MALAYSIAN HOUSE PRICE INDEX IN FUTURE

2023-2030

WAN MAISARAH BINTI WAN ALIAS SD20042

NG JIE HAO SD20036

Electives Project Report

Centre for Mathematical Sciences

UNIVERSITI MALAYSIA PAHANG

CONTENTS

CHAPTER 1

INTRODUCTION

1.1 Project Background

The Malaysian House Price Index (MHPI) serves as a crucial indicator of the real estate market's performance and trends of the housing market that measures the average changes in house prices in Malaysia over time. The index provides valuable insights into the state of the real estate sector, which is an important component of Malaysia's economy. With the introduction of the Twelfth Malaysian Plan (RMK12) and its transformative approach, understanding the implications of RMK12 on the housing market becomes essential for organisations operating in this sector. The MHPI is calculated based on the prices of residential properties sold and purchased in Malaysia. It takes into account various factors such as location, property type, size, and other relevant variables. By analysing these factors, the index reflects the overall movement of house prices and serves as a benchmark for monitoring price fluctuations in the housing market.

The HPI and overall house prices are closely related but might not always move in perfect sync. The HPI reflects the average price changes in the housing market, considering factors like inflation, demand and supply dynamics, and market conditions. It provides a standardised measure to assess price movements over time. On the other hand, overall house prices are influenced by a wide range of factors, including local market conditions, property characteristics, buyer and seller behaviour, and economic factors specific to a given region.

RMK12, on the other hand, alludes to the 12th Malaysian Plan. The Malaysian Plans, a set of recurring, five-year development plans, govern how Malaysia functions. These plans lay forth the nation's plans for economic and social development as well as the government's priorities and policies for the given time frame. The 2021–2025 time frame is covered by the 12th Malaysian Plan (RMK12). Over a five-year period, it lays out the objectives for Malaysia's economic expansion, social advancement, and environmental sustainability. The plan takes into account various aspects such as infrastructure development, human capital development, economic diversification, and sustainable development, among others. The government uses RMK12 as a road map when creating policies, allocating funds, and putting programmes into action to meet its stated goals. It covers the possibilities and problems facing Malaysia throughout the given time period and offers a thorough framework to direct the country's development efforts. The plan intends to promote equitable and sustainable growth, boost people's well-being, and enhance Malaysia's competitiveness globally. As for considering the impact of RMK12, this project aims to analyse historical MHPI data from 2010 to 2022 and forecast future trends to support organisational sustainability.

1.2 Problem Statement

The MHPI is a statistical index that analyses changes in Malaysian residential real estate prices over time. It acts as a crucial marker for the stability and general health of the property market. The MHPI, however, confronts a number of difficulties and problems that must be resolved, including a lack of accuracy and representativeness. Due to limitations in data collecting and sampling techniques, the MHPI might not correctly reflect the true changes in home prices. The index might not accurately reflect the complete housing market because it does not include all geographic areas or all types of properties. As a result, inaccurate assumptions about the state of the housing market may be drawn. According to Director of CPI Land Sdn Bhd Chung Shan Tat (2023), apart from the increase in construction raw material prices, the soaring costs of labour and financing are also contributing factors to the rise in housing prices. Other than that, another challenge facing people nowadays is the rising inflation and higher borrowing costs have tightened potential buyers' budgets, simultaneously forcing sellers to raise their prices to cover the higher cost of property investments. This is because inflation issues keep on arising in difficulties people are facing today.

1.3 Project Questions

i. What is the historical data and future trend of Malaysian House Price Index?

ii. How to analyse the distribution of house price index with different states and identify regions with higher house price index?

iii. What can we do with the deep learning model in predicting house price index indices in Malaysia?

1.4 Project Objectives

i. To forecast the future trends of the Malaysian House Price Index based on historical data by using times series analysis.
ii. To analyse the distribution of house price index with different states and identify regions with higher house price index by using geographical information systems.
iii. To evaluate the performance of deep learning model LSTM networks, in predicting house price indices in Malaysia.

1.5 Project Scopes

The scopes of the project are:

i. The dataset is based on a reported house price index in Malaysia only.
ii. Forecasts are made until December 2030.
iii. Forecasts are made regarding time series analysis and deep learning.

CHAPTER 2

DATA COLLECTION AND PREPARATION

2.1 Introduction

Data preparation and collecting are essential elements in the construction process To maintain the accuracy and representativeness of the Malaysian House Price Index (MHPI). Find trustworthy and complete sources of property market information. It is crucial to check that the data sources encompass a wide geographic area and a variety of property kinds. Choose a representative sample for the index that accurately reflects the housing market. Property types include apartments, landed homes, commercial properties, and properties from various price ranges should all be represented in the sample.

2.2 Data Collection

In this subtopic, the required data are in secondary data which include the Malaysian House Price Index, Malaysia (2009-2022), Malaysian House Price Index by state, Malaysia, (2009-2022) and the GADM data shape file that provides information about Malaysia Map. We will collect data related to Malaysian House Price Index from the website Pusat Maklumat Harta Tanah Negara Malaysia https://napic2.jpph.gov.my/ms for this project. This project will use secondary data obtained from various reputable sources and agencies.

We decided to do the forecasting of the Malaysian House Price Index and here the steps to collect the data:

a. Visit the website https://napic2.jpph.gov.my/ms and navigate to the relevant section or dataset that provides information on Malaysian House Price Index.

b. Find and identify the dataset that includes historical Malaysia House Price Index data for the desired period (2009-2022) and is available for download.

c. Download the dataset in a format that is compatible with the data analysis tools such as in (.csv) format.

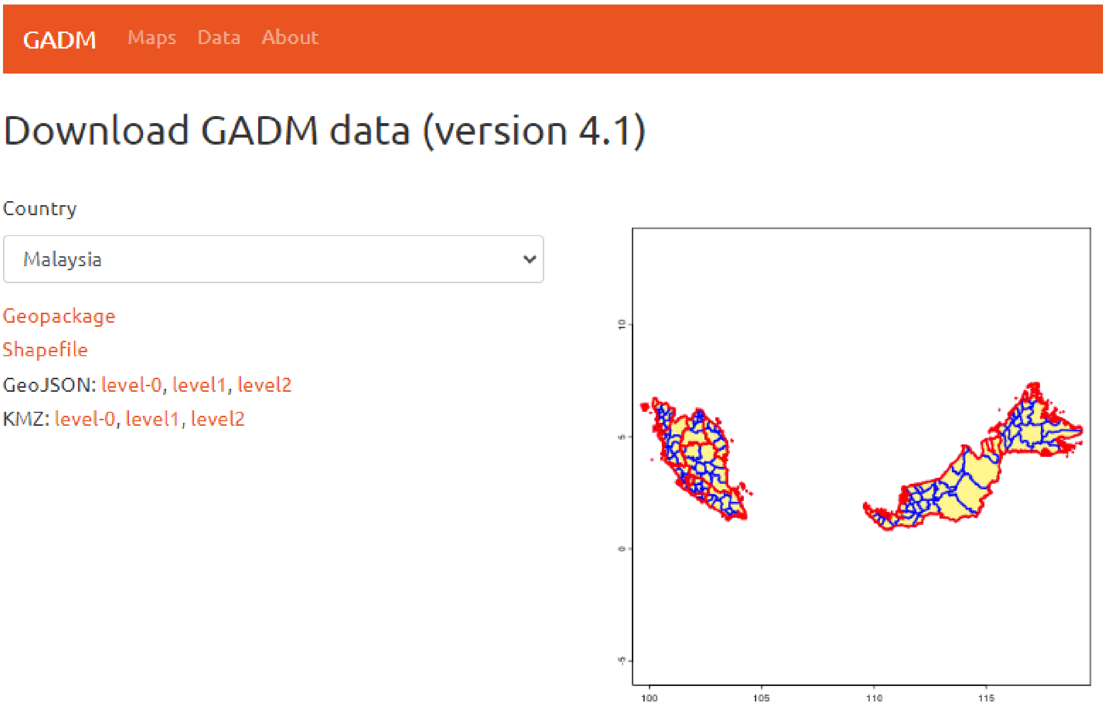
d. Ensure that the dataset includes variables such as house price index date, location, severity, and any other relevant information that aligns with the research objectives.

Below are the files that we collected and will be used in this project:

i. Malaysia House Price Index, Malaysia (2009-2022)
https://napic2.jpph.gov.my/ms/archives/indeks-harga-rumah-malaysia

ii. Malaysia House Price Index by state, Malaysia, (2009-2022)
https://napic2.jpph.gov.my/ms/archives/indeks-harga-rumah-malaysia

iii. Malaysia Map shape file from the Global Aviation Data Management (GADM).
https://gadm.org/download_country.html



2.3 Data Preparation

Jadual I : Indeks Harga Semua Rumah Mengikut Negeri

Table I : All House Price Index by State

State	2019 Q2	2019 Q3	2019 Q4	2020 Q1	2020 Q2	2020 Q3	2020 Q4	2021 Q1	2021 Q2	2021 Q3	2021 Q4	2022 Q1	2022 Q2	2022 Q3	2022 Q4	2023 Q1[P]
All House Price Index (2010 = 100)																
Malaysia	**197.6**	**199.5**	**198.8**	**199.7**	**200.4**	**199.9**	**201.2**	**201.1**	**202.5**	**202.0**	**205.0**	**205.9**	**207.8**	**212.4**	**213.0**	**210.1**
Kuala Lumpur	197.0	198.5	196.3	198.3	197.8	195.6	194.3	191.1	190.6	189.0	192.9	191.9	191.5	193.0	194.2	192.6
Selangor	202.0	204.3	202.9	201.3	202.0	201.2	202.9	203.8	205.9	205.5	210.4	210.2	212.1	218.0	217.7	213.6
Johor	226.8	228.6	230.2	233.2	232.6	235.2	238.2	240.3	242.0	241.3	243.9	242.7	246.2	252.3	254.0	251.8
Pulau Pinang	195.6	197.4	194.0	195.8	197.2	195.1	197.3	194.2	196.4	190.2	193.1	193.7	198.6	203.1	205.9	204.9
Negeri Sembilan	190.2	192.2	196.2	197.8	200.1	202.4	198.4	202.0	203.3	207.1	210.1	214.3	218.5	223.6	224.5	222.5
Perak	191.5	194.1	197.0	201.5	205.8	205.3	207.5	209.6	213.7	217.6	214.2	215.8	221.8	225.9	226.9	222.7
Melaka	171.7	175.9	178.5	180.2	179.7	183.7	184.8	187.5	190.4	193.0	194.7	197.7	199.5	205.1	207.0	207.7
Kedah	176.3	179.3	184.1	183.7	184.7	187.3	189.8	191.5	196.0	199.1	203.0	207.0	208.5	215.5	215.0	211.3
Pahang	179.4	177.7	175.6	177.0	175.6	179.4	178.5	177.6	181.9	182.5	185.6	189.6	192.2	198.3	198.0	194.8
Terengganu	172.9	173.9	172.9	170.4	173.1	176.5	177.0	175.3	177.2	180.1	172.7	180.1	183.0	185.1	182.9	183.3
Kelantan	166.7	171.6	177.9	179.1	182.9	188.3	193.8	194.5	200.0	204.1	205.4	210.5	215.2	222.7	222.6	220.1

COVER | Page 2 | CONTENT | CHANGE | TABLE I | TABLE II | TABLE III | TABLE IV | TABLE V | TABLE VI | TABLE VII | TABLE VIII | TABLE IX

Based on the table above, this is the MHPI data that we collected from the websites we mentioned above in Section 2.2 (Data Collection). After getting this data, we extracted some of the information from the table and made new (.csv) files to achieve the objectives of this project.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1	Negeri	2009	2010	2011	2012	2013	2014	2015	2016	2017	2018	2019	2020	2021	2022	Growth	Average HPI
2	Kuala Lum	91.9	100	112.5	128	146.1	159.4	171.1	184.4	197.7	198.5	197.9	196.5	190.9	192.7	100.8	161.9714
3	Selangor	92.9	100	112.3	129.1	139	151	163.2	177.7	191.5	198.2	202.1	201.9	206.4	214.5	121.6	162.8429
4	Johor	101.8	100	109	122.4	148.8	169.9	179.2	193.4	206.2	218.5	227.5	234.8	241.9	248.8	147	178.7286
5	Pulau Pin	96.9	100	109.5	123.5	143	159.8	171.4	180.7	189.8	191.7	195.7	196.4	193.5	200.3	103.4	160.8714
6	Negeri Se	96.9	100	111.1	120	127.2	138.5	149.6	162.4	174.3	186.9	192	199.7	205.6	220.2	123.3	156.0286
7	Perak	96.4	100	111.8	123.2	132.3	144.4	155.6	164.1	171.2	181.3	192.4	205	213.8	222.6	126.2	158.15
8	Melaka	95.7	100	104.4	109.5	116.9	122.1	132	142.2	152.4	166.7	173.8	182.1	191.4	202.3	106.6	142.25
9	Kedah	94.4	100	107.5	113.3	124.4	131.1	139.1	148.2	155.6	167.3	178	186.4	197.4	211.5	117.1	146.7286
10	Pahang	102.3	100	115.6	131	139.9	151	161	167.5	173	173.1	176.5	177.6	181.9	194.5	92.2	153.2071
11	Terenggar	95	100	115.8	128.8	143	150.3	159	167.2	170.9	171.2	173.1	174.3	176.3	182.8	87.8	150.55
12	Kelantan	91.6	100	106.2	110	124.2	131.1	135.7	136.6	143.4	154.8	169.9	186	201	217.8	126.2	143.45
13	Perlis	96	100	114.5	116.4	126.1	141.1	149	150.7	155.7	167.5	184.6	192.2	195.1	205.5	109.5	149.6
14	Sabah	90	100	112.1	126	139.4	149.6	158.5	158.3	166.3	176.9	180.2	178.5	183.3	191.9	101.9	150.7857
15	Sarawak	94.9	100	106.8	119.6	132.9	140.8	150.4	157.9	165.7	172.8	178.5	188.4	190.2	197.4	102.5	149.7357

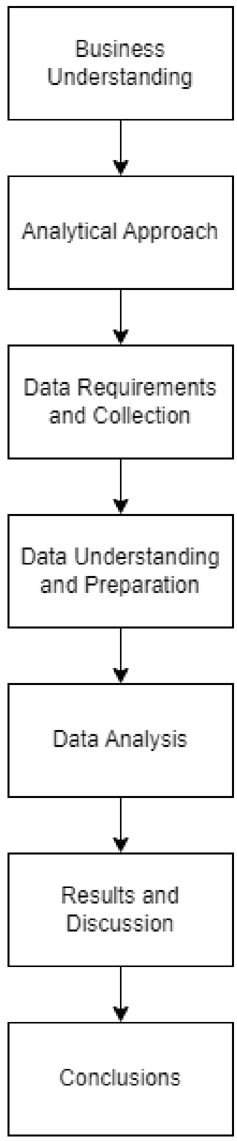
This is one of the new tables that we made after acquiring some of the information. This table is about the MHPI by state from 2009 until year 2022. There is also a growth column that shows changes of house price from 2009 until 2022 and average HPI.

CHAPTER 3

METHODOLOGY

3.1 Introduction

This chapter will explain the method and approach used in this project. A methodology of a project section gives a thorough explanation of the strategy, tactics, and processes employed to carry out the project and accomplish its goals. It describes the overall structure of the framework and procedures used to gather data, examine information, and come to conclusions. It will also explain the work flow throughout completing this project in a flowchart.



