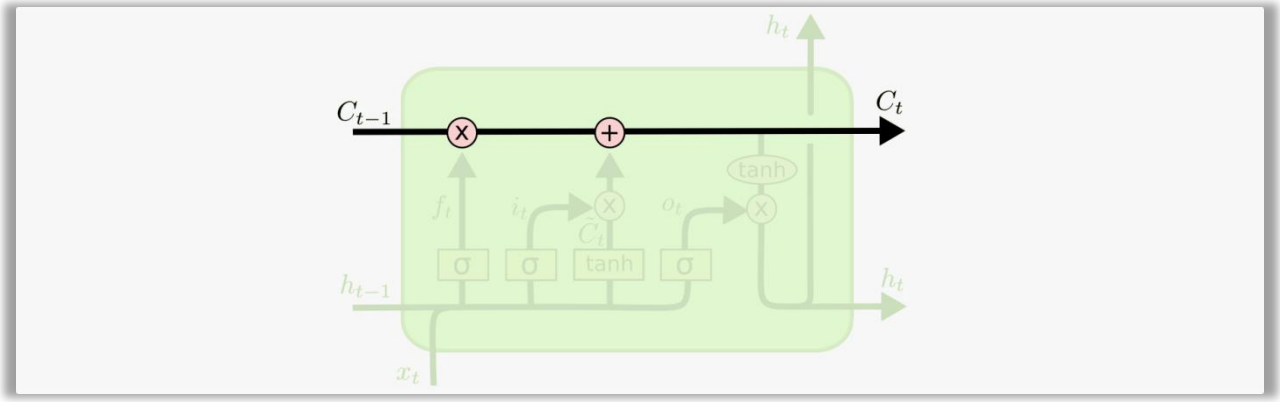

Figure 4-14: A global state passing through LSTM cell

The LSTM does have the ability to remove or add information to the cell state, carefully regulated by structures called gates.

Gates are a way to optionally let information through. They are composed out of a sigmoid neural net layer and a pointwise multiplication operation.

Preparing Data for applying LSTM

Scale data values between 0 and 1 for faster convergence using gradient descent. Prepare input for each of data point as set of previous 12 data point, e.g. for 13th data point a_{13}, the input will be set (a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}) and the output will be a_{13}. Store all these input sets in an input list and outputs in output list.

LSTM Model Specification

We have four layers model, which is arranged as following:

1. LSTM layer of 200 nodes
2. LSTM layer of 200 nodes
3. LSTM layer of 150 nodes
4. Dense layer with 'linear' activation.

After defining our model, we compile it using mean-squared-error as loss function and Adam optimizer. We didn't need to add dropout or regularization because we already scaled or normalized our data, avoiding overfitting.

Now we fit our model for prepared input and output with batch size of 12 for 30 epochs.

Then we forecast trade amount for next 12 months one by one, however we can do this in one go as well. These predicted values will be scaled in between 0 and 1 hence we need to inverse transform these values to get actual value for the actual total trade amount corresponding to that month.

Code Snippet:

```
#converting dataset into x_train and y_train
scaler = MinMaxScaler()
smoothing_window_size = 24
for di in range(0, 156, smoothing_window_size):
    scaler.fit(train[di:di+smoothing_window_size, :])
    train[di:di+smoothing_window_size, :] =
scaler.transform(train[di:di+smoothing_window_size, :])

# Normalize test data
valid = scaler.transform(valid)

x_train, y_train = [], []
for i in range(no_of_sig_days, len(train)):
    x_train.append(train[i-no_of_sig_days: i, :])
    y_train.append(train[i, 0])

# building neural networks and adding layers
model = Sequential()
model.add(LSTM(200, input_shape=(x_train.shape[1], 1), return_sequences=True))
model.add(LSTM(200, input_shape=(x_train.shape[1],1), return_sequences=True))
model.add(LSTM(150, input_shape=(x_train.shape[1],1), return_sequences=False))
model.add(Dense(1,activation='linear'))

# compile model
model.compile(loss='mean_squared_error', optimizer='Adam')

from keras.callbacks import EarlyStopping, ModelCheckpoint
early_stop = EarlyStopping(monitor='val_loss', patience=10, mode='min')
checkpoint = ModelCheckpoint(mode + '_model_best_weight_last_13preds_comp.h5',
monitor='val_loss', save_best_only=True, mode='min', period=1)
model.fit(x_train, y_train, epochs=30, batch_size=4, callbacks=[early_stop, checkpoint],
verbose=1, validation_data=(x_test, y_test)) # verbose=1 shows us the progress of epochs

# make predictions
pred = model.predict(x_test, batch_size=1)
pred = scaler.inverse_transform(pred)
```

