NIIT University, Neemrana, Rajasthan-301705
July 2020

End semester report on R & D Project (NU 302)

Academic Year- 2019-20

on

Indexation of Commodity Derivatives

A dissertation
Submitted in partial fulfillment of the requirements for the award of the degree
Bachelor of Technology

by

Jatin Kumar Jain	**BT17GCS159**	**CSE**
Sanya Kapoor	**BT17GCS098**	**CSE**
Sakshi Dhawan	**BT17GCS097**	**CSE**
Vista Vincent	**BT17GCS150**	**CSE**
Nikhil Khairnar	**BT17GCS243**	**CSE**

Under supervision of
Prof. Gyanesh Jain

CERTIFICATE BY SUPERVISOR

This is to certify that the present R&D project entitled Indexation of Commodity Derivatives being submitted to NIIT University, Neemrana, in partial fulfilment of the requirements for the award of the Degree of Bachelor of Technology, in the area of BT/CSE/ECE/GIS, embodies faithful record of original research carried out by Sakshi Dhawan, Vista Vincent, Sanya Kapoor, Jatin Kumar Jain and Nikhil Khairnar. They have worked under my guidance and supervision and that this work has not been submitted, in part or full, for any other degree or diploma of NIIT or any other University.

Place: NIIT University

Name of the Supervisor with signature

Date: 15th July 2020

NU
NIIT UNIVERSITY
THE UNIVERSITY OF THE FUTURE
Industry-linked | Technology-based | Research-driven | Seamless

DECLARATION BY STUDENTS

We hereby declare that the project report entitled Indexation of Commodity Derivatives which is being submitted for the partial fulfilment of the Degree of Bachelor of Technology, at NIIT University, Neemrana, is an authentic record of our original work under the guidance of Prof. Gyanesh Jain. Due acknowledgements have been given in the project report to all other related work used. This has previously not formed the basis for the award of any degree, diploma, associate/fellowship or any other similar title or recognition in NIIT University or elsewhere.

Place: NIIT University

Date: 15th July 2020

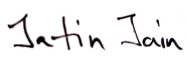
Jatin Kumar Jain	BT17GCS159	CSE	
Sanya Kapoor	BT17GCS098	CSE	
Nikhil Khairnar	BT17GCS243	CSE	
Vista Vincent	BT17GCS15	CSE	
Sakshi Dhawan	BT17GCS097	CSE	

Table of Contents

1. Introduction

Financial markets are fettered to be inclusive, and if such inclusiveness can be brought about via the availability of various products meeting various needs of the participants, it will encourage sustainable growth of the markets and hence be efficient. Out of all the ways available to include commodities as an asset in one's portfolio, the 'index way' stands out the most due to its simplicity and its ability to give traders an insight into the price and performance of a given collection of commodities that are appropriately weighed. The collection may be broad-based, consisting of a variety of commodities across sectors, or can consist of a few commodities that track a particular sector.

Benefits of having an index:

The commodity indices reflect the general trend in commodity prices and hence indicate the fundamentals of the wider commodity economy or the commodity sector they describe. The indices can also be used for benchmarking, in order to assess the performance of an investment portfolio consisting of the particular commodity components.

Why trade in commodity indices?

The risk of trading an index futures contract is comparatively lesser than for trading a single commodity futures contract. One can draw parallels to the equity market where trading with index funds provides a safer alternative to trading with a singular asset. It is more difficult to make an index turbulent and unstable compared to an individual equity asset.

After the introduction of commodity markets to institutional investors beginning with June 2017, India has already seen its initial participation of category 3 Investment Funds. Soon, mutual funds and portfolio management services providers and hedge fund managers may become an active and integral part of the commodity derivative

ecosystem. Commodity index captivates the essence of performance in the commodities without the complexity of understanding the details of each commodity. Introducing commodity indices trading will present investors new commodity index-based products with notable benefits, including diversification, low costs, competitive performance, and, most importantly, the primitiveness to participate in commodity markets, which in turn will encourage more and more people to get into this thinly traded segment of the Indian market.

Challenges in building a tradeable index in Commodity Derivative Segment:

1. Estimating the Market Cap:

In the equity market, the market cap of companies and their floating stock plays a major role in deciding the weights that they enjoy in the index script. However, there are no such measurements in the commodity market. For example, the market cap or size of the wheat market can only be based on an estimate, there are a lot of factors that come into play, the classification into various brackets as well as the quality and the crop size. The Final crop estimates come very late.

2. Variable pricing of the commodity:

Even if the market cap is somehow estimated, crop size multiplied by the Minimum Selling Price is not a viable indicator as commodities are traded above or below the MSP.

3. Thinly traded market segment:

The commodity derivative market has low liquidity compared to the equity market. This leads to unstable trends in the market.

The existing commodity indices in India:

MCX and iCOMDEX: (Non-tradeable index)(Uses Static Weights)

Thomson Reuters and MCX collaboratively launched their commodity index series named Thomson Reuters-MCX India Commodity Indices (iCOMDEX), tracing the performance of commodities listed on the MCX exchange. collection includes a composite index comprising of 11 commodities, sector-based indices (bullion and base metals), and single commodity indices for gold, copper, and crude oil. The methodology used to construct these indices observes the International Organisation of Securities Commissions (IOSCO) guidelines for standards. Every existing commodity in the index is weighted two-thirds by its liquidity and one-third by its domestic physical market size.