3.2 Business Understanding

According to the problem statement of this project, the problem has been defined in terms of house price index. Thus, each step is generalised in overcoming house price index problem-related problems. There are three things that have to be in business understanding which are questions, goals and objectives. As for the questions of this project, what is the historical data and future trend of Malaysian House Price Index? The goal of this project is to forecast the future Malaysian House Price Index (MHPI). First objective of this project is to forecast the future trends and potential growth of the Malaysian House Price Index based on historical data by using times series analysis. Secondly is to analyse the distribution of house price index with different states and identify regions with higher house price index by using geographical information systems. Finally is to evaluate the performance of deep learning model LSTM networks, in predicting house price indices in Malaysia.

3.3 Analytical Approach

This project will apply three (3) analytical approaches according to the elective courses, which are the Time Series Analysis (BSD4463), Geographical Information System (BSD4663) and Deep Learning (BSD4543). Each course will be applied to each objective as follows:

i. Time Series Analysis (BSD4463) - To forecast the future trends of the Malaysia House Price Index based on historical data by using times series analysis.

ii. Geographical Information System (BSD4663) - To analyse the distribution of house price index different states and identify regions with higher house price index by using geographical information systems.

iii. Deep Learning (BSD4543) - To compare and evaluate the performance of different deep learning architectures LSTM networks in predicting house price indices in Malaysia.

3.4 Data Requirements & Collection

Malaysian House Price Index (MHPI) data were collected from the website Pusat Maklumat Harta Tanah Negara Malaysia https://napic2.jpph.gov.my/ms and Database of Global Administrative Areas https://gadm.org/maps/MYS.html. The details will be explained in Section 2.2 Data Requirements & Collection.

3.5 Data Preparation

The data collected will be prepared and cleansed according to the desires of this project. As the data is secondary, data must be transformed and formatted according to each objective requirement. Data preparation involves several steps, including data cleaning, transformation, and formatting. The data preparation has been explained in detail in Section 2.3 Data Preparation.

3.6 Modelling

Modelling used in this project are long short-term memory (LSTM) and Double Exponential Smoothing model.

3.6.1 Long short-term memory (LSTM)

LSTM is a sequential data model designed to address the issue of long-term memory decay in Simple Recurrent Neural Networks (RNNs). LSTM includes three essential components: forget gate, input gate, and output gate. LSTM networks are suitable for classification, processing, and making predictions based on time series data. Unlike other recurrent neural networks, the LSTM's cell state is designed to prevent memory loss over time (Park et al., 2020). LSTM memory cell can be seen in the figure below:

$$f(t) = \sigma(Wf * [h(t-1), x(t)] + bf)$$

$$i(t) = \sigma(Wi * [h(t-1), x(t)] + bi)$$

$$o(t) = \sigma(Wo * [h(t-1), x(t)] + bo)$$

Where:

$i(t)$ = input gate

$f(t)$ = forget gate

$o(t)$ = output gate

o = sigmoid function

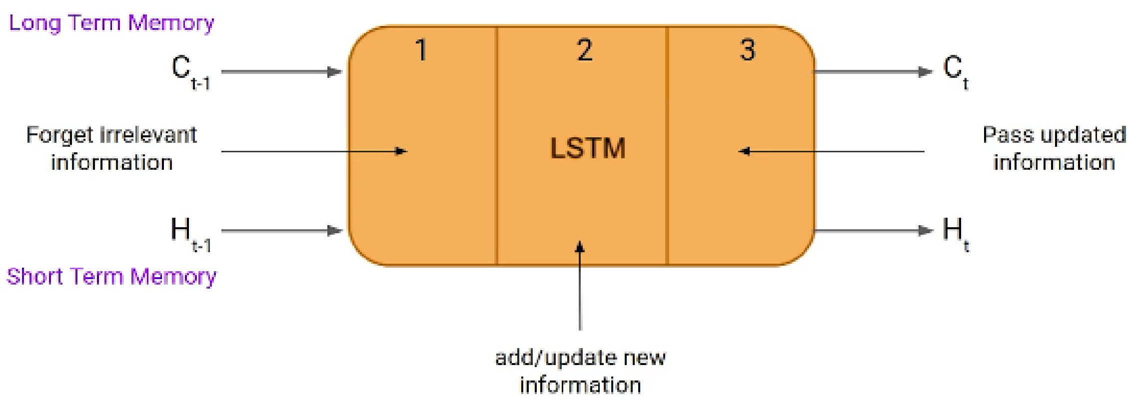
w = weight for the neurons

$x(t)$ = input at times

b = biases

$h(t-1)$ = Autoregression effect at time

Component of LSTM memory cell.



3.6.2 Double Exponential Smoothing

The exponentially smoothed series or current level estimate:

$$L_t = \alpha Y_t + (1-\alpha)(L_{t-1} + T_{t-1})$$

The trend estimate:

$$T_t = \beta(L_t - L_{t-1}) + (1-\beta)T_{t-1}$$

Forecast p periods into the future:

$$\hat{Y}_{t+p} = L_t + pT_t$$

where

L_t= estimate of current level
α=smoothing constant for the level
Y_t= actual value of series in period t
T_t= estimate of current trend
β=smoothing constant for the trend
p= periods to be forecast in the future
$\hat{Y}_{t+p}$ = forecast for p periods into the future

3.7 Data Analysis, Results & Discussion

The final data from the method and strategy employed in this project, according to each necessary aim, will be applied in the detailed step of the analysis. In accordance with the project objectives and the data gathered, it will also describe how each finding should be interpreted. The details will be explained in Chapter 4, Data Analysis, Results and Discussion.

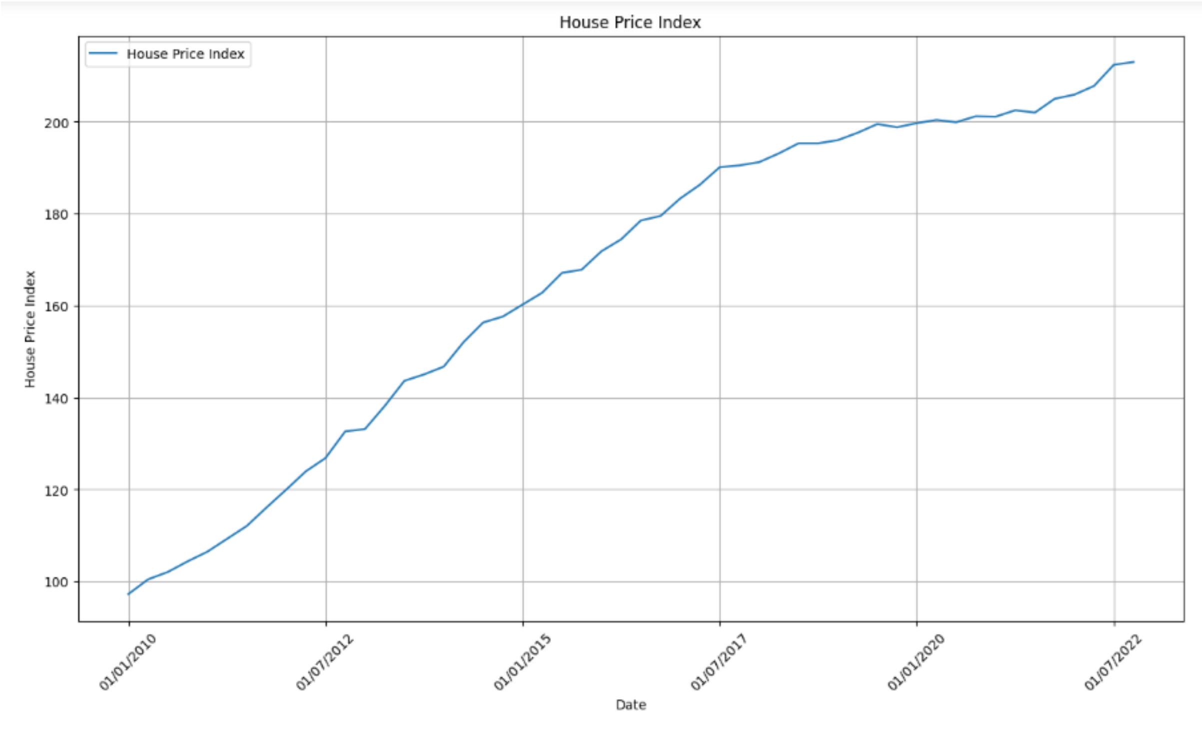
CHAPTER 4

DATA ANALYSIS, RESULTS AND DISCUSSION

4.1 Introduction

This chapter focuses on the data collection process, followed by an exploratory data analysis and data preparation. The findings derived from the applied methods and approaches in this project will be presented and discussed. Additionally, the interpretation of each finding will be provided, aligning them with the project objectives and the data collected.

4.2 Time Series Analysis



The plot of the historical data of the house price index suggests a relatively strong upward trend in the data from the year 2010 to the year 2022 for both data. The data also does not vary about a fixed level, exhibits an overall upward trend, and the variances increase as the series increases, suggesting that the data are stationary in variance but non-stationary in mean.

Based on plot above, in order to forecast the value of the future number of house price index, three methods are suggested to be tested and evaluated, which are:

i: Simple Exponential Smoothing
ii: Double Exponential Smoothing (Holt's Method)
iii: Triple Exponential Smoothing (Holt-Winter Method)

4.2.1 Simple Exponential Smoothing

Simple Exponential Smoothing is a basic time series forecasting method that assigns exponentially decreasing weights to historical observations in order to forecast future values. It is a commonly used technique for forecasting data without trends or seasonal patterns. The formula for Simple Exponential Smoothing can be expressed as follows:

$$\hat{Y}_{t+1} = \alpha Y_t + (1-\alpha)\hat{Y}_t)$$

where

$\hat{Y}_{t+1}$ = forecast value for the next period
α= smoothing constant
Y_t= actual value of series in period t
$\hat{Y}_t$= forecast for period t

Out[6]:

SimpleExpSmoothing Model Results

Dep. Variable:	House Price Index	**No. Observations:**	42
Model:	SimpleExpSmoothing	**SSE**	861.708
Optimized:	True	**AIC**	130.892
Trend:	None	**BIC**	134.368
Seasonal:	None	**AICC**	131.973
Seasonal Periods:	None	**Date:**	Sun, 25 Jun 2023
Box-Cox:	False	**Time:**	15:55:41
Box-Cox Coeff.:	None		

	coeff	code	optimized
smoothing_level	0.6000000	alpha	False

This is Simple Exponential Smoothing model results. As the results above show, there are no trends and are not seasonal.

4.2.2 Double Exponential Smoothing (Holt's Method)

Double Exponential Smoothing, also known as Holt's method, is an extension of Simple Exponential Smoothing that can handle data with trends. It incorporates an additional component called the trend component to capture and forecast the trend in the data. The formula for Double Exponential Smoothing can be expressed as follows:

The exponentially smoothed series or current level estimate:

$$L_t = \alpha Y_t + (1 - \alpha)(L_{t-1} + T_{t-1})$$

The trend estimate:

$$T_t = \beta(L_t - L_{t-1}) + (1 - \beta)T_{t-1}$$

Forecast p periods into the future:

$$\hat{Y}_{t+p} = L_t + pT_t$$

where

L_t= estimate of current level
α=smoothing constant for the level
Y_t= actual value of series in period t
T_t= estimate of current trend
β=smoothing constant for the trend
p= periods to be forecast in the future
$\hat{Y}_{t+p}$ = forecast for p periods into the future

Out[15]:

Holt Model Results

Dep. Variable:	House Price Index	**No. Observations:**	42
Model:	Holt	**SSE**	103.456
Optimized:	False	**AIC**	45.862
Trend:	Additive	**BIC**	52.813
Seasonal:	None	**AICC**	48.262
Seasonal Periods:	None	**Date:**	Sun, 25 Jun 2023
Box-Cox:	False	**Time:**	15:55:41
Box-Cox Coeff.:	None		

	coeff	**code**	**optimized**
smoothing_level	0.6000000	alpha	False
smoothing_trend	0.5000000	beta	False
initial_level	98.987870	l.0	False
initial_trend	0.000000	b.0	False

This is Double Exponential Smoothing (Holt's Method) model results. As the results above, the trends are additive and non seasonal.

4.2.3 Triple Exponential Smoothing (Holt-Winter Method)

Triple Exponential Smoothing, also known as Holt-Winters Method, is an extension of Double Exponential Smoothing that can handle data with both trend and seasonality. It incorporates an additional component called the seasonal component to capture and forecast seasonal patterns in the data. The formula for Triple Exponential Smoothing can be expressed as follows:

The exponentially smoothed series or current level estimate:

$$L_t = \alpha \frac{Y_t}{S_{t-s}} + (1-\alpha)(L_{t-1} + T_{t-1})$$

The trend estimate:

$$T_t = \beta(L_t - L_{t-1}) + (1-\beta)T_{t-1}$$

The seasonality estimate:

$$S_t = \gamma \frac{Y_t}{L_t} + (1-\gamma)S_{t-s}$$

Forecast p periods into the future:

$$\hat{Y}_{t+p} = (L_t + pT_t)S_{t-s+p}$$

where

L_t= estimate of current level
α=smoothing constant for the level
Y_t= actual value of series in period t
T_t= estimate of current trend
β=smoothing constant for the trend
S_t= seasonal estimate
γ= smoothing constant for the seasonality
p= periods to be forecast in the future
s= length of the seasonality
$\hat{Y}_{t+p}$ = forecast for p periods into the future

Out[24]:

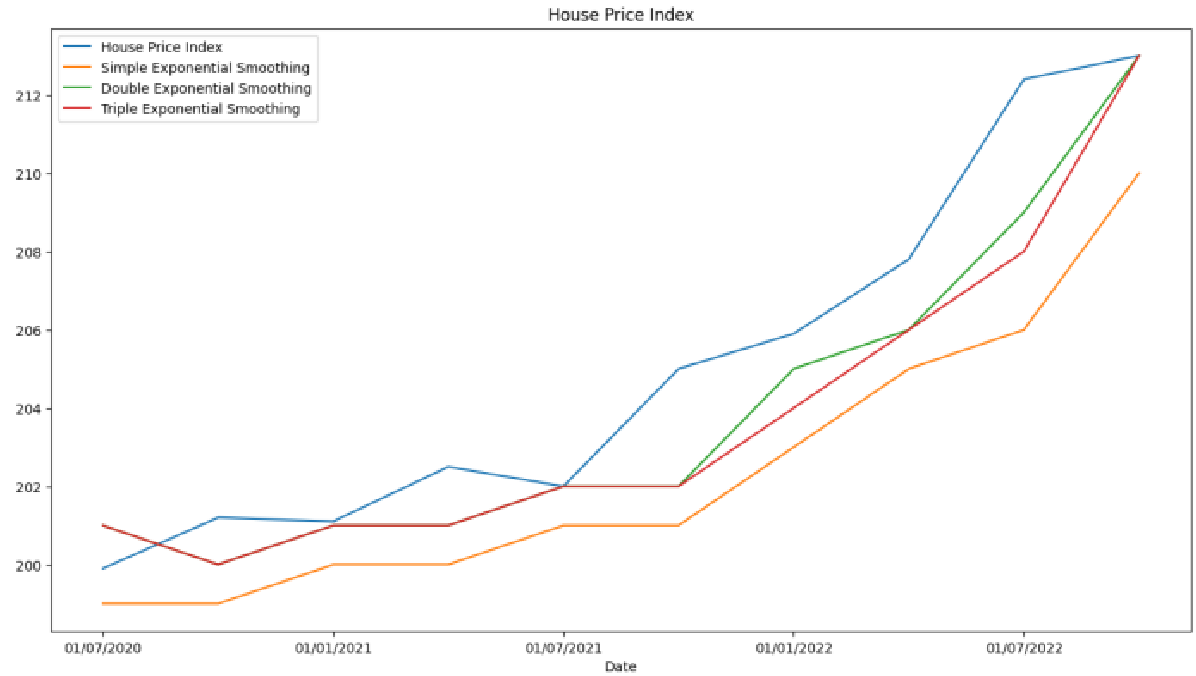
ExponentialSmoothing Model Results

Dep. Variable:	House Price Index	**No. Observations:**	42
Model:	ExponentialSmoothing	**SSE**	78.607
Optimized:	True	**AIC**	34.325
Trend:	Additive	**BIC**	41.276
Seasonal:	None	**AICC**	36.725
Seasonal Periods:	None	**Date:**	Sun, 25 Jun 2023
Box-Cox:	False	**Time:**	15:55:42
Box-Cox Coeff.:	None		

	coeff	code	optimized
smoothing_level	0.5243658	alpha	True
smoothing_trend	0.5243658	beta	True
initial_level	95.279551	l.0	True
initial_trend	2.1521446	b.0	True

This is Triple Exponential Smoothing (Holt-Winter Method) model results. As the results above, the trends are additive and non seasonal.