Plot:

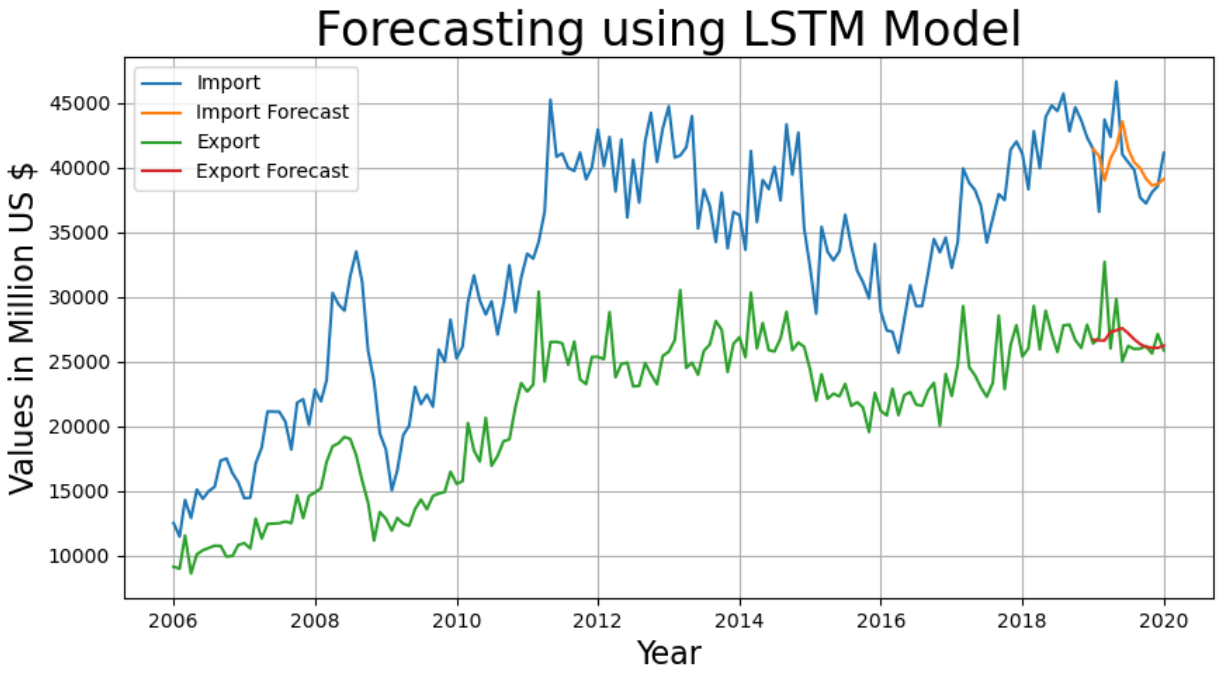

Figure 4-15: Depiction of Import/Export Forecasting using LSTM Model

Result:

RMSE Import/Export = 2637.71, 2043.39

4.10 Comparison among Models

All the models perform quite well but the best observation which we have observed is of Seasonal ARIMA and LSTM.

Below is the RMSE Comparisons –

Model\Error	Import RMSE	Export RMSE
Exponential Smoothing	3771.36	2042.23
Auto Regressive (AR) model	2718.07	1738.07
Moving Average (MA) model	7743.73	4284.66
Holt-Winters Method	4417.84	1850.49
ARIMA Multiplicative	3739.46	1646.06
ARIMA Additive	3286.15	2026.4
Seasonal ARIMA	3391.32	**1310.58**
LSTM	**2637.71**	2043.39

Table 4-1: Comparison between All Models used in the thesis

The comparisons are so close because one model is the extension of the other with some advancement. Holt-winters is the advancement of the Simple exponential smoothing. ARIMA is the combination of AR and MA model, Seasonal ARIMA is the advancement of the ARIMA model. Only LSTM is altogether different model, which gives us best results for Import Trade Forecasting while ARIMA stood up for Export Trade Forecasting.

- To get more accurate forecasting we should fine tune our model's parameter.
- Our models are able to forecast right trend of trade Import/Export amount among which ARIMA and LSTM stand out.
- As we have only 169 data points (156 training + 13 validations), it's too much to ask for a very high accuracy.
- More data points will improve our model's accuracy.

Chapter 5

Case Study: Getting Rid of Trade Deficit

Problem: India has maintained an overall trade deficit since 1990 and in the past decade, the deficit has increased manifold. For a developing country like India, the current account deficit occupies the center stage in policy discussions, as the persistent discrepancies in current account and rising levels of trade deficit pose risks to the sustainability of high economic growth and macroeconomic stability. (Tiwari 2010)

Reasons: While there's no doubt that India has the potential to gain momentum in terms of exports but India's negligent export of manufactured goods is a vital cause of the trade deficit. China has blocked India's exports to a large extent. This is because China has cheaper labor than our country and also much better Infrastructure. Countries wanting to import look towards cheaper options, for which China is a ready and willing source.