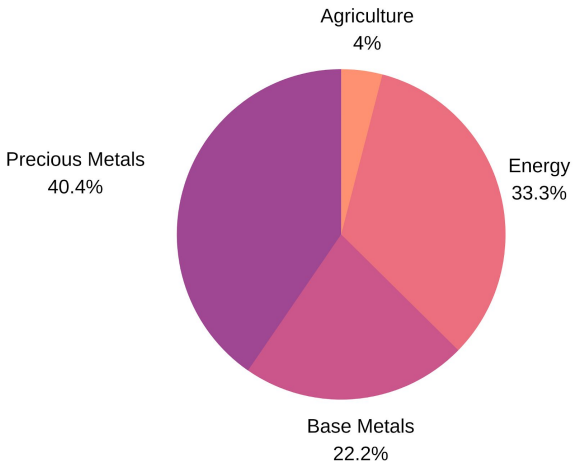


Figure 1: iCOMDEX Component Weights

The MCX composite index has only two Agri commodities with a total weight amounting to less than 5%, while crude oil is one of the heaviest commodities accounting for almost 34% weight. Hence the index may be more representative of the global markets than of the Indian commodity sector.

The problem with static weight assignment for index calculation:

The static weight assignment for calculating the commodity index disregards the trends and the seasonality of its various constituents. For example,the contribution of the energy sector could be higher during summers, agriculture sector could be booming during certain months and could be down in others. These almost repeating patterns throughout the year are completely disregarded with static weight assignments. A methodology that

assigns weights to the constituents commodities dynamically depending on the historical patterns could be a closer approximation of a true market indicator.

2. Problem Statement:

Indexation of commodity derivatives in India

Using artificial intelligence, develop an index for commodity derivatives that can be traded and the trade values should be a reflection of the market sentiments.

Description: The research aims at building a commodity index that could act as a true market indicator keeping in mind the seasonality and trends of the various constituents making up the index.The research looks into a neural network-based weight assignment methodology trained over historical data to build a commodity index that is sensitive to yearly patterns and could be a closer approximation to real market sentiments. If successful the index could potentially be used as a barometer to market sentiments and be used as a tradeable index.

3. Review of literature

With the huge amount of financial market data available, financial time-series can be analyzed towards the prediction of market trends. Time series data is very useful to get the trends that can be used to train a neural network. [1]

Financial market movement direction can be predicted through DNN where we backtest a simple trading strategy over 43 different commodity FX future mid-prices at a 5-minute interval.[2]

Predictable patterns in the implied volatility of commodity options can be analyzed to a certain degree of accuracy using latent factor and parametric frameworks[3]

The behavior of successful traders can be replicated through DLNN which will possibly perform better than the human trader in limit-order-book financial market.[4]

The empirical features of seasonality in the agriculture segment of the commodity market can be estimated to a certain degree of accuracy using a multi-factor stochastic volatility model.[5]

Deep Learning models significantly outperform traditional Machine Learning counterparts for financial time series forecasting.[6]

Implementation of machine learning algorithms with the very famous BN-S(Barndorff-Nielsen and Shephard) model can add long-term dependence which it usually lacks, enabling us to predict trends or fluctuations by incorporating only 1 extra factor - theta. [7]

RNN structures are superior and versatile over various other estimators for trend detection in time series.[8]

Increasing import restrictions can increase the commodity output of exports when labour, capital and domestic factors are complements in an economy. [9]

Predicting commodities become extremely important for the government to estimate market factors in advance to take efficient policy measures.[10]

Through Time-series clustering we can track the mechanism that generates the time series which eventually helps us to predict future values of the given time-series. The clustering of time series is categorized depending upon whether they work directly or indirectly on the data or features extracted from data. There are various clustering approaches viz Wavelet Transform, Hough Transform, Haar wavelet transform, Dynamic Time Warping(DTW), etc. and selection of these approaches purely depends on the data. For Non-stationary signals, DTW is good as it uses K-medoids based Genetic clustering or k-means clustering algorithms.[11]

An Integrated system where wavelet transforms and recurrent neural networks based on an artificial bee colony algorithm(ABC) can be combined for stock price forecasting. where haar wavelet is used to decompose and deNoise the stock time series where ABC algorithm is used to optimize the RNN weights and biases. The paper recorded an outstanding performance of the wavelet-based processing model but when it comes to more complex time-series data then it showed some insufficiencies.[12]

When raw time series prediction without wavelet deNoising is compared with time series Prediction with wavelet deNoising where prediction is done on FeedForward neural Network it turns out that the effect of wavelet deNoising is not satisfactory.[13]

It is known that Dynamic Time Warping is superior when it comes to the clustering of large time-series data compared to Euclidean distance. The paper purposes the approach that can reduce CPU time of DTW[14]

Massive Time Series Dataset clustering can be done by DTW as any time algorithm. Where the algorithm was centered around a novel data-adaptive approximation to DTW which can be quickly computed and is much better than currently known linear-time approximations. Though it takes inconsequential cost in terms of time it achieves benefit when it comes to effective clustering.[15]

Day-ahead solar irradiance forecasting was done by DWT-CNN-Model. wherein the model consists of three major parts: Dwt for solar irradiance sequence decomposition, CNN for local feature extractor, and LSTM for irradiance sequence decomposition. The Prediction of this model is compared to another six solar irradiance forecasting model and it turns out that this model has high superiority in solar irradiance forecast.[16]

Daily peak Forecasting can be accurately done by a gated recurrent network combining Dynamic time Wrapping (DTW).here the shaped based DTW distance is used to match similar load curves to capture trends in load changes. The paper proposed that the model achieves satisfactory results with other algorithms using the same dataset.[17]

4. Proposed methodology

- **Workflow**

  As previously defined in our objective, to build a true market indicator that is able to capture dynamic weights from our model, a forward feed recurrent neural network was the initial step to gain insight into the predicted opening prices of the 13 commodities that have been considered in our model.