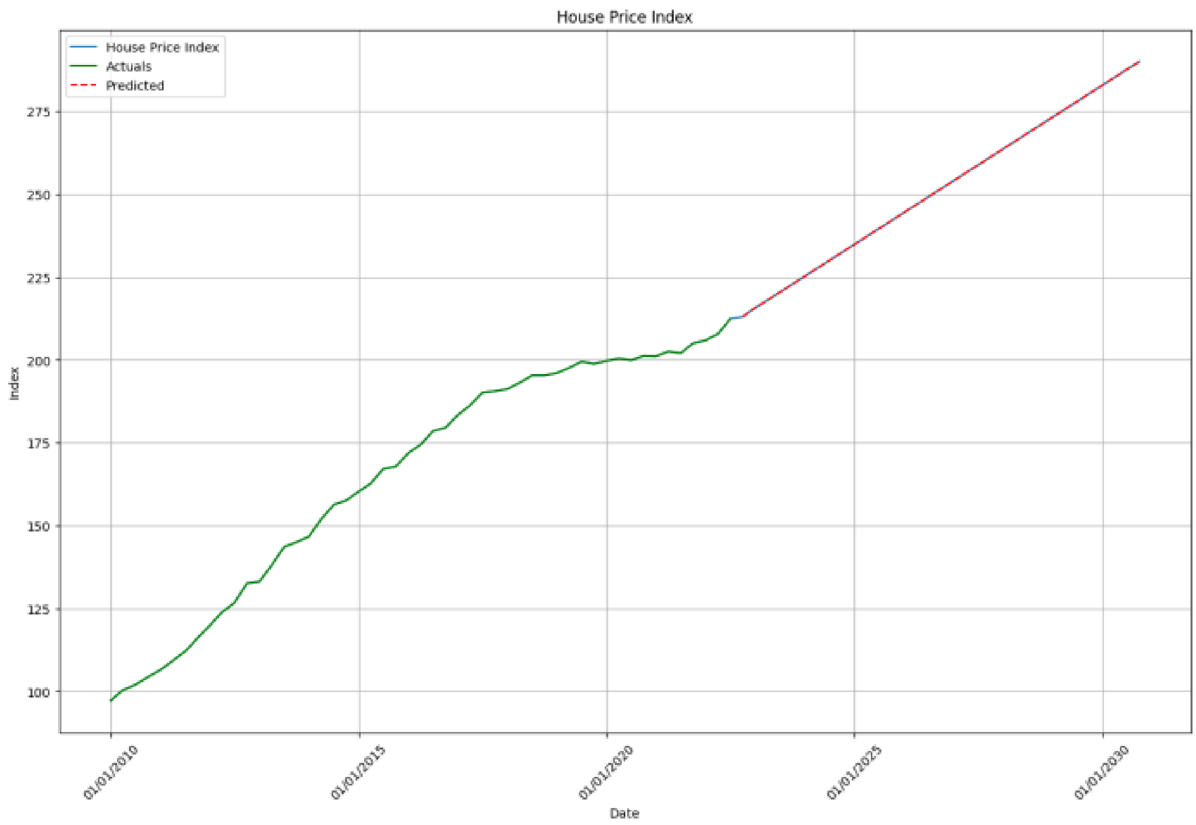
After doing all of the Exponential Smoothing model one by one, we plotting a graph to foresee the overall results as figure below:



As plot above, the orange colour is Simple Exponential Smoothing, green is the Double Exponential Smoothing and red colour is Triple Exponential Smoothing. All these Exponential Smoothing Models are increasing as the year increases.

Model	MAE	MSE	RMSE
SES	2.68	9.63	3.10
DES	1.30	2.95	1.71
TES	1.50	4.01	2.00

Overall, the Double Exponential Smoothing model performed the best among the three models, as it achieved the lowest error metrics (MAE, MSE, and RMSE). It had the smallest average prediction error and exhibited the best overall performance in terms of prediction accuracy. Therefore, the Double Exponential Smoothing model can be considered the most suitable model for forecasting the house price index. The model forecasting for data will be using Double Exponential Smoothing (Holt's Method).

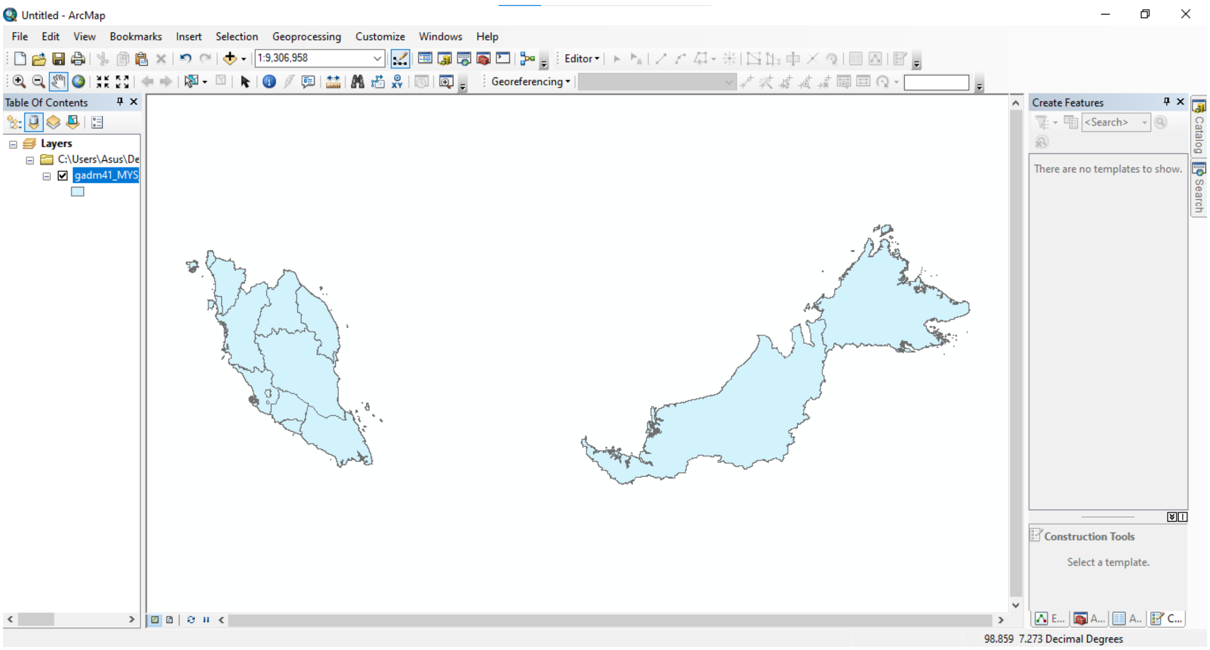


Based on the figure above, it is a prediction regarding the House Price Index in the future. The green line indicates the actual House Price Index from the year 2010 until year 2022. Then, from the year 2023 till 2030 is the predicted value of the House Price Index in Malaysia. As we can see from the pattern, it seems the MHPI continues to increase as the year increases.

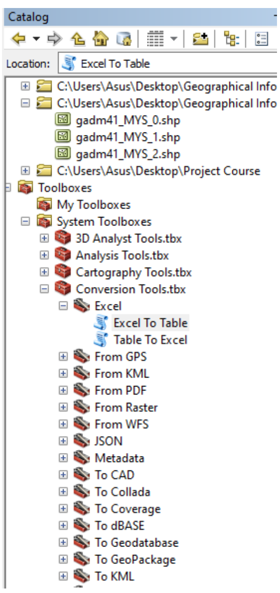
4.3 Geographical Information System

4.3.1 The Distribution of Average House Price in Malaysia

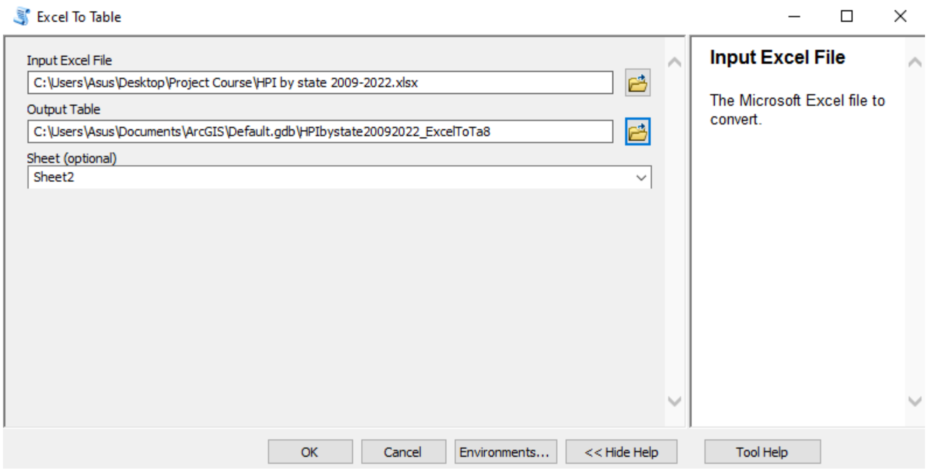
First, import the GADM data shape file into ArcMap as shown below.



Next, conversion tools are used to import the excel file. Excel to table is chosen to import the excel into the table in the ArcMap.



Then, select the correct excel file HPI by state into ArcMap.



The table of the Malaysia map file contains FID, Shape, ID_0, ISO, NAME_0, ID_1, NAME_1, TYPE_1, ENGTYPE_1, NL_NAME_1 and VARNAME_1.

gadm41_MYS_1

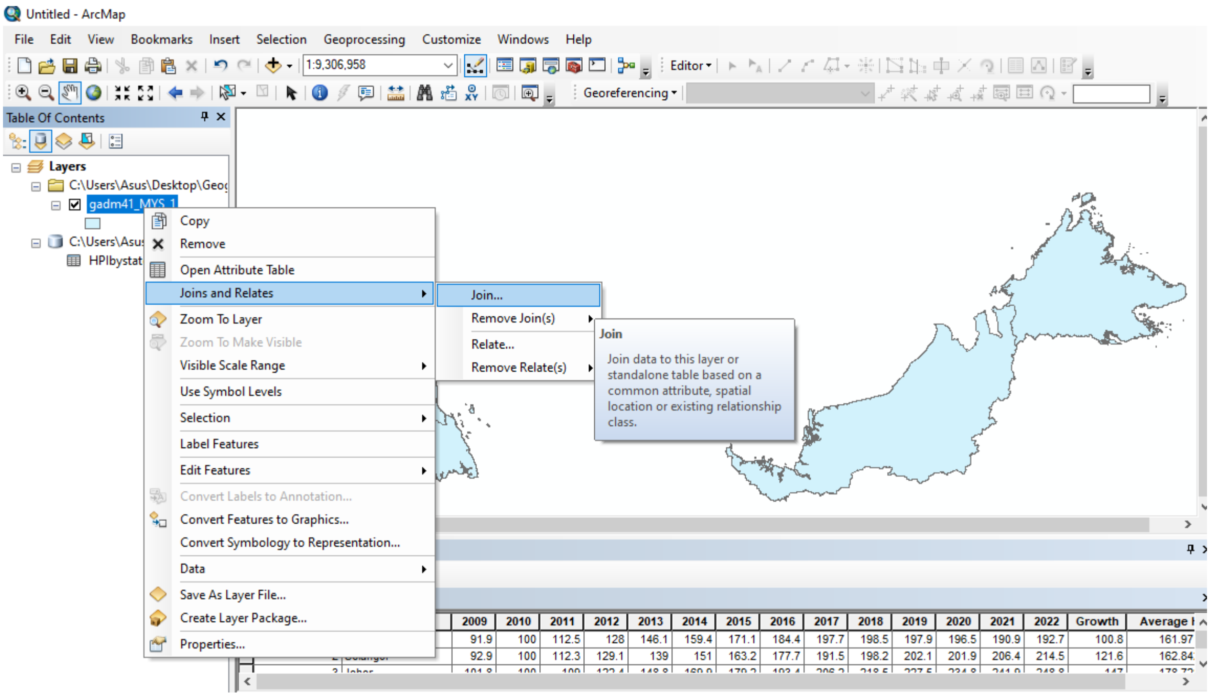
<table>
<tr><th>FID</th><th>Shape *</th><th>GID_1</th><th>GID_0</th><th>COUNTRY</th><th>NAME_1</th><th>VARNAME_1</th><th>NL_NAME_1</th><th>TYPE_1</th><th>ENGTYPE_1</th></tr>
<tr><td>0</td><td>Polygon</td><td>MYS.1_1</td><td>MYS</td><td>Malaysia</td><td>Johor</td><td>Johor Darul Takzim|Johore</td><td>NA</td><td>Negeri</td><td>State</td></tr>
<tr><td>1</td><td>Polygon</td><td>MYS.2_1</td><td>MYS</td><td>Malaysia</td><td>Kedah</td><td>Kedah Darul Aman</td><td>NA</td><td>Negeri</td><td>State</td></tr>
<tr><td>2</td><td>Polygon</td><td>MYS.3_1</td><td>MYS</td><td>Malaysia</td><td>Kelantan</td><td><Null></td><td>NA</td><td>Negeri</td><td>State</td></tr>
<tr><td>3</td><td>Polygon</td><td>MYS.4_1</td><td>MYS</td><td>Malaysia</td><td>Kuala Lumpur</td><td>Federal Territory of Kuala Lumpu</td><td>NA</td><td>Wilayah Persekutuan</td><td>Federal Territory</td></tr>
<tr><td>4</td><td>Polygon</td><td>MYS.5_1</td><td>MYS</td><td>Malaysia</td><td>Labuan</td><td>Federal Territory of Labuan</td><td>NA</td><td>Wilayah Persekutuan</td><td>Federal Territory</td></tr>
<tr><td>5</td><td>Polygon</td><td>MYS.6_1</td><td>MYS</td><td>Malaysia</td><td>Melaka</td><td>Malacca|Malaka</td><td>NA</td><td>Negeri</td><td>State</td></tr>
<tr><td>6</td><td>Polygon</td><td>MYS.7_1</td><td>MYS</td><td>Malaysia</td><td>Negeri Sembilan</td><td>Negeri Sembilan Darul Khusus|Neg</td><td>NA</td><td>Negeri</td><td>State</td></tr>
<tr><td>7</td><td>Polygon</td><td>MYS.8_1</td><td>MYS</td><td>Malaysia</td><td>Pahang</td><td>Pahang Darul Makmur</td><td>NA</td><td>Negeri</td><td>State</td></tr>
<tr><td>8</td><td>Polygon</td><td>MYS.9_1</td><td>MYS</td><td>Malaysia</td><td>Perak</td><td>Perak Darul Ridzuan</td><td>NA</td><td>Negeri</td><td>State</td></tr>
<tr><td>9</td><td>Polygon</td><td>MYS.10_1</td><td>MYS</td><td>Malaysia</td><td>Perlis</td><td>Perlis Indra Kayangan</td><td>NA</td><td>Negeri</td><td>State</td></tr>
<tr><td>10</td><td>Polygon</td><td>MYS.11_1</td><td>MYS</td><td>Malaysia</td><td>Pulau Pinang</td><td>Penang and Province Wellesley|Pe</td><td>NA</td><td>Negeri</td><td>State</td></tr>
<tr><td>11</td><td>Polygon</td><td>MYS.12_1</td><td>MYS</td><td>Malaysia</td><td>Putrajaya</td><td>Federal Territory of Putrajaya</td><td>NA</td><td>Wilayah Persekutuan</td><td>Federal Territory</td></tr>
<tr><td>12</td><td>Polygon</td><td>MYS.13_1</td><td>MYS</td><td>Malaysia</td><td>Sabah</td><td>North Borneo</td><td>NA</td><td>Negeri</td><td>State</td></tr>
<tr><td>13</td><td>Polygon</td><td>MYS.14_1</td><td>MYS</td><td>Malaysia</td><td>Sarawak</td><td>NA</td><td>NA</td><td>Negeri</td><td>State</td></tr>
<tr><td>14</td><td>Polygon</td><td>MYS.15_1</td><td>MYS</td><td>Malaysia</td><td>Selangor</td><td>Selangor Darul Ehsan</td><td>NA</td><td>Negeri</td><td>State</td></tr>
<tr><td>15</td><td>Polygon</td><td>MYS.16_1</td><td>MYS</td><td>Malaysia</td><td>Trengganu</td><td>Terengganu Darul Iman</td><td>NA</td><td>Negeri</td><td>State</td></tr>
</table>