Chances in Near Future: If India is to get out of trade deficit in near future then we'd have to make many reforms in the field of economy as well as trade. It'll be beneficial if we can modernize our agriculture, which is significant for a healthy economy with trade surplus. Some of suggested reforms by (Bansal 2020) are following –

- Last year, the US, after realizing that there exists a huge trade deficit in the country's exports with regard to India, had introduced several tariffs on Indian exports. India can do the same for cheap Chinese products.
- The US has imposed wide trade sanctions on China and India has the perfect opening to extract a favorable deal from the US.
- India can no longer support its trade on raw materials, and for this the 'Make in India' initiative needs to be made a priority.
- With the relaxation of corporate taxes, India can expect a growth in investments. Finance Minister Nirmala Sitharaman has slashed the basic corporate tax rate for domestic companies to 22 percent instead of the previous 30 percent.
- India also needs to concentrate on the service sector. Service sector exports have been booming since the past one decade, primarily due to the influx of the IT sector.

While the problem areas have been identified, it is to be noted that all the three sectors, agricultural, manufacturing and service, need to grow independently of each other and not at the cost of the other.

Conclusion

Throughout the thesis, we found answers of all of our motivational questions mentioned in **Abstract**. So we have all the insights and many forecasting models in our hands. And we have also forecasted the Near Future Trade value with our best models. The results are quite satisfactory and we see that there is an upwards trend in Import forecast and slightly upwards trend in the Export forecast. The more we export, and the lesser we import, is beneficial for the Indian economy. This shows that India will be leading in terms of trade, if we modify our foreign trade policy according to the case study.

Also since we're in huge trade deficit with China in terms of trade, we are too much dependent on raw electronic products imported from Chine and this is not different for other developing countries too. We need to make strict policies towards Chinese products if we want to have any hope about profiting from trade.

About Our Forecasting Models: We used 8 time series forecasting models. LSTM model performed better for Import Data as expected but surprisingly Seasonal ARIMA outperformed LSTM in Forecasting Export Data, which we didn't expect at the beginning of this project.

We found out that ARIMA captures seasonality much better than any other model, even with **lesser data and randomness.** First few validation data points may have caused LSTM to underperform since the trend goes down steeply. Also for any Neural Network, we need a lot of data points to get decent enough results, and this explains a lot. **Overall all of our models' performance gave good results among which Seasonal ARIMA and LSTM performed better than other models.**

We hope that our successors will find this thesis useful and can apply our methods without trouble. We're open-sourcing our codes and data, which we used for forecasting and analysis and it'll be available on GitHub.

GitHub Link: https://github.com/akarshsomani/Indian-Trade-Data-Analysis-and-Forecasting

Future Prospects

- Making Existing Models more robust by tuning them even more.
- Using new models specially Attention based models which stores more information.
- Including some factor to encounter the randomness in data which causes high frequency fluctuations which in turn results in below expected performance of models.
- Getting more data points for better forecasting specially for LSTM model since it operates on a neural network which needs more and more data for better predictions.
- Using this thesis for even more case studies similar to the one we discussed above.

Bibliography

Bansal, Vivek. "Plugging India's trade deficit: How to give exports a leg up." *CNBCTV*, 2020.

Chatfield, C. "The Holt-Winters Forecasting Procedure." *ResearchGate*, 1978.

Lyashenko, Oksana. "The Application of the ARIMA-models for forecasting the Dynamics of Foreign Trade of Ukraine." *ResearchGate*, 2020.

Nimisha Tomar, Durga Patel, Akshat Jain. "Air Quality Index Forecasting using Auto-regression Models." *ResearchGate*, 2020.

S. Vemuri, R. Balasubramanian, E.F. Hill. "Load Forecasting using Moving Average Stochastic Models." *ResearchGate*, 1974.

Sandip Roy, Sankar Prasad Biswas, Subhajyoti Mahata, Rajesh Bose. "Time Series Forecasting using Exponential Smoothing to Predict the Major Atmospheric Pollutants." *ResearchGate*, 2018.

Tiwari, Aviral. "Is trade deficit sustainable in India? An inquiry." *ResearchGate*, 2010.

Yi-Ting Tsai, Yu-Ren Zeng, Yue-Shan Chang. "Air Pollution Forecasting Using RNN with LSTM." *ReasearchGate*, 2018.