The conventional back-propagation updates each weight and bias by going back to the neural network and considering the portion of error attributed to the output stage.

However, an approach that helps to extract features from these commodities and group them on the basis of their price trends seems suitable to determine which part of the year their weight in the market must be the highest, for this clustering, the Dynamic Time Warping technique and hierarchical clustering are utilized.

The objective of Dynamic Time warping is to produce a distance metric between two input time series. The similarity or dissimilarity of two-time series is typically calculated by converting the data into vectors and calculating the Euclidean distance between those points.

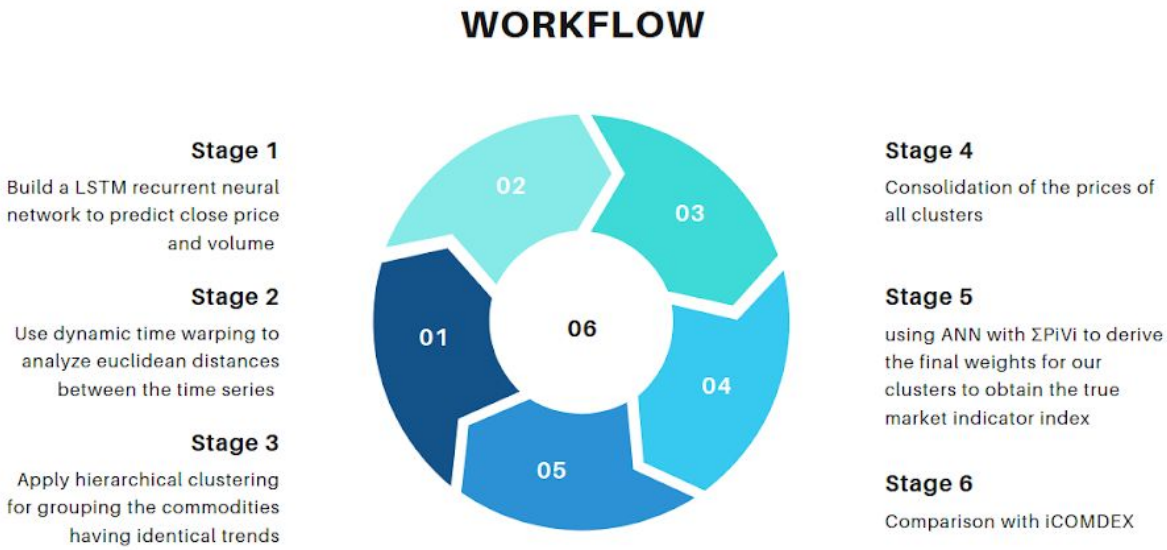


Figure 2: Project Workflow

Then by using hierarchical clustering we obtain roughly 5 clusters each containing about 1-4 commodities. This stands to be an advantage as it is unlike the predefined clustering that MCX works on. The result of hierarchical clustering is a tree-based representation of the objects, which is also known as a dendrogram. It is an approach to partition clustering for identifying groups in the dataset.

After identifying the clusters the next step is to assign weights to each cluster in a way that best represents the market sentiments/trend. For this, the consolidated prices of each of the 5 clusters are fed into a Deep Artificial Neural Network with the label as the summation of all the commodity prices multiplied by their respective volume. label=**ΣPiVi.**

After running the model we will arrive at the weights that have to be assigned to each individual cluster and essentially estimate the index.

- **Technology**

  The data for all the commodities are stored in a pandas data frame using investpy python library. It stores information about the commodities including volume, low, opening, closing price. Then the information is preprocessed to get the desired output like span(High - Low) of the day. We remove all the days in which no trading occurs by removing every record having volume as zero. Then to further simplify the process of creating an RNN we keep all the commodities of the same shape.

  Keeping the end goal in mind of creating a dynamic index-based, by adjusting weights, using past trends we started our work by creating a simple RNN.

  We take in 12 commodities and through a loop and save them in a NumPy array. Then using Keras from TensorFlow we are able to generate a RNN. We divide our dataset into tests and train data with training making 67% of the data.

  We are running a single layer LSTM for this purpose and that's enough as we are able to get a low loss of about 0.00018 only after 10 epochs. LSTM takes a parameter lookback which is set to 20 days as there are mostly 20 trading days in a month and to predict seasonality in the data a month lookback provides better results.

  The loss function used is mean square error and optimizer is adam which is most common and used often. Firstly we were doing it for a single commodity but now it is stored for all 12 of them. We are only taking close prices as input in the LSTM model.

The next step was using all 12 commodities RNN results and applying dynamic time warping to create clusters. We use fastdtw to create a (12,12) correlation matrix. Then we use a dendrogram using scipy to get clusters which will make our index.

The PiVi value of each commodity is calculated and a shift of it is taken keeping the first value as zero. Clusters make up our X input and PiVi makes our labels.

Then using a simple 4 layer ANN with tanh as activation functions we get an index as an output. We take all four clusters and create a dataframe and then convert it to a numpy array to give it as input to our ANN.. Then we get weights assigned to every cluster using get_weights() and that is saved in a list. These weights are then used to create the index.

5. Results and Analysis

The data from investpy after preprocessing it and saving it into a dataframe.

	Date	Open	Close	Volume	span
0	2011-12-19	611.0	588.4	4227	24.9
1	2011-12-20	587.0	580.7	3147	19.0
2	2011-12-21	581.3	591.0	3372	18.9
3	2011-12-22	591.4	588.3	2165	10.8
4	2011-12-23	588.3	590.5	1404	6.4
...	...	...	...	...	...
2184	2020-04-07	2020.0	2031.2	24	58.0
2185	2020-04-08	2031.0	2048.4	5	29.0
2191	2020-04-20	1740.0	1735.5	2	9.0
2192	2020-04-21	1785.0	1785.0	1	0.0
2194	2020-04-23	1838.5	1785.0	2	107.0

2125 rows × 5 columns

Figure 3: investPy Data

To obtain a tradeable index for the commodity derivatives we have firstly built an LSTM model to predict the closed price and volume from the time series data obtained from Investpy. The dataset has been divided into two parts, 67% of train data and 33% of test data.