The table of the HPI by state file contains state and each year. The column NAME_1 contains the same data with column Negeri in the HPI by state data, then the table can be joined by these columns.

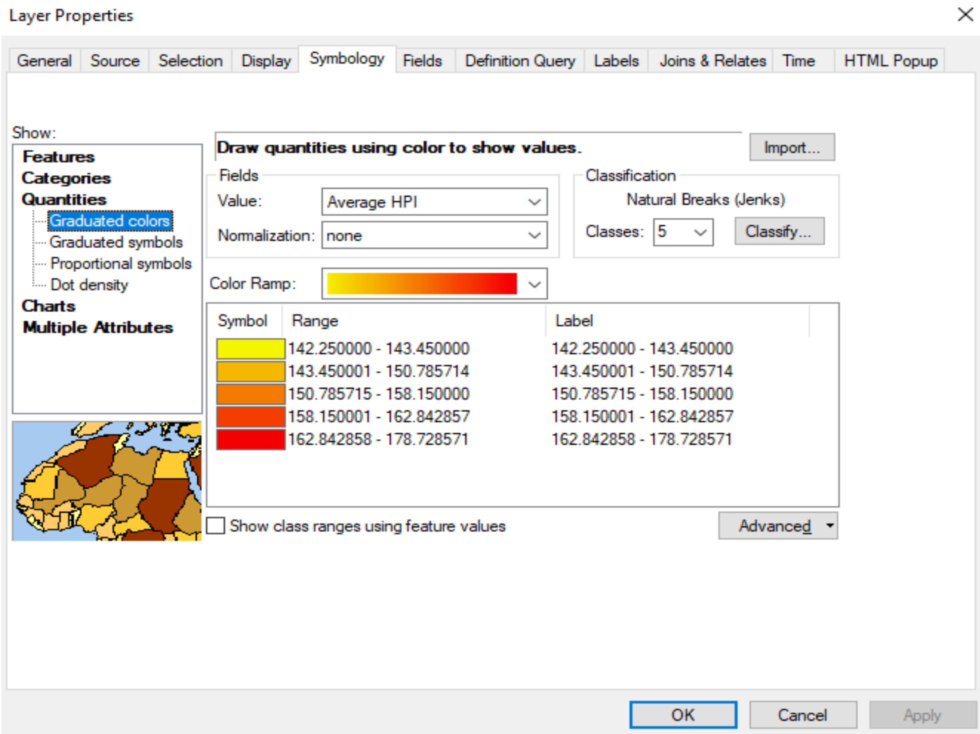
HPIbystate20092022_ExcelToTa9

OBJECTID *	State	2009	2010	2011	2012	2013	2014	2015	2016	2017	2018	2019	2020	2021	2022	Growth	Average HPI
1	Kuala Lumpur	91.9	100	112.5	128	146.1	159.4	171.1	184.4	197.7	198.5	197.9	196.5	190.9	192.7	100.8	161.97142
2	Selangor	92.9	100	112.3	129.1	139	151	163.2	177.7	191.5	198.2	202.1	201.9	206.4	214.5	121.6	162.84285
3	Johor	101.8	100	109	122.4	148.8	169.9	179.2	193.4	206.2	218.5	227.5	234.8	241.9	248.8	147	178.72857
4	Pulau Pinang	96.9	100	109.5	123.5	143	159.8	171.4	180.7	189.8	191.7	195.7	196.4	193.5	200.3	103.4	160.87142
5	Negeri Sembilan	96.9	100	111.1	120	127.2	138.5	149.6	162.4	174.3	186.9	192	199.7	205.6	220.2	123.3	156.02857
6	Perak	96.4	100	111.8	123.2	132.3	144.4	155.6	164.1	171.2	181.3	192.4	205	213.8	222.6	126.2	158.1
7	Melaka	95.7	100	104.4	109.5	116.9	122.1	132	142.2	152.4	166.7	173.8	182.1	191.4	202.3	106.6	142.2
8	Kedah	94.4	100	107.5	113.3	124.4	131.1	139.1	148.2	155.6	167.3	178	186.4	197.4	211.5	117.1	146.72857
9	Pahang	102.3	100	115.6	131	139.9	151	161	167.5	173	173.1	176.5	177.6	181.9	194.5	92.2	153.20714
10	Terengganu	95	100	115.8	128.8	143	150.3	159	167.2	170.9	171.2	173.1	174.3	176.3	182.8	87.8	150.5
11	Kelantan	91.6	100	106.2	110	124.2	131.1	135.7	136.6	143.4	154.8	169.9	186	201	217.8	126.2	143.4
12	Perlis	96	100	114.5	116.4	126.1	141.1	149	150.7	155.7	167.5	184.6	192.2	195.1	205.5	109.5	149.
13	Sabah	90	100	112.1	126	139.4	149.6	158.5	158.3	166.3	176.9	180.2	178.5	183.3	191.9	101.9	150.78571
14	Sarawak	94.9	100	106.8	119.6	132.9	140.8	150.4	157.9	165.7	172.8	178.5	188.4	190.2	197.4	102.5	149.73571

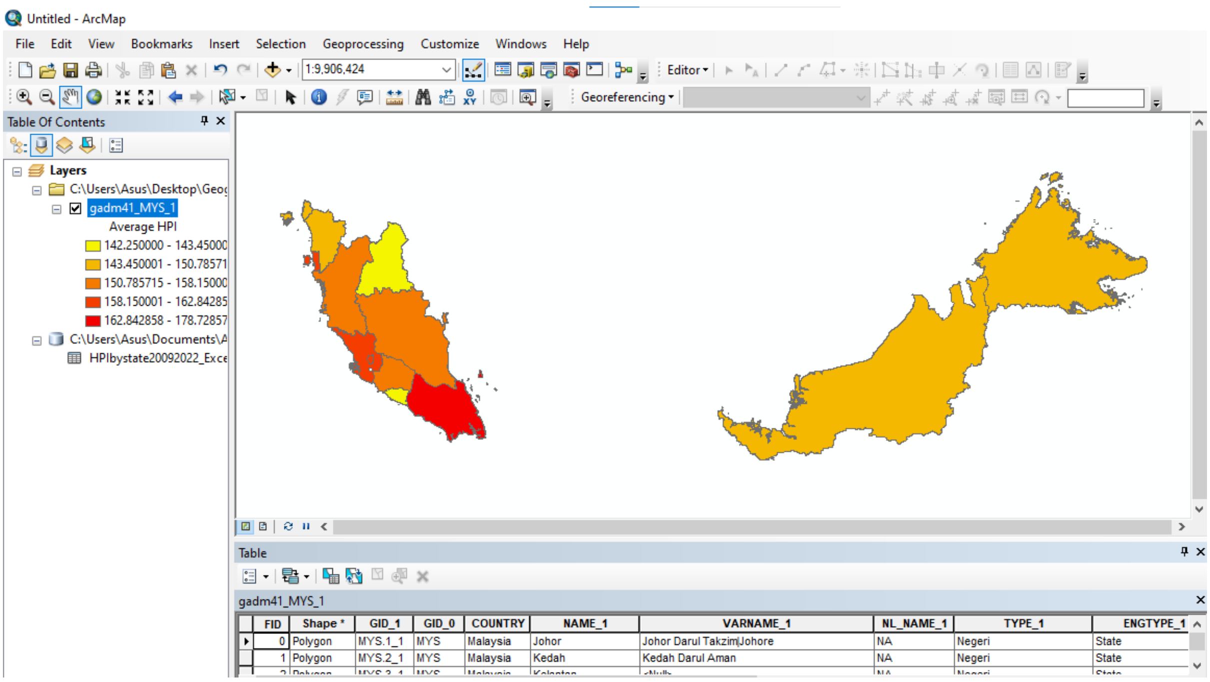
As the data from the shape file table is the same as the state, it can be joined into the Malaysia map.



After merging or joining the relevant data, the next step is to visualise the information on the Malaysia map. The map's colour can be used to represent different ranges based on the average House Price Index (HPI) in Malaysia.

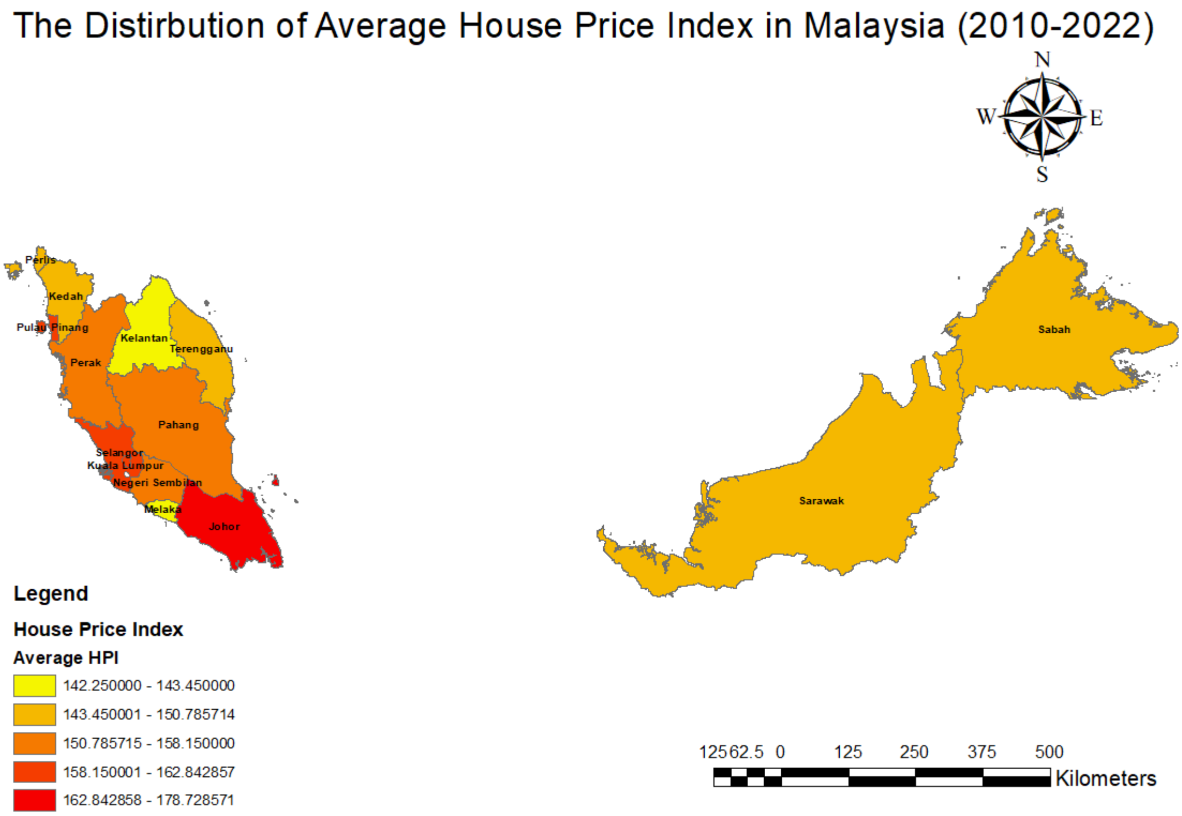


The heat map will be shown, and the range of the colour legends can identify the distribution of the house price index in Malaysia.



Lastly, elements such as a title, north arrow, legend, label features and grid are added to the map. The colour red can be assigned to represent the highest HPI values, indicating regions with the highest average house prices in Malaysia. The colour orange can represent the median range of HPI values, representing areas with moderate average house prices. Finally, the colour yellow can represent the lower range of HPI values, indicating regions with relatively lower average house prices.

4.3.2 Result: Geographical Information System

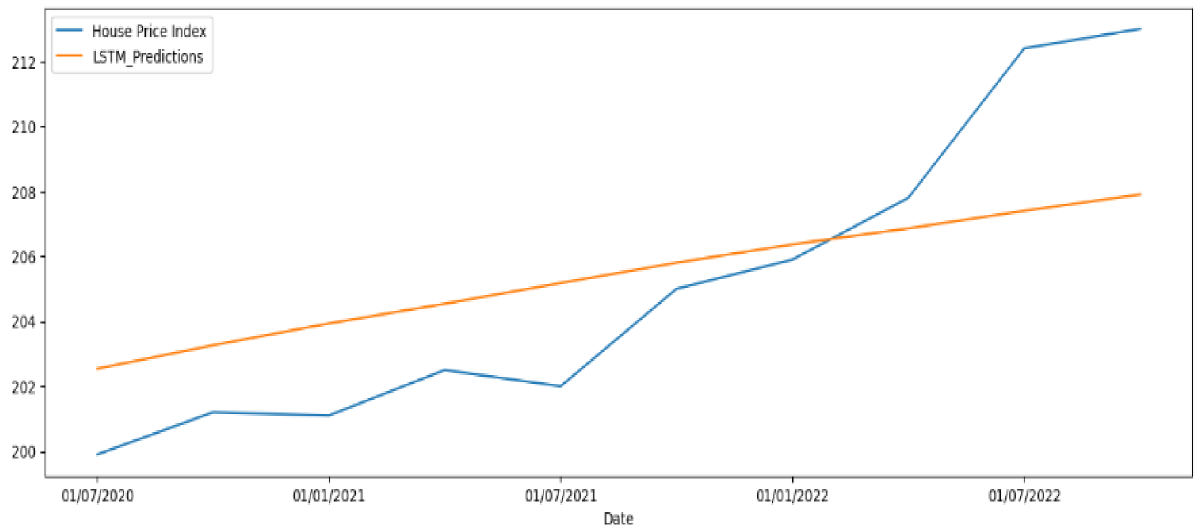


Based on the graph above, we can observe that the average house price index trend across different states in Malaysia from the years 2010 to 2022. The graph reveals that Johor consistently has the highest average house price index during this period, indicating that houses in Johor tend to have higher prices compared to other states. On the other hand, Melaka has the lowest average house price index among the states. Following Johor, Selangor and Kuala Lumpur show the next highest average house price indices. This information provides insights into the relative housing market performance among these states. It suggests that Johor has relatively higher housing prices on average, while Selangor and Kuala Lumpur also exhibit strong housing market performance in terms of price levels.