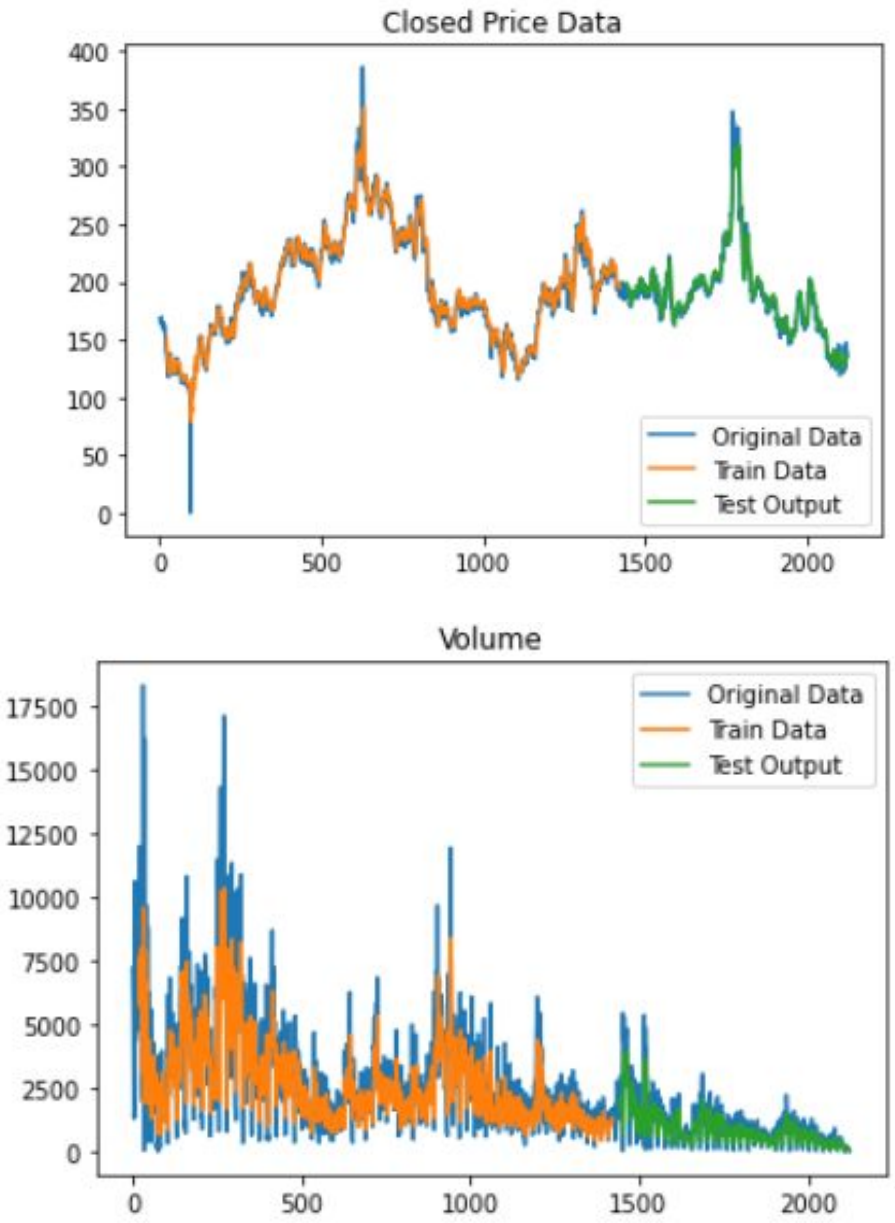


Figure 4: Predicted Values for Volume and Close Price

From the above graphs it can be observed that after training the model with the train dataset, the test output is in alignment with the original data of closed price and volume. The price predicted has a loss of the order of magnitude of -4 which is low.

After obtaining the predicted values of closed price and volume, feature extraction is carried out and the close commodities are clustered together using Dynamic Time

Warping. Our model is attempting to create a group of those commodities that show similarity on the basis of their price trends, it does not depend on the predefined clusters that MCX works on.

After computing, roughly 5 clusters are obtained containing 1-4 commodities each which are different from the standard clusters and the result of this hierarchical clustering is represented in the form of a dendrogram.

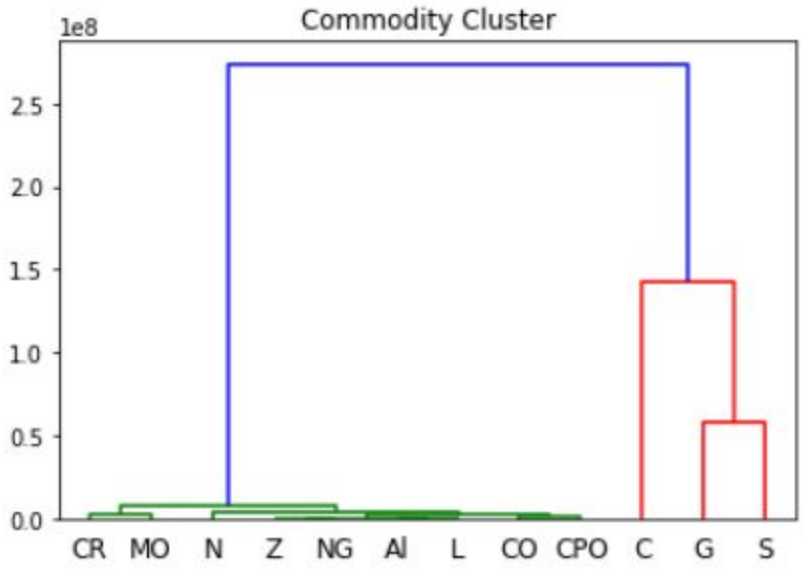


Figure 5: Dendrogram representing the different clusters

From the above dendrogram the following clusters can be obtained:

Cluster 1 - Cardamom & Mentha Oil

Cluster 2 - Cotton Gold & Silver

Cluster 3 - Nickel

Cluster 4 - Lead, Aluminium, Zinc & Natural Gas

Cluster 5 - Copper & Crude Palm Oil

The cluster values are saved in a pandas dataframe.

	cluster1	cluster2	cluster3	cluster4	Nickel
0	1057.714111	26744.085286	161.126820	443.201111	600.819336
1	1066.752136	26794.468099	161.227074	442.751602	605.365906
2	1080.765594	26903.315755	160.581617	442.675003	607.241516
3	1089.342804	26927.597656	159.996239	442.739944	609.591064
4	1095.456696	27012.566406	158.862055	442.933014	610.436890
...	...	...	...	...	...
676	1580.974976	32051.145508	136.347958	519.236908	902.156921
677	1594.812317	32178.699870	136.985138	518.457077	904.432800
678	1606.083191	32180.374349	137.247707	518.737564	921.026001
679	1623.172913	32181.388997	138.943398	519.293015	924.995239
680	1585.461365	32068.380208	139.095772	519.031219	926.435120

681 rows × 5 columns

Figure 6: Consolidated values for the price of each cluster

Then we make labels for our ANN model which is PiVi. Then we calculate the fraction change of the PiVi value from the previous day to feed it as our label.

	0	Fraction
0	1.532878e+09	0.000000
1	2.008269e+09	0.270126
2	1.557996e+09	-0.253873
3	1.362566e+09	-0.134031
4	2.309066e+09	0.527473
...	...	...
676	3.824652e+09	0.341202
677	2.646405e+09	-0.368266
678	3.765319e+09	0.352631
679	4.085766e+09	0.081676
680	4.649962e+09	0.129350

681 rows × 2 columns

Figure 7: Fraction value depicting the change in value in a span of 24 hours

Then both are converted into numpy arrays and feed into our ANN model. Then after running the model we get the weights for all individual clusters.

```
a = model2.get_weights()
avg = np.array([])
for i in range(5):
  sum = 0
  for j in range(16):
    sum = sum + a[0][i][j]
  avg = np.append(avg, sum/16)
print("The final weights are", avg)
```

```
The final weights are [ 0.04059269  0.00269106  0.00720449 -0.0139355   0.03417151]
```

Figure 8: The weights after running ANN

	cluster1	cluster2	cluster3	cluster4	Nickel	Index
0	1057.714111	26744.085286	161.126820	443.201111	600.819336	130.420830
1	1066.752136	26794.468099	161.227074	442.751602	605.365906	131.085640
2	1080.765594	26903.315755	160.581617	442.675003	607.241516	132.007909
3	1089.342804	26927.597656	159.996239	442.739944	609.591064	132.496590
4	1095.456696	27012.566406	158.862055	442.933014	610.436890	132.991467
...	...	...	...	...	...	...
676	1580.974976	32051.145508	136.347958	519.236908	902.156921	175.002042
677	1594.812317	32178.699870	136.985138	518.457077	904.432800	176.000221
678	1606.083191	32180.374349	137.247707	518.737564	921.026001	177.027240
679	1623.172913	32181.388997	138.943398	519.293015	924.995239	177.863799
680	1585.461365	32068.380208	139.095772	519.031219	926.435120	176.082822

681 rows × 6 columns

Figure 9: The index generated with dynamic weights

	agri	precious	metal	energy	index
0	10603.178589	30107.361328	286.926181	541.234060	901.878887
1	10622.288025	30167.640625	287.381808	545.306544	903.746730
2	10654.655457	30307.179688	287.416042	548.548650	907.876667
3	10620.542664	30381.603516	287.696100	551.253876	909.962459
4	10620.596130	30510.492188	288.483000	551.924037	913.651005
...	...	...	...	...	...
676	9008.670532	40045.142578	342.421704	660.558100	1183.305165
677	9032.662842	40224.248047	343.423544	661.357417	1188.493271
678	9121.479187	40146.113281	346.587625	664.219442	1186.639261
679	9167.393921	40107.240234	346.735507	675.640610	1185.933390
680	9071.112854	39995.880859	347.701636	674.816132	1182.496975

681 rows × 5 columns

Figure 10: Index we obtain using iCOMDEX weights

6. Conclusion

So, according to our proposed methodology the cluster that is supposed to be assigned is different from that of the predefined cluster assigned by iCOMDEX index. In iCOMDEX the clusters are based on similar types of commodity. Bullions, Agriculture, metals and energy are iCOMDEX clusters whereas we have Cotton and Crude Palm Oil in one of the clusters based on their similar trends rather than their types. Also, according to our assumptions the market behaves in a different way and to accommodate it one must take the weights of each cluster dynamically. By the use of the deep ANN model we were able to assign weights to the cluster. This has to be repeated every 25 days to incorporate the new data and dynamically update the weights. By our proposed methodology we got some satisfactory results and also a few drawbacks. These drawbacks may be, because our dataset is small and few a checkpoints that are not being handled correctly, we have highlighted some of our shortcomings as follows:

1. We are only taking into account 12 commodities out of all the 40 traded commodities in MCX. The 12 commodities in consideration are the most traded commodities on the MCX exchange
2. The dataset that we had at our disposal had only 2125 data points, which is conventionally very small for a good prediction
3. The data is extracted via a third party library called Investpy, the reliability is not 100% as it is not from the direct source.
4. We could have missed a few parameters important to predict the market, we being novice in this field and due to time constraint might have missed on some important market features.
5. Our fundamental data based approach for index building could be flawed.