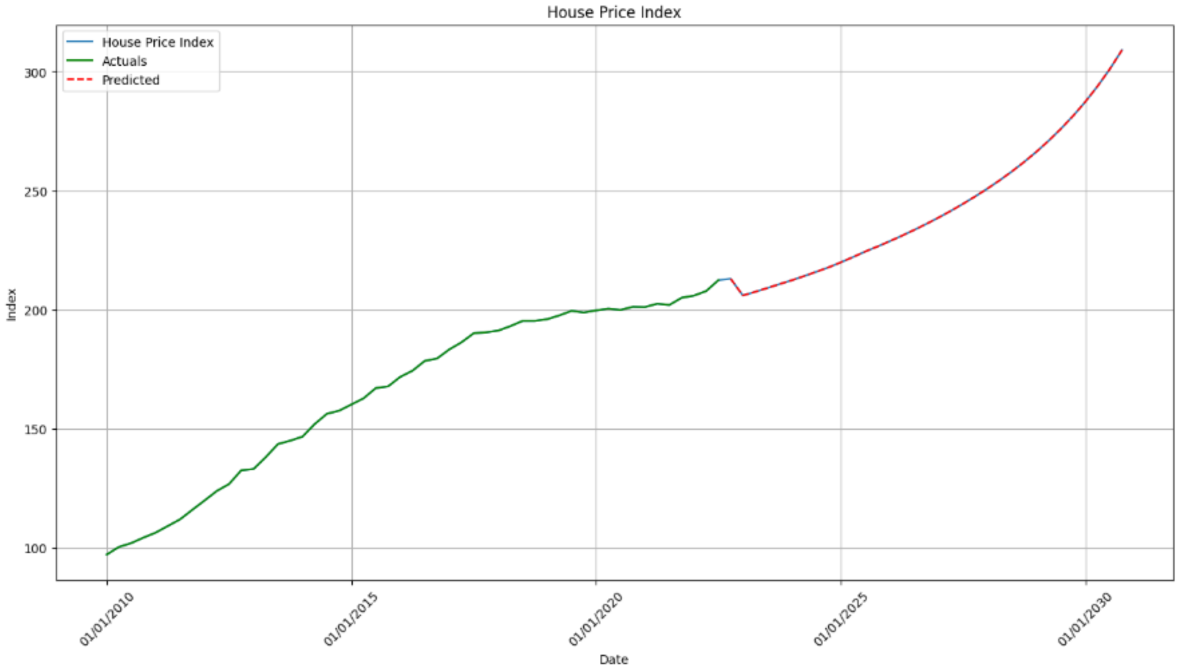
4.4 Deep Learning

The long short-term memory (LSTM) model is trained using historical data from 2010 to 2022. The model learns the patterns and trends in the historical data, enabling it to make predictions for the future in years 2023 to 2030. Comparing the actual values with the model predictions, we can see that the LSTM predictions are relatively close to the actual values. However, there are slight differences between the predicted values and the actual values.

Date	House Price Index	LSTM_Predictions
01/07/2020	199.9	202.540876
01/10/2020	201.2	203.261587
01/01/2021	201.1	203.933212
01/04/2021	202.5	204.540077
01/07/2021	202.0	205.178301
01/10/2021	205.0	205.802525
01/01/2022	205.9	206.367402
01/04/2022	207.8	206.861577
01/07/2022	212.4	207.400692
01/10/2022	213.0	207.902901



The line plot displays the LSTM predictions along with the actual data. Both lines show an uptrend, indicating that the model's predictions generally follow the slightly same upward trend as the actual data.



The line plot represents the forecast data generated by the LSTM model with the actual data. The forecast data represents the predicted values of the house price index from 2023 to 2030. The trend displayed in the plot shows an uptrend from the year 2010 to 2030, indicating that the model predicts an overall increase in the house price index over that time period.

Model	MAE	MSE	RMSE
LSTM	2.51	8.62	2.94

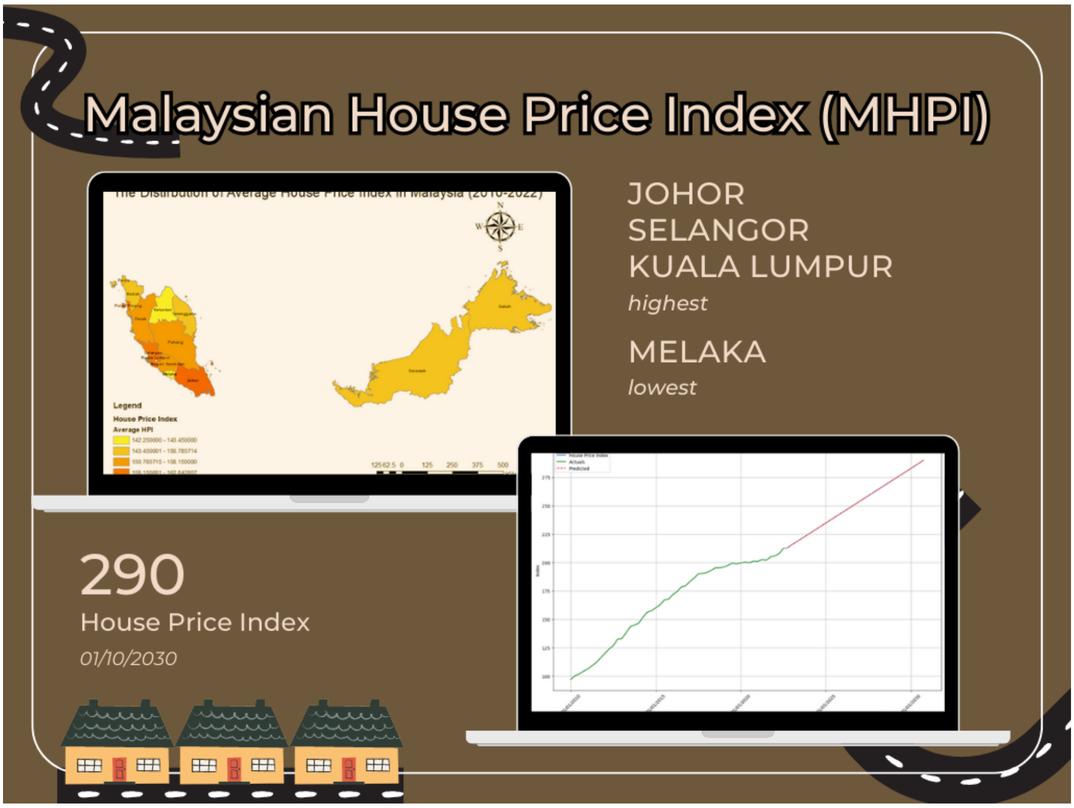
Overall, the lower values of MAE, MSE, and RMSE indicate that the LSTM model performs reasonably well in predicting the house price index. It suggests that the model's predictions are relatively close to the actual values.

CHAPTER 5

RESULTS AND CONCLUSIONS

This project has been focused on helping the government to achieve the Twelfth Malaysian Plan (RMK12) which specifically in Malaysian House Price Index (MHPI). This study is also relevant to the future of the RMK12 plan. Three courses were combined to do the forecasting of Malaysian House price Index (MHPI).

First, this project succeeded in forecasting the future trend of Malaysian House Price Index (MHPI) by using Double Exponential Smoothing (Holt's Method), which the model evaluated resulted in the existence of a non-seasonal trend. Second, this project determined that Johor has the highest average house price index indicating that houses in Johor tend to have higher average compared to other states while Melaka has the lowest average house price index. Most states managed to record slight growth, including Kuala Lumpur, Selangor and Johor. The LSTM model in the time series analysis of the house price index proved beneficial. This approach allowed for capturing long-term dependencies and making predictions about future trends based on the available house price index data.



In a nutshell, Double Exponential Smoothing has the lowest error metrics (MAE, MSE, and RMSE) and the smallest average prediction error. DES also has the best overall performance which is better than the performance of LSTM in terms of prediction of accuracy.

In conclusion, the continuous increase in house prices can have several implications and potential consequences. As the price of houses keeps rising, it can lead to various challenges and concerns for both individuals and the overall economy. High house prices may result in reduced affordability, making it more difficult for individuals, particularly first-time homebuyers, to enter the housing market. Thus, these findings are important for governments, policymakers, and stakeholders to pay close attention and take necessary steps to maintain a stable housing market. This involves addressing affordability issues, encouraging responsible lending practices, and creating an environment that supports the development of housing. Hence, it can minimise the negative impacts of increasing house prices and ensure a sustainable and accessible housing market for everyone.

REFERENCES

Abdullah, J., Zanudin, K., & Marzukhi, M. A. (2022). TWELFTH MALAYSIA PLAN: PROSPECTIVE IMPACTS ON URBAN AND REGIONAL DEVELOPMENT. *PLANNING MALAYSIA*, *20*. https://doi.org/10.21837/pm.v20i23.1170

Park, K., Jung, Y., Kim, K., & Park, S. K. (2020). Determination of deep learning model and optimum length of training data. *Water (Switzerland)*, *12*(12). https://doi.org/10.3390/w12123537

Graves, A., & Schmidhuber, J. (2005). Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks*, *18*(5), 602–610. https://doi.org/https://doi.org/10.1016/j.neunet.2005.06.042

House, M. (2023). *INDEKS HARGA RUMAH MALAYSIA*. www.jpph.gov.my

Silver, M. (2012). *Why House Price Indexes Differ: Measurement and Analysis; by Mick Silver; IMF Working Paper 12/125; May 1, 2012*.

Appendix A (TSA)

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
data = pd.read_csv("MHPIquarter.csv",header = 0, index_col = 0)
data.head()
```

Out[1]:

	House Price Index
Date	
01/01/2010	97.2
01/04/2010	100.4
01/07/2010	102.0
01/10/2010	104.3
01/01/2011	106.4

In [2]:
```
data.describe()
```

Out[2]:

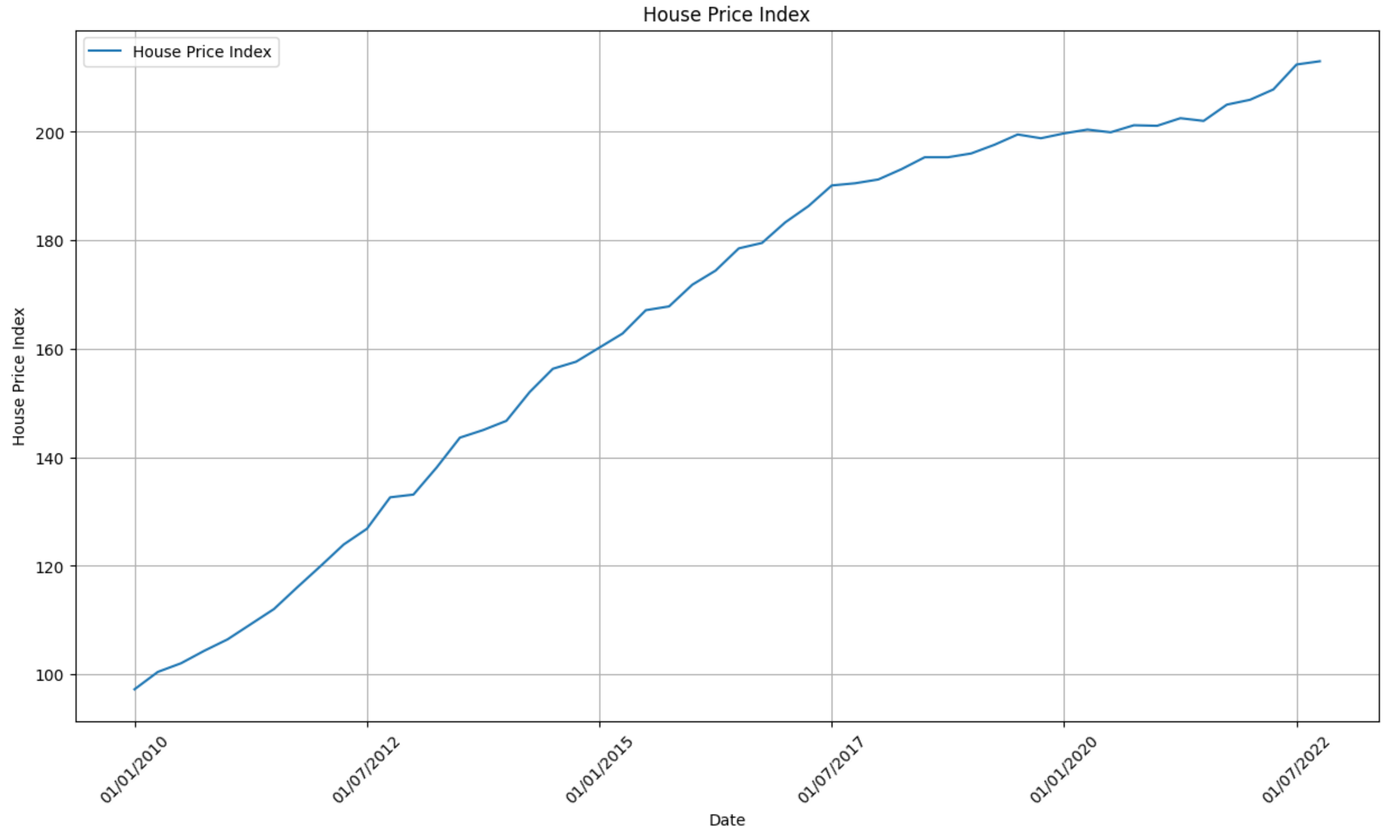
	House Price Index
count	52.000000
mean	166.405769
std	36.128417
min	97.200000
25%	136.850000
50%	176.450000
75%	198.975000
max	213.000000

In [3]:
```
data.isna().sum()
```

Out[3]:
```
House Price Index    0
dtype: int64
```

In [4]:
```
#plot the time series data
from matplotlib import pyplot
data.plot(figsize = (15, 8), title = 'House Price Index', fontsize = 10)
# Set labels and title
plt.xlabel('Date')
plt.ylabel('House Price Index')

# Add gridlines
plt.grid(True)

# Rotate x-axis labels for better readability
plt.xticks(rotation=45)

# Add a legend
plt.legend()
```

```
# Display the plot
plt.show()
```



In [5]:

```
#Split data training (in-sample) & testing (out-sample)
training = data[0:42]
testing = data[42:]
```

In [6]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from statsmodels.tsa.api import SimpleExpSmoothing

# Create the simple exponential smoothing model and fit it to the training data
model = SimpleExpSmoothing(training)
model = model.fit(smoothing_level=0.6)
model.summary()
```

```
C:\Users\Asus\anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:471: ValueWa
rning: A date index has been provided, but it has no associated frequency information an
d so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
```

Out[6]:

SimpleExpSmoothing Model Results

Dep. Variable:	House Price Index	**No. Observations:**	42
Model:	SimpleExpSmoothing	**SSE**	861.708
Optimized:	True	**AIC**	130.892
Trend:	None	**BIC**	134.368
Seasonal:	None	**AICC**	131.973
Seasonal Periods:	None	**Date:**	Sun, 25 Jun 2023
Box-Cox:	False	**Time:**	15:55:41
Box-Cox Coeff.:	None		

	coeff	code	optimized
smoothing_level	0.6000000	alpha	False
initial_level	98.987870	l.0	True

In [7]:
```python
# initialize variables for storing predictions and actual values
predictions = []
actuals = []

# iterate over each time step in the testing data
for i in range(len(testing)):
    # make one-step ahead forecast
    yhat = model.forecast()

    # store prediction and actual value
    predictions.append(yhat)
    actuals.append(testing.iloc[i])

    # add actual value to training data
    training = pd.concat([training, pd.DataFrame([testing.iloc[i]], columns=training.col

    # retrain model on updated training data
    model = SimpleExpSmoothing(training)
    model = model.fit(smoothing_level=0.6)

    last_actual_value = testing.iloc[-1]
```

```
C:\Users\Asus\anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:834: ValueWa
rning: No supported index is available. Prediction results will be given with an integer
index beginning at `start`.
  return get_prediction_index(
```

In [8]:
```python
actuals
```

Out[8]:
```
[House Price Index    199.9
 Name: 01/07/2020, dtype: float64,
 House Price Index    201.2
 Name: 01/10/2020, dtype: float64,
 House Price Index    201.1
 Name: 01/01/2021, dtype: float64,
 House Price Index    202.5
 Name: 01/04/2021, dtype: float64,
 House Price Index    202.0
 Name: 01/07/2021, dtype: float64,
 House Price Index    205.0
 Name: 01/10/2021, dtype: float64,
 House Price Index    205.9
 Name: 01/01/2022, dtype: float64,
 House Price Index    207.8
 Name: 01/04/2022, dtype: float64,
 House Price Index    212.4
 Name: 01/07/2022, dtype: float64,
 House Price Index    213.0
 Name: 01/10/2022, dtype: float64]
```

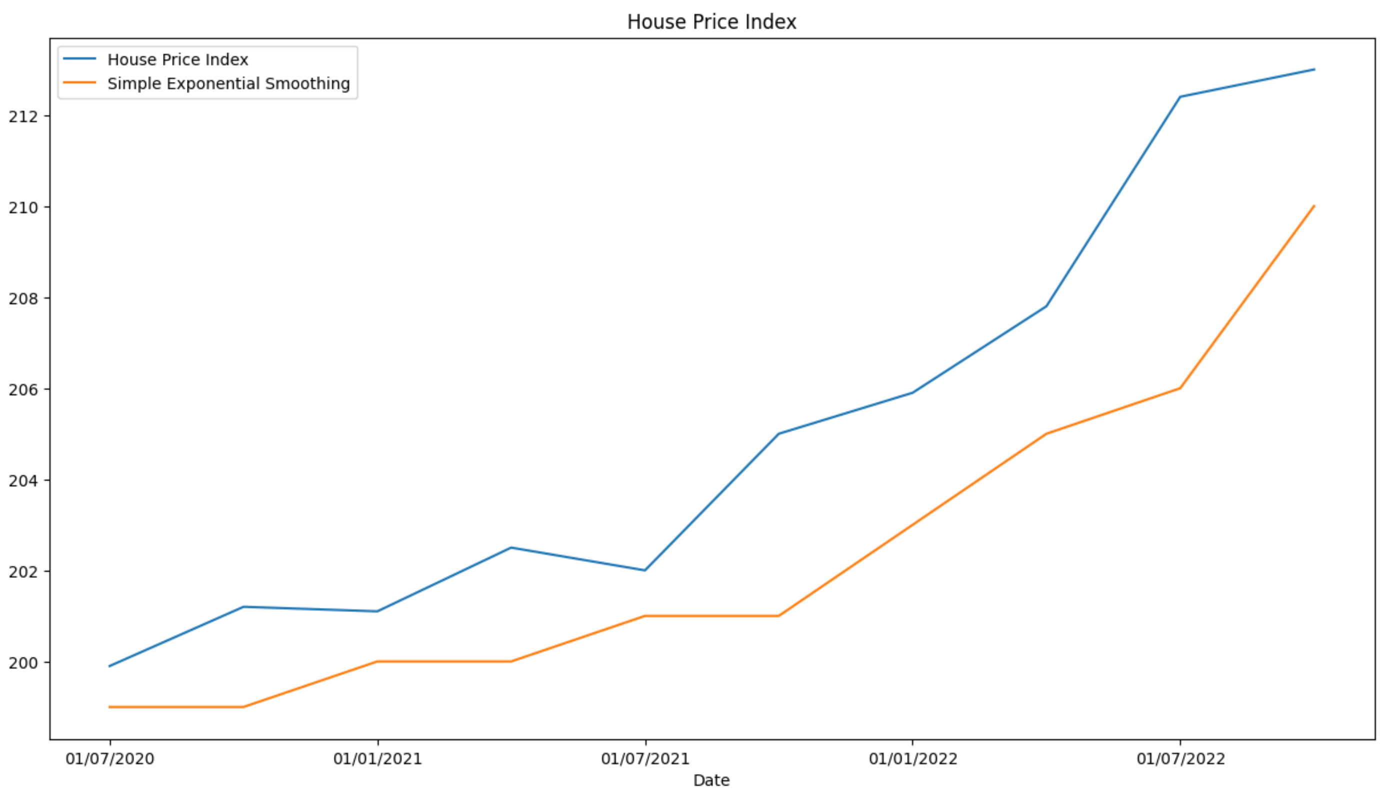
In [9]:
```python
predictions
```

Out[9]:
```
[42    199.950781
 dtype: float64,
 43    199.920312
 dtype: float64,
 44    200.688125
 dtype: float64,
 45    200.93525
 dtype: float64,
```

```
46    201.8741
dtype: float64,
47    201.94964
dtype: float64,
48    203.779856
dtype: float64,
49    205.051942
dtype: float64,
50    206.700777
dtype: float64,
51    210.120311
dtype: float64]
```

In [10]:

```
import numpy as np
a=np.array(predictions) #convert predictions into array
a=np.asarray(a, dtype = 'int') #convert into integer
testing_forecasted=pd.DataFrame(a, columns=['Simple Exponential Smoothing']) #convert in
testing=testing.reset_index()#reset index testing data
frames=[testing, testing_forecasted]#combine actual and forecast testing data
result = pd.concat(frames, axis=1)
result= result.set_index('Date')
#plot comparison actual out-sample with forecasted out-sample
import matplotlib.pyplot as plt
result.plot(figsize=(15, 8),title = 'House Price Index', fontsize = 10)
plt.show()
```



In [11]:

```
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
from statsmodels.tools.eval_measures import rmse
forecast_rmse_error = rmse(testing['House Price Index'], testing_forecasted['Simple Expo
forecast_mse_error = forecast_rmse_error**2
forecast_mean_value = data['House Price Index'].mean()
forecast_mae_error = mean_absolute_error(testing['House Price Index'], testing_forecaste

print(f"MAE: {forecast_mae_error}")
print(f"MSE Error: {forecast_mse_error}\nRMSE Error: {forecast_rmse_error}\nMean: {forec
```

```
MAE: 2.680000000000001
MSE Error: 9.632000000000014
RMSE Error: 3.103546358603334
Mean: 166.4057692307692
```

In [12]:
```
# Generate forecasts for the next n_periods points beyond the end of the testing set
n_periods = 32 # one step ahead
forecasts1=model.forecast(steps=n_periods)
forecasts1
```

Out[12]:
```
52    211.848124
53    211.848124
54    211.848124
55    211.848124
56    211.848124
57    211.848124
58    211.848124
59    211.848124
60    211.848124
61    211.848124
62    211.848124
63    211.848124
64    211.848124
65    211.848124
66    211.848124
67    211.848124
68    211.848124
69    211.848124
70    211.848124
71    211.848124
72    211.848124
73    211.848124
74    211.848124
75    211.848124
76    211.848124
77    211.848124
78    211.848124
79    211.848124
80    211.848124
81    211.848124
82    211.848124
83    211.848124
dtype: float64
```

Holt's method

In [13]:
```
data.head()
```

Out[13]:

	House Price Index
Date	
01/01/2010	97.2
01/04/2010	100.4
01/07/2010	102.0
01/10/2010	104.3
01/01/2011	106.4

In [14]:
```
#Split data training (in-sample) & testing (out-sample)
training = data[0:42]
testing = data[42:]
```

In [15]:
```
from statsmodels.tsa.api import ExponentialSmoothing, SimpleExpSmoothing, Holt
```

```python
# Fit the double exponential smoothing model to the training data
model2 = Holt(training, initialization_method="known", initial_level=98.987870, initial_
model2 = model2.fit(smoothing_level=0.6, smoothing_trend=0.5, optimized=False)
model2.summary()
```

```
C:\Users\Asus\anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:471: ValueWa
rning: A date index has been provided, but it has no associated frequency information an
d so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
```

Out[15]:

Holt Model Results

Dep. Variable:	House Price Index	**No. Observations:**	42
Model:	Holt	**SSE**	103.456
Optimized:	False	**AIC**	45.862
Trend:	Additive	**BIC**	52.813
Seasonal:	None	**AICC**	48.262
Seasonal Periods:	None	**Date:**	Sun, 25 Jun 2023
Box-Cox:	False	**Time:**	15:55:41
Box-Cox Coeff.:	None		

	coeff	code	optimized
smoothing_level	0.6000000	alpha	False
smoothing_trend	0.5000000	beta	False
initial_level	98.987870	l.0	False
initial_trend	0.000000	b.0	False

In [16]:

```python
# initialize variables for storing predictions and actual values
predictions2 = []
actuals = []

# iterate over each time step in the testing data
for i in range(len(testing)):
    # make one-step ahead forecast
    yhat = model2.forecast()

    # store prediction and actual value
    predictions2.append(yhat)
    actuals.append(testing.iloc[i])

    # add actual value to training data
    training = pd.concat([training, pd.DataFrame([testing.iloc[i]], columns=training.col

    # retrain model on updated training data
    model2 = Holt(training, initialization_method="known", initial_level=98.987870, init
    model2 = model2.fit(smoothing_level=0.6, smoothing_trend=0.5, optimized=False)

    last_actual_value = testing.iloc[-1]
```

```
C:\Users\Asus\anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:834: ValueWa
rning: No supported index is available. Prediction results will be given with an integer
index beginning at `start`.
  return get_prediction_index(
```

In [17]:

```python
predictions2
```

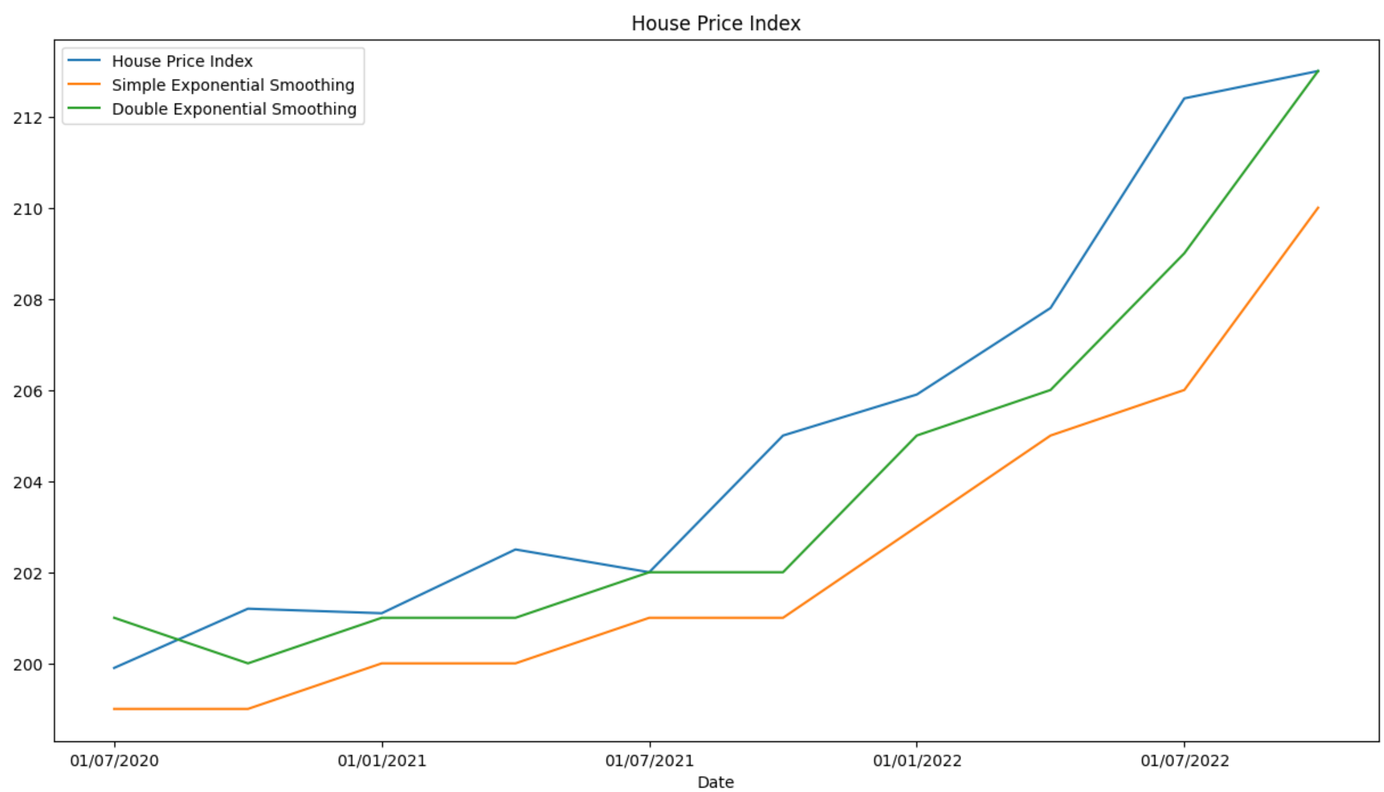
Out[17]:

```
[42    201.074574
 dtype: float64,
```

```
43    200.623735
dtype: float64,
44    201.396278
dtype: float64,
45    201.556412
dtype: float64,
46    202.743542
dtype: float64,
47    202.695332
dtype: float64,
48    205.167448
dtype: float64,
49    206.91606
dtype: float64,
50    209.020687
dtype: float64,
51    213.636332
dtype: float64]
```

In [18]:

```
import numpy as np
b=np.array(predictions2) #convert predictions into array
b=np.asarray(b, dtype = 'int') #convert into integer
testing_forecasted2=pd.DataFrame(b, columns=['Double Exponential Smoothing']) #convert i
testing=testing.reset_index()#reset index testing data
frames=[testing, testing_forecasted, testing_forecasted2]#combine actual and forecast te
result = pd.concat(frames, axis=1)
result= result.set_index('Date')
#plot comparison actual out-sample with forecasted out-sample
import matplotlib.pyplot as plt
result.plot(figsize=(15, 8),title = 'House Price Index', fontsize = 10)
plt.show()
```