For Future work, we will like to propose that when dealing with the commodity market the selection of dataset should be given importance and proper labeling of dataset should be incorporated in order to achieve better accuracy through our proposed methodology.

7. References

[1] Authors: A. Chakraborti, M. Patriarca, M. S. Santhanam, April 13, 2007, Financial time-series analysis: A brief overview

[2] Authors: Matthew Dixon, Diego klabjan, jin Hoon Bang

[3] Authors: Fearghal Kearney, Han Lin Shang, Lisa Sheenan, February 17, 2017, Implied Volatility Surface Predictability: The Case of Commodity Markets

[4] Authors:Arthur le calvez, Dave cliff, November 18-21, 2018, Deep Learning can Replicate Adaptive traders in a limit-order-book financial market

[5] Authors: Lorenz Schneider, Bertrand Tavin, November 27, 2018, Seasonal Stochastic Volatility and the Samuelson Effect in Agricultural Futures Markets

[6] Authors: Omer Berat Sezera, M. Ugur Gudeleka, Ahmet Murat Özbayoğlu, November 29, 2019, Financial Time Series Forecasting with Deep Learning : A Systematic Literature Review: 2005-2019

[7] Authors: Indranil sengupta, william Nganje, Erik Hanson, December 2, 2019, Refinements of Barndorff-Nielsen & shephard model:Analysis of crude oil price

[8] Authors: Alexandre Miot, Gilles Drigout,December 10, 2019, An empirical study of neural networks for trend detection in time series

[9] Authors:Yoshiaki Nakada, November 28, 2018, The effects of energy and commodity prices on commodity output in a three-factor, two-good general equilibrium trade model

[10] Authors: Gautam Prasad, Upendra Reddy Vuyyuru, Mithun Das Gupta, June 14, 2019, Agriculture Commodity Arrival Prediction using Remote Sensing Data: Insights and Beyond

[11] Authors: Sangeeta Rani, Geeta sikka,August 15, 2012, Recent Techniques of clustering of Time Series Data : A survey.

[12] Authors: Tsung-jung Hsieh, Hsiao-Fen Hsiao, Wei-Chang Yeh, February 6,2010, Forecasting stock markets using wavelet transforms and recurrent neural networks:An integrated system based on artificial bee colony algorithm.

[13] Authors: Lipo Wang, Shekar Gupta, 2013, Neural Networks and wavelet DeNoising for stock trading and prediction.

[14] Authors: Chotirat Ann Ratanamahatana, Eamonn Keogh, 2012, Making Time-Series Classification More Accurate Using Learned Constraints.

[15] Authors: Qiang zhu, Gustavo Batista, Thanawin Rakthanmanon, Eamonn Keogh, 2012, A novel approximation to Dynamic time warping allows anytime clustering of massive time series datasets.

[16] Authors: Fei Wang, Yili Yu, Zhanyao Zhang, Jie Li, Zhao Zhen, Kangping Li, August 1, 2018, Wavelet Decomposition and Convolutional LSTM Networks Based Improved Deep Learning Model for Solar Irradiance Forecasting.

[17] Authors: Zeyuan Yu, Zhewen Niu, Wenhu Tang, Qinghua Wu, January 5, 2019, Deep Learning for Daily Peak Load Forecasting - A Novel Gated Recurrent Neural Network Combining Dynamic Time Warping.

8. Source Code

```
#imports
!pip install investpy
import tensorflow as tf
from tensorflow import keras
import matplotlib.pyplot as plt
import statsmodels.tsa as ts
from numpy import array
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error
import investpy
import numpy as np
import pandas as pd
import math

#Fetching data from Investpy
```

```
 df = investpy.commodities.get_commodity_historical_data(commodity="MCX
Mentha Oil",
                                                        from_date='19/12/2011',
                                                           to_date='26/4/2020',
                                                           country='india',
                                                           order='ascending',
interval='Daily')
 df = df.drop(columns=['Currency'])
 df['span'] = df['High'] - df['Low']
 df = df[df['Volume'] > 0]
 df = df.reset_index()
 df = df.drop(columns=['High' , 'Low'])
 Df

#Preprocessing of data
dfs = investpy.commodities.get_commodity_historical_data(commodity="MCX
Cardamom",
                                                        from_date='19/12/2011',
                                                           to_date='26/4/2020',
                                                           country='india',
                                                           order='ascending',
interval='Daily')
dfs = dfs.reset_index()
dfs = dfs.drop(columns=['Currency'])
dfs['span'] = dfs['High'] - dfs['Low']
dfs = dfs.drop(columns=['High' , 'Low'])
dfs = dfs[dfs['Volume'] > 0]
dfs = dfs[dfs.Date.isin(df.Date)]
Dfs

#Saving it in numpy array
n = dfs.values
n = n.reshape((n.shape[0], n.shape[1], 1))
N.shape

#Implementing same steps for all 12 commodities
```

```
mcx = ['MCX Aluminum Mini','MCX Copper','MCX Gold Mini','MCX Lead
Mini','MCX Nickel','MCX Silver Micro','MCX Zinc Mini',"MCX Mentha
Oil",'MCX Cotton',
       'MCX Crude Palm Oil','MCX Natural Gas']
for i in mcx:
  df2 = investpy.commodities.get_commodity_historical_data(commodity=i,
                                                    from_date='19/12/2011',
                                                       to_date='26/4/2020',
                                                       country='india',
                                                       order='ascending',
interval='Daily')
  df2 = df2.reset_index()
  df2 = df2.drop(columns=['Currency'])
  df2['span'] = df2['High'] - df2['Low']
  df2 = df2[df2['Volume'] > 0]
  df2 = df2.drop(columns=['High' , 'Low'])
  df2 = df2[df2.Date.isin(df.Date)]
  if(df2.shape[0] > 2125):
    s =  df2.shape[0] - 2125
    df2 = df2[s:]
  card = df2.values
  card = card.reshape((card.shape[0], card.shape[1], 1))
  print(card.shape)
  n = np.concatenate((n,card), axis = 2)
  plt.plot(df2['Close'])
plt.show()
print(n.shape)

#RNN for the close price of All 12 commodities
Final_testPredicts = np.array([])
from keras.models import Sequential
from keras.layers.recurrent import LSTM
from keras.layers import Activation, Dense
def create_dataset(dataset, look_back=1):
  dataX, dataY = [], []
  for i in range(len(dataset)-look_back-1):
    a = dataset[i:(i+look_back), 0]
```

```
    dataX.append(a)
    dataY.append(dataset[i + look_back, 0])
  return np.array(dataX), np.reshape((np.array(dataY)),(-1,1))

for i in range (0,12):
  dataset = np.array([])
  for j in range(0,2125):
    dataset = np.append (dataset,n[j][2][i])
  dataset = dataset.astype('float32')
  scaler = MinMaxScaler(feature_range=(0, 1))
  dataset = scaler.fit_transform(np.reshape(dataset,(-1, 1)))
  train_size = int(len(dataset) * 0.67)
  test_size = len(dataset) - train_size
  train, test = dataset[0:train_size,:],
dataset[train_size:len(dataset),:]
  print(len(train), len(test))
  look_back = 20
  trainX, trainY = create_dataset(train, look_back)
  testX, testY = create_dataset(test, look_back)
  trainX = np.reshape(trainX, (trainX.shape[0], 1, trainX.shape[1]))
  testX = np.reshape(testX, (testX.shape[0], 1, testX.shape[1]))
  model = Sequential()
  model.add(LSTM(4, input_shape=(1,20)))
  model.add(Dense(1))
  model.compile(loss='mean_squared_error',
optimizer='adam',metrics=["accuracy"])
  model.fit(trainX, trainY, epochs=10, batch_size=1, verbose=2)
  trainPredict = model.predict(trainX)
  testPredict = model.predict(testX)
  trainPredict = scaler.inverse_transform(trainPredict)
  trainY = scaler.inverse_transform(trainY)
  testPredict = scaler.inverse_transform(testPredict)
  testY = scaler.inverse_transform(testY)
  trainPredictPlot = np.empty_like(dataset)
  trainPredictPlot[:, :] = np.nan
  trainPredictPlot[look_back:len(trainPredict)+look_back, :] =
trainPredict
```

```
  testPredictPlot = np.empty_like(dataset)
  testPredictPlot[:, :] = np.nan
  testPredictPlot[len(trainPredict)+(look_back*2)+1:len(dataset)-1, :]
= testPredict
  Final_testPredicts = np.append(Final_testPredicts, testPredict)
  plt.plot(scaler.inverse_transform(dataset))
  plt.plot(trainPredictPlot)
  plt.plot(testPredictPlot)
  testPredictPlot
  plt.show()

#Reshaping the output
Final_testPredicts = np.reshape(Final_testPredicts, [12,681])
Final_testPredicts[0]

#Dynamic Time Warping
from fastdtw import fastdtw
from scipy.spatial.distance import euclidean
final_matrix = np.empty((12,12))
for i in range(12):
  matrix = np.array([])
  for j in range(12):
    distance, path = fastdtw(Final_testPredicts[i],