In [19]:

```
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
from statsmodels.tools.eval_measures import rmse
forecast2_rmse_error = rmse(testing['House Price Index'], testing_forecasted2['Double Ex
forecast2_mse_error = forecast2_rmse_error**2
forecast2_mean_value = data['House Price Index'].mean()
forecast2_mae_error = mean_absolute_error(testing['House Price Index'], testing_forecast
```

```
print(f"MAE: {forecast2_mae_error}")
print(f"MSE Error: {forecast2_mse_error}\nRMSE Error: {forecast2_rmse_error}\nMean: {for
```

```
MAE: 1.3
MSE Error: 2.952000000000005
RMSE Error: 1.7181385275931638
Mean: 166.4057692307692
```

In [20]:
```
# Generate forecasts for the next n_periods points beyond the end of the testing set
n_periods = 32 # one step ahead
forecasts2=model2.forecast(steps=n_periods)
forecasts2
```

Out[20]:
```
52    215.651690
53    218.048847
54    220.446005
55    222.843162
56    225.240319
57    227.637477
58    230.034634
59    232.431791
60    234.828949
61    237.226106
62    239.623263
63    242.020421
64    244.417578
65    246.814735
66    249.211893
67    251.609050
68    254.006207
69    256.403365
70    258.800522
71    261.197679
72    263.594837
73    265.991994
74    268.389151
75    270.786309
76    273.183466
77    275.580623
78    277.977781
79    280.374938
80    282.772096
81    285.169253
82    287.566410
83    289.963568
dtype: float64
```

In [21]:
```
forecasts2
dse= pd.DataFrame(forecasts2)
dse.to_csv('HPI_dse.csv')
```

Holt-Winter's method

In [22]:
```
data.head()
```

Out[22]:

	House Price Index
Date	
01/01/2010	97.2
01/04/2010	100.4
01/07/2010	102.0

01/10/2010	104.3
01/01/2011	106.4

In [23]:
```
#Split data training (in-sample) & testing (out-sample)
training = data[0:42]
testing = data[42:]
```

In [24]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from statsmodels.tsa.api import ExponentialSmoothing, SimpleExpSmoothing, Holt

# Fit triple exponential smoothing model without seasonal component and quarterly period
model3 = ExponentialSmoothing(training['House Price Index'], trend='add', seasonal=None,
model3.summary()
```

```
C:\Users\Asus\anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:471: ValueWa
rning: A date index has been provided, but it has no associated frequency information an
d so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
```

Out[24]:

ExponentialSmoothing Model Results

Dep. Variable:	House Price Index	**No. Observations:**	42
Model:	ExponentialSmoothing	**SSE**	78.607
Optimized:	True	**AIC**	34.325
Trend:	Additive	**BIC**	41.276
Seasonal:	None	**AICC**	36.725
Seasonal Periods:	None	**Date:**	Sun, 25 Jun 2023
Box-Cox:	False	**Time:**	15:55:42
Box-Cox Coeff.:	None		

	coeff	**code**	**optimized**
smoothing_level	0.5243658	alpha	True
smoothing_trend	0.5243658	beta	True
initial_level	95.279551	l.0	True
initial_trend	2.1521446	b.0	True

In [25]:
```
# initialize variables for storing predictions and actual values
predictions3 = []
actuals = []

# iterate over each time step in the testing data
for i in range(len(testing)):
    # make one-step ahead forecast
    yhat = model3.forecast()

    # store prediction and actual value
    predictions3.append(yhat)
    actuals.append(testing.iloc[i])

    # add actual value to training data
    training = pd.concat([training, pd.DataFrame([testing.iloc[i]], columns=training.col

    # retrain model on updated training data
```

```
    model3 = ExponentialSmoothing(training['House Price Index'], trend='add', seasonal=N
    model3.summary()

    last_actual_value = testing.iloc[-1]
```

```
C:\Users\Asus\anaconda3\lib\site-packages\statsmodels\tsa\base\tsa_model.py:834: ValueWa
rning: No supported index is available. Prediction results will be given with an integer
index beginning at `start`.
  return get_prediction_index(
```

In [26]:

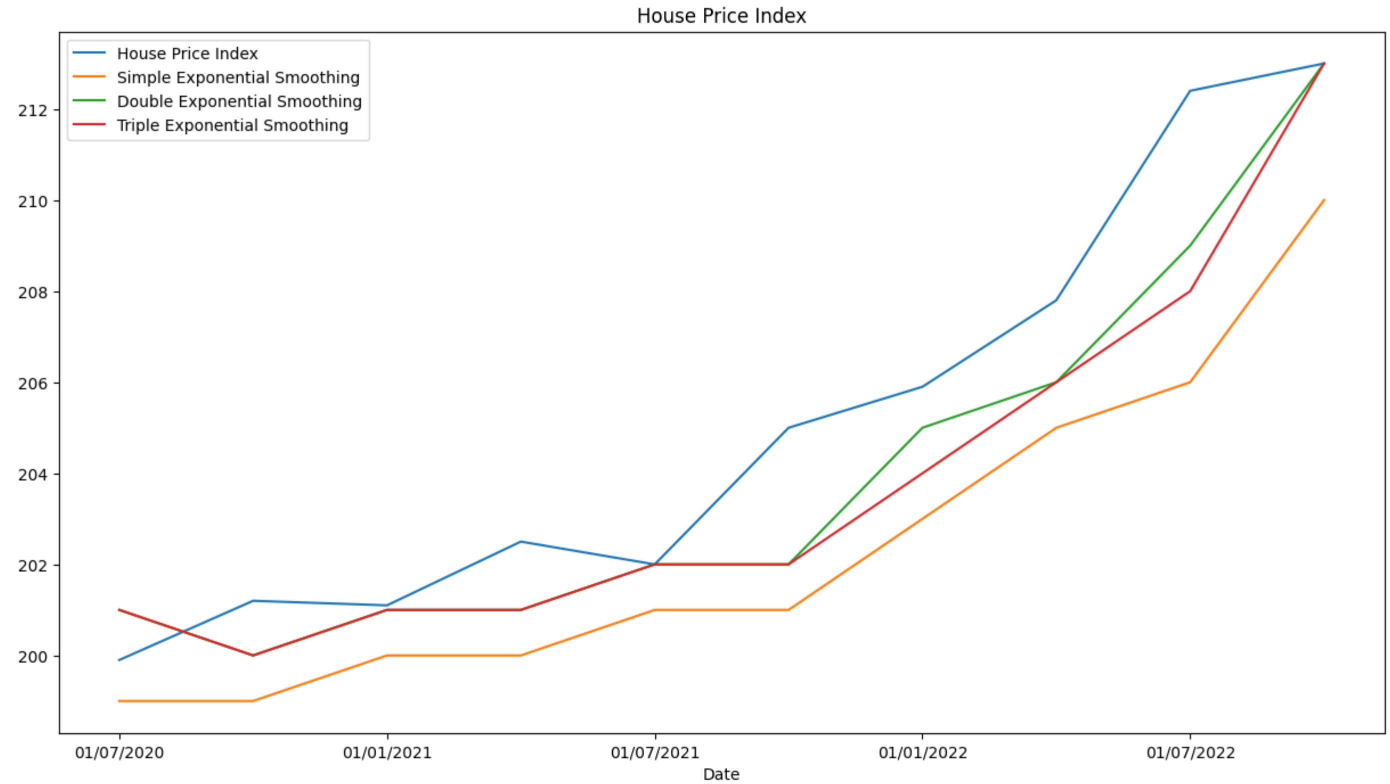
```
predictions3
```

Out[26]:

```
[42    201.12456
 dtype: float64,
 43    200.743037
 dtype: float64,
 44    201.379509
 dtype: float64,
 45    201.548328
 dtype: float64,
 46    202.62449
 dtype: float64,
 47    202.702294
 dtype: float64,
 48    204.939662
 dtype: float64,
 49    206.775795
 dtype: float64,
 50    208.956056
 dtype: float64,
 51    213.610948
 dtype: float64]
```

In [27]:

```
import numpy as np
c=np.array(predictions3) #convert predictions into array
c=np.asarray(c, dtype = 'int') #convert into integer
testing_forecasted3=pd.DataFrame(c, columns=['Triple Exponential Smoothing']) #convert i
testing=testing.reset_index()#reset index testing data
frames=[testing, testing_forecasted, testing_forecasted2, testing_forecasted3]#combine a
result = pd.concat(frames, axis=1)
result= result.set_index('Date')
#plot comparison actual out-sample with forecasted out-sample
import matplotlib.pyplot as plt
result.plot(figsize=(15, 8),title = 'House Price Index', fontsize = 10)
plt.show()
```



In [28]:
```
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
from statsmodels.tools.eval_measures import rmse
forecast3_rmse_error = rmse(testing['House Price Index'], testing_forecasted3['Triple Ex
forecast3_mse_error = forecast3_rmse_error**2
forecast3_mean_value = data['House Price Index'].mean()
forecast3_mae_error = mean_absolute_error(testing['House Price Index'], testing_forecast

print(f"MAE: {forecast3_mae_error}")
print(f"MSE Error: {forecast3_mse_error}\nRMSE Error: {forecast3_rmse_error}\nMean: {for
```

```
MAE: 1.5
MSE Error: 4.012000000000006
RMSE Error: 2.0029977533686867
Mean: 166.4057692307692
```

In [29]:
```
# Generate forecasts for the next n_periods points beyond the end of the testing set
n_periods = 32 # one step ahead
forecasts3=model3.forecast(steps=n_periods)
forecasts3
```

Out[29]:
```
52    215.741659
53    218.236183
54    220.730706
55    223.225230
56    225.719754
57    228.214277
58    230.708801
59    233.203325
60    235.697849
61    238.192372
62    240.686896
63    243.181420
64    245.675943
65    248.170467
66    250.664991
67    253.159514
68    255.654038
69    258.148562
70    260.643085
71    263.137609
```

```
72    265.632133
73    268.126656
74    270.621180
75    273.115704
76    275.610227
77    278.104751
78    280.599275
79    283.093798
80    285.588322
81    288.082846
82    290.577369
83    293.071893
dtype: float64
```

In [34]:

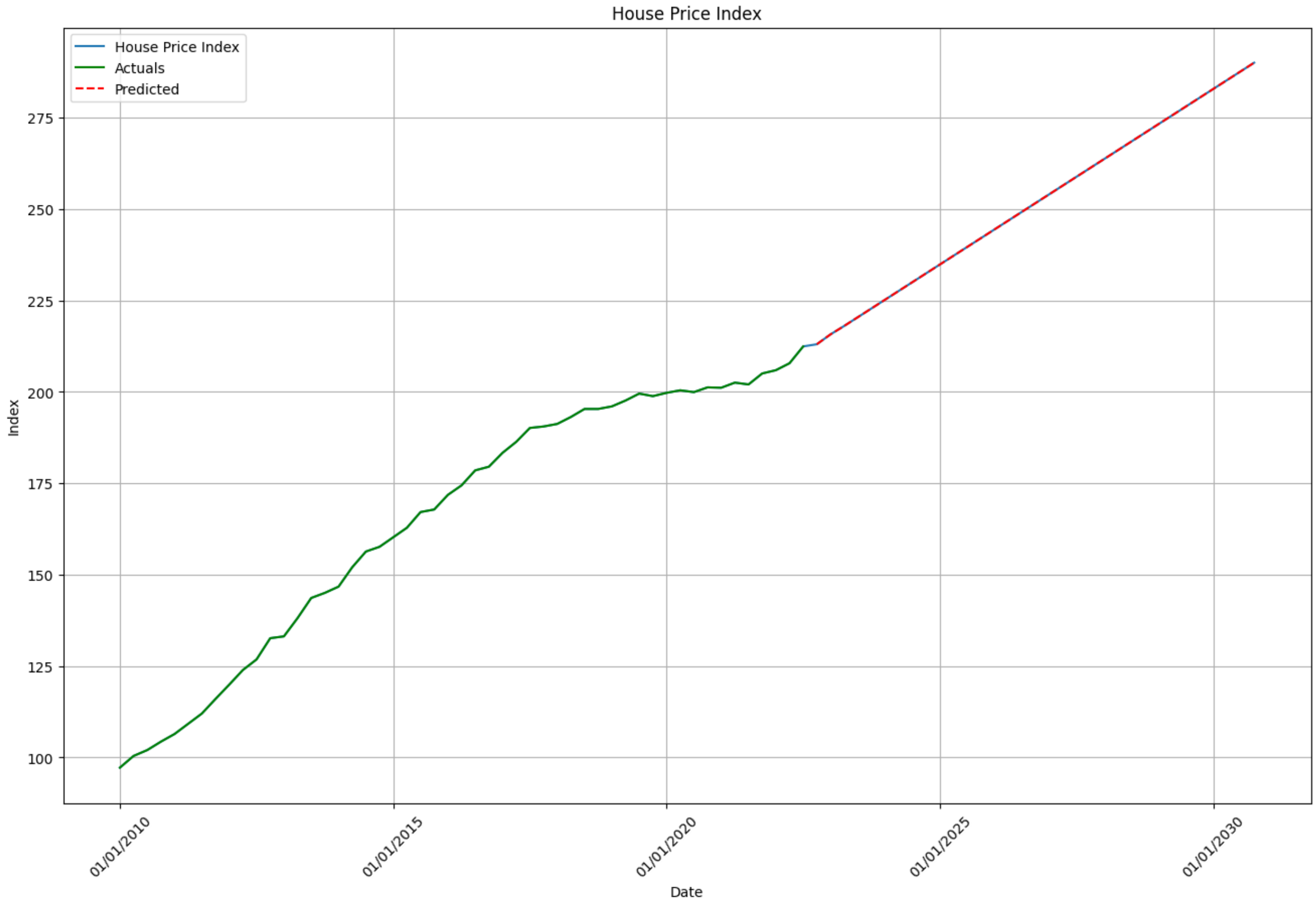
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
data = pd.read_csv("HPI_dse_final.csv",header = 0, index_col = 0)
data.tail()
```

Out[34]:

	House Price Index
Date	
01/10/2029	280.4
01/01/2030	282.8
01/04/2030	285.2
01/07/2030	287.6
01/10/2030	290.0

In [35]:

```
from matplotlib import pyplot as plt

# Plot the entire data
data.plot(figsize=(16, 10), title='House Price Index', fontsize=10)

# Define the start and end indices
start_index = 51
end_index = 84

# Plot the first segment with a solid green line
plt.plot(data.index[:start_index], data['House Price Index'][:start_index], color='green

# Plot the second segment with a dashed yellow line
plt.plot(data.index[start_index:end_index + 1], data['House Price Index'][start_index:en

# Set labels and title
plt.xlabel('Date')
plt.ylabel('Index')

# Add gridlines
plt.grid(True)

# Rotate x-axis labels for better readability
plt.xticks(rotation=45)

# Add a legend
plt.legend()

# Display the plot
plt.show()
```


House Price Index
House Price Index
Actuals
Predicted
Index
Date
275
250
225
200
175
150
125
100
01/01/2010
01/01/2015
01/01/2020
01/01/2025
01/01/2030

Appendix B (Deep Learning)

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
data = pd.read_csv("MHPIquarter.csv",header = 0, index_col = 0)
data.tail()
```

Out[1]:

Date	House Price Index
01/10/2021	205.0
01/01/2022	205.9
01/04/2022	207.8
01/07/2022	212.4
01/10/2022	213.0

In [2]:
```
data.describe()
```

Out[2]:

	House Price Index
count	52.000000
mean	166.405769
std	36.128417
min	97.200000
25%	136.850000
50%	176.450000
75%	198.975000
max	213.000000

In [3]:
```
#Split data training (in-sample) & testing (out-sample)
training = data[0:42]
testing = data[42:]
```

In [4]:
```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
```

In [5]:
```
scaler.fit(training[['House Price Index']])
scaled_train_data = scaler.transform(training[['House Price Index']])
scaled_test_data = scaler.transform(testing[['House Price Index']])
```

In [6]:
```
from keras.preprocessing.sequence import TimeseriesGenerator

n_input = 10
n_features= 1
generator = TimeseriesGenerator(scaled_train_data, scaled_train_data, length=n_input, ba
```

In [7]:
```
from keras.models import Sequential
from keras.layers import Dense
```