Final_testPredicts[j] , dist=euclidean)
    matrix = np.append(matrix,distance)
    final_matrix[i][j] = distance
Final_matrix

#Dendrogram to get the clusters
import numpy as np
from scipy.cluster.hierarchy import dendrogram, linkage
from scipy.spatial.distance import squareform
from scipy.spatial.distance import pdist
import matplotlib.pyplot as plt
mat = final_matrix
dists = squareform(pdist(mat))
linkage_matrix = linkage(dists, "complete")
```

```
dendrogram(linkage_matrix,
labels=["CR","Al","CO","G","L","N","S","Z","MO","C","CPO","NG"])
plt.title("Complete Link")
plt.show()

#Calculating original Price for our test data
Final_Price = np.empty((12,681))
for i in range(12):
  for j in range(681):
    Final_Price[i][j] = n[j+1443][2][i]

Final_Price_Predicts = Final_Price
#Calculating original Volume for our test data
Final_Volume = np.empty((12,681))
for i in range(12):
  for j in range(681):
    Final_Volume[i][j] = n[j+1443][3][i]

Final_Volume_Predicts = Final_Volume

#Cluster creation
Cardamom = Final_testPredicts[0]
Al = Final_testPredicts[1]
Copper = Final_testPredicts[2]
Gold = Final_testPredicts[3]
Lead = Final_testPredicts[4]
Nickel = Final_testPredicts[5]
Silver = Final_testPredicts[6]
Zinc = Final_testPredicts[7]
MenthaOil = Final_testPredicts[8]
Cotton = Final_testPredicts[9]
Cpo = Final_testPredicts[10]
Ng = Final_testPredicts[11]
cluster1 = np.array([])
cluster2 = np.array([])
cluster3 = np.array([])
cluster4 = np.array([])
```

```
for i in range(681):
  cl1 = (Cardamom[i] + MenthaOil[i])/2
  cl2 = (Cotton[i] + Gold[i] + Silver[i])/3
  cl3 = (Lead[i] + Al[i] + Zinc[i] + Ng[i])/4
  cl4 = (Copper[i] + Cpo[i])/2
  cluster1 = np.append(cluster1,cl1)
  cluster2 = np.append(cluster2,cl2)
  cluster3 = np.append(cluster3,cl3)
  cluster4 = np.append(cluster4,cl4)

#Making pandas dataframe and then to numpy array to get input
cluster = np.array([])
final_file={"cluster1":cluster1,"cluster2":cluster2,"cluster3":cluster3
,"cluster4":cluster4,"Nickel":Nickel}
f=pd.DataFrame(final_file)
f_np = f.to_numpy()
F_np.shape

#Calculating PiVi
Cardamom_V = Final_Volume_Predicts[0]
Al_V = Final_Volume_Predicts[1]
Copper_V = Final_Volume_Predicts[2]
Gold_V = Final_Volume_Predicts[3]
Lead_V = Final_Volume_Predicts[4]
Nickel_V = Final_Volume_Predicts[5]
Silver_V = Final_Volume_Predicts[6]
Zinc_V = Final_Volume_Predicts[7]
MenthaOil_V = Final_Volume_Predicts[8]
Cotton_V = Final_Volume_Predicts[9]
Cpo_V = Final_Volume_Predicts[10]
Ng_V = Final_Volume_Predicts[11]
P_V= np.array([])
for i in range(681):
  pv = Cardamom[i]*Cardamom_V[i] + Al[i]*Al_V[i] +
Copper[i]*Copper_V[i] + Gold[i] *Gold_V[i] + Lead[i]*Lead_V[i] +
Nickel[i]*Nickel_V[i]+ Silver[i]*Silver_V[i] + Zinc[i]*Zinc_V[i] +
```

```
MenthaOil[i]*MenthaOil_V[i] + Cotton[i]*Cotton_V[i] + Cpo[i]*Cpo_V[i] +
Ng[i]*Ng_V[i]
  P_V = np.append(P_V,pv)
P_V.shape

#Converting to Dataframe and getting the shift
from math import log
P_Vnew =pd.DataFrame(P_V)
P_Vnew.set_index(Final_Dates)
P_Vnew["Fraction"] = np.log(P_Vnew[0])- np.log(P_Vnew[0].shift(1))
P_Vnew
P_Vnew["Fraction"][0] = 0

#Converting to numpy array
PV = P_Vnew["Fraction"].to_numpy()

#Dividing dataset into train and test
trainX, testX = f_np[:457, :], f_np[457:, :]
trainy, testy = PV[:457], PV[457:]

#Scaling our X axis
scaler_x = MinMaxScaler(feature_range=(0, 0.95))
trainX = scaler_x.fit_transform(trainX)
trainX.shape
testX = scaler_x.transform(testX)
testX.shape

#ANN model
from keras.layers import Dropout
model2 = Sequential()
model2.add(Dense(16, input_shape = (5,), activation='tanh'))
model2.add(Dropout(0.4))
model2.add(Dense(8, activation='tanh'))
model2.add(Dropout(0.2))
model2.add(Dense(4, activation='tanh'))
model2.add(Dense(1, activation='tanh'))
#Comping and fitting the model
```

```
from keras.callbacks import EarlyStopping
es = EarlyStopping(monitor='val_loss', mode='min', verbose=1,
patience=15)
model2.compile(loss='mean_squared_error', optimizer='adam',
metrics=['mse'])
model2.fit(trainX, trainy, validation_data=(testX, testy), epochs=200,
batch_size=1, callbacks=[es])

#Getting weights
a = model2.get_weights()
avg = np.array([])
for i in range(5):
  sum = 0
  for j in range(16):
    sum = sum + a[0][i][j]
  avg = np.append(avg, sum/16)
print("The final weights are", avg)

#Scatter Plot
trainPredict = model2.predict(trainX)
testPredict = model2.predict(testX)
plt.scatter(trainPredict, trainy)
plt.show()

#Making of Index
f["Index"] = f['cluster1']*avg[0] + f['cluster2']*avg[1] +
f['cluster3']*avg[2] + f['cluster4']*avg[3] + f['Nickel']*avg[4]

#Comdex
agriculture = np.array([])
precious = np.array([])
metal = np.array([])
energy = np.array([])
for i in range(681):
  agri = (Cardamom[i] + Cotton[i])/2
  pr_metal = (Gold[i] + Silver[i])/2
  mt = (Lead[i] + Al[i] + Zinc[i] + Copper[i] + Nickel[i])/5
```

```
  eng = (MenthaOil[i] + Ng[i] + Cpo[i])/3
  agriculture = np.append(agriculture,agri)
  precious = np.append(precious,pr_metal)
  metal = np.append(metal,mt)
  energy = np.append(energy,eng)
comdex={"agri":agriculture,"precious":precious,"metal":metal,"energy":e
nergy}
G=pd.DataFrame(comdex)
G

#Making Comdex Index using the weights defined by MCX
G['index'] = G['agri']*0.0028 + G['precious']*0.0284 +
G['metal']*0.0156 + G['energy']*0.0234
G

#Plotting Both Indexes
Indexs = {'comdex': G['index'], 'dynamic' : f['Index']}
Indexs =pd.DataFrame(Indexs)
Indexs
plt.figure()
Indexs.plot()
plt.show()
```