```python
from keras.layers import LSTM
# Reduce the model complexity:
# With a limited amount of data, a complex model may lead to overfitting.
# You can reduce the number of LSTM units in each layer or decrease the number of layers
# For example, you could try using fewer units, such as LSTM(64) or even LSTM(32)
# Modify the model architecture
lstm_model = Sequential()
lstm_model.add(LSTM(64, return_sequences=True, input_shape=(n_input, n_features)))
lstm_model.add(LSTM(32, return_sequences=False))
lstm_model.add(Dense(5))
lstm_model.add(Dense(1))
lstm_model.compile(optimizer='adam', loss='mse')

# Print the model summary
lstm_model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 10, 64)            16896

 lstm_1 (LSTM)               (None, 32)                12416

 dense (Dense)               (None, 5)                 165

 dense_1 (Dense)             (None, 1)                 6

=================================================================
Total params: 29,483
Trainable params: 29,483
Non-trainable params: 0
_________________________________________________________________
```

In [8]:

```python
# Train the model with modified settings
lstm_model.fit_generator(generator,epochs=10)
```

```
Epoch 1/10
C:\Users\Asus\AppData\Local\Temp\ipykernel_3572\982418532.py:2: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future version. Please use `Model.fit`, which supports generators.
  lstm_model.fit_generator(generator,epochs=10)
32/32 [==============================] - 4s 8ms/step - loss: 0.1078
Epoch 2/10
32/32 [==============================] - 0s 9ms/step - loss: 0.0056
Epoch 3/10
32/32 [==============================] - 0s 7ms/step - loss: 0.0011
Epoch 4/10
32/32 [==============================] - 0s 7ms/step - loss: 5.1933e-04
Epoch 5/10
32/32 [==============================] - 0s 8ms/step - loss: 5.5601e-04
Epoch 6/10
32/32 [==============================] - 0s 8ms/step - loss: 4.5186e-04
Epoch 7/10
32/32 [==============================] - 0s 8ms/step - loss: 5.3774e-04
Epoch 8/10
32/32 [==============================] - 0s 8ms/step - loss: 7.1869e-04
Epoch 9/10
32/32 [==============================] - 0s 9ms/step - loss: 7.6774e-04
Epoch 10/10
32/32 [==============================] - 0s 9ms/step - loss: 4.1424e-04
```

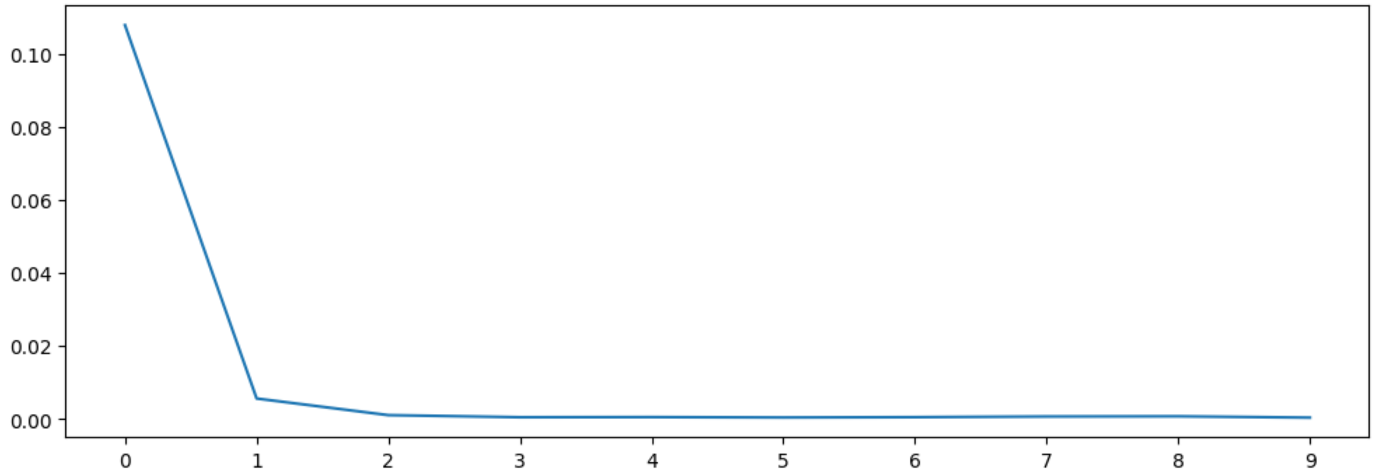
Out[8]:

```
<keras.callbacks.History at 0x286502bf1c0>
```

In [9]:

```python
losses_lstm = lstm_model.history.history['loss']
plt.figure(figsize=(12,4))
```

```
plt.xticks(np.arange(0,21,1))
plt.plot(range(len(losses_lstm)),losses_lstm);
```



In [10]:
```
lstm_predictions_scaled = list()

batch = scaled_train_data[-n_input:]
current_batch = batch.reshape((1, n_input, n_features))

for i in range(len(testing)):
    lstm_pred = lstm_model.predict(current_batch)[0]
    lstm_predictions_scaled.append(lstm_pred)
    current_batch = np.append(current_batch[:,1:,:],[[lstm_pred]],axis=1)
```

```
1/1 [==============================] - 1s 790ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 14ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 27ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 37ms/step
1/1 [==============================] - 0s 34ms/step
```

In [11]:
```
lstm_predictions_scaled
```

Out[11]:
```
[array([1.0299087], dtype=float32),
 array([1.0376477], dtype=float32),
 array([1.045003], dtype=float32),
 array([1.051794], dtype=float32),
 array([1.0591698], dtype=float32),
 array([1.0665799], dtype=float32),
 array([1.0734533], dtype=float32),
 array([1.0796205], dtype=float32),
 array([1.0864625], dtype=float32),
 array([1.0929877], dtype=float32)]
```

In [12]:
```
lstm_predictions = scaler.inverse_transform(lstm_predictions_scaled)
```

In [13]:
```
lstm_predictions
```

Out[13]:
```
array([[203.48657341],
       [204.28524513],
       [205.04431543],
       [205.74514618],
       [206.50632019],
       [207.27104959],
       [207.9803813 ],
       [208.61683359],
```

```
        [209.32292976],
        [209.99632616]])
```

In [14]:
```
testing['LSTM_Predictions'] = lstm_predictions
```

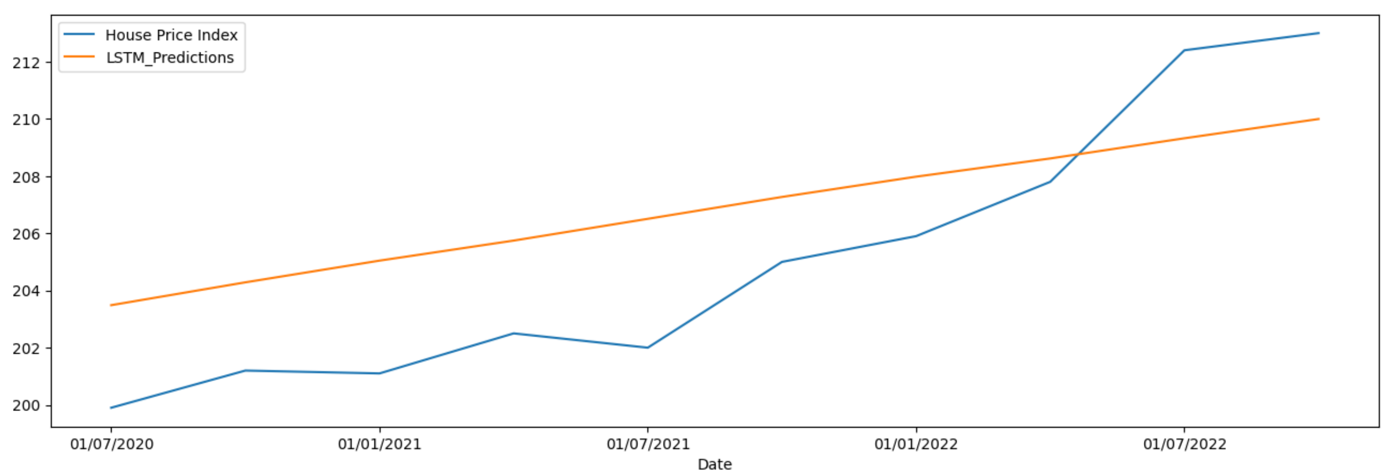
```
C:\Users\Asus\AppData\Local\Temp\ipykernel_3572\2224942254.py:1: SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
guide/indexing.html#returning-a-view-versus-a-copy
  testing['LSTM_Predictions'] = lstm_predictions
```

In [15]:
```
testing
```

Out[15]:

Date	House Price Index	LSTM_Predictions
01/07/2020	199.9	203.486573
01/10/2020	201.2	204.285245
01/01/2021	201.1	205.044315
01/04/2021	202.5	205.745146
01/07/2021	202.0	206.506320
01/10/2021	205.0	207.271050
01/01/2022	205.9	207.980381
01/04/2022	207.8	208.616834
01/07/2022	212.4	209.322930
01/10/2022	213.0	209.996326

In [16]:
```
testing['House Price Index'].plot(figsize = (16,5), legend=True)
testing['LSTM_Predictions'].plot(legend = True);
```



In [17]:
```
from sklearn.metrics import mean_squared_error, mean_absolute_error

# Calculate the mae, mse ,rmse
mae = mean_absolute_error(testing['House Price Index'], testing['LSTM_Predictions'])
mse = mean_squared_error(testing['House Price Index'], testing['LSTM_Predictions'])
rmse = np.sqrt(mean_squared_error(testing['House Price Index'], testing['LSTM_Prediction
# Print the MSE
print('MSE:', mse)
```

```python
print('RMSE:', rmse)
print('MAE:', mae)
```

```
MSE: 9.742105347179308
RMSE: 3.1212345870150977
MAE: 2.961660890579219
```

In [18]:

```python
n_periods = 32  # Number of periods to forecast

lstm_predictions_scaled = list()

batch = scaled_train_data[-n_input:]
current_batch = batch.reshape((1, n_input, n_features))

for i in range(n_periods):
    lstm_pred = lstm_model.predict(current_batch)[0]
    lstm_predictions_scaled.append(lstm_pred)
    current_batch = np.append(current_batch[:, 1:, :], [[lstm_pred]], axis=1)

# Rescale the predicted values to the original scale
lstm_predictions2 = scaler.inverse_transform(lstm_predictions_scaled)
```

```
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 9ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 27ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 10ms/step
```

In [20]:

```python
lstm_predictions2
lstm= pd.DataFrame(lstm_predictions2)
lstm.to_csv('HPI_lstm.csv')
```

In [21]:

```python
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
data = pd.read_csv("HPI_lstm_final.csv",header = 0, index_col = 0)
data.tail()
```

Out[21]:

House Price Index

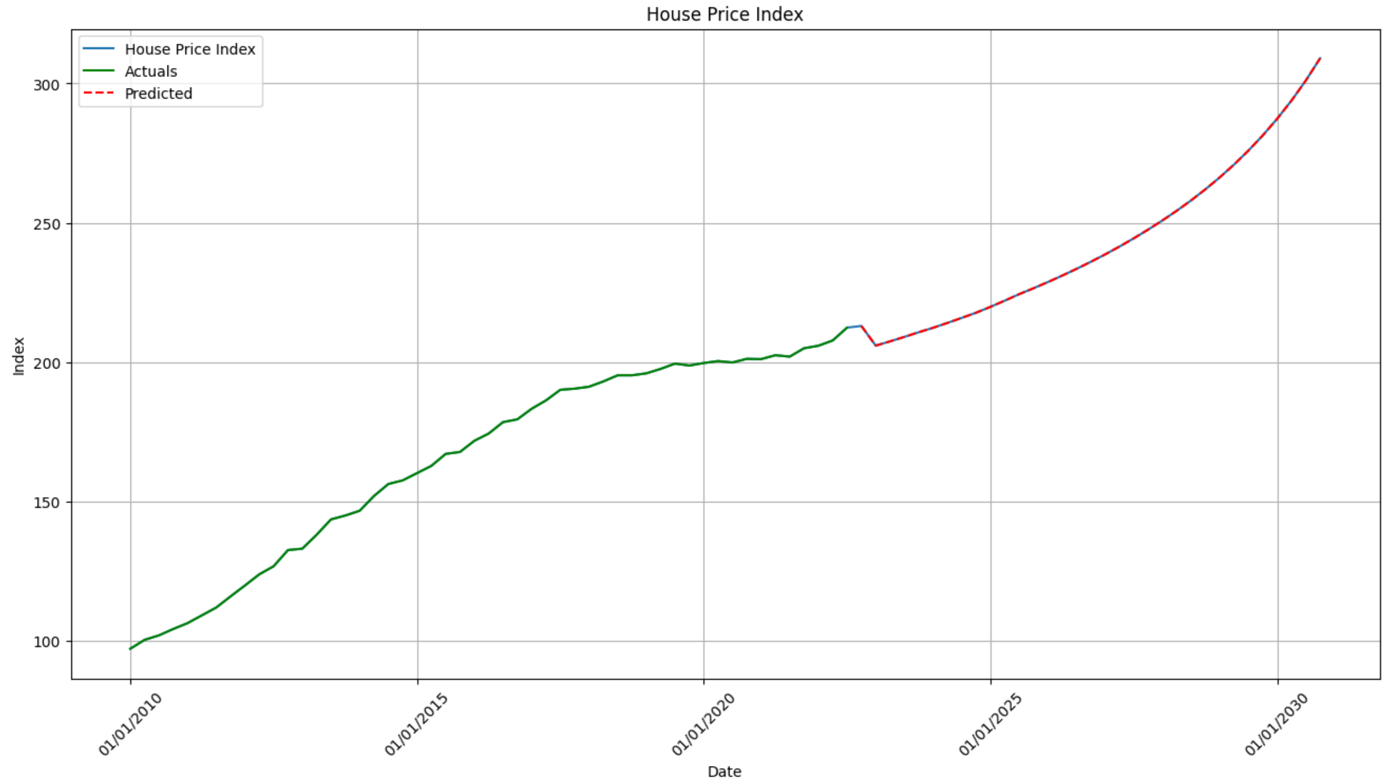
Date	
01/10/2029	281.364755
01/01/2030	287.308831
01/04/2030	293.835254
01/07/2030	301.039526
01/10/2030	309.037390

In [22]:
```
data.describe()
```

Out[22]:

	House Price Index
count	84.000000
mean	195.852727
std	50.532949
min	97.200000
25%	162.150000
50%	200.150000
75%	224.922070
max	309.037390

In [26]:
```
from matplotlib import pyplot as plt

# Plot the entire data
data.plot(figsize=(16, 8), title='House Price Index', fontsize=10)

# Define the start and end indices
start_index = 51
end_index = 84

# Plot the first segment with a solid green line
plt.plot(data.index[:start_index], data['House Price Index'][:start_index], color='green

# Plot the second segment with a dashed yellow line
plt.plot(data.index[start_index:end_index + 1], data['House Price Index'][start_index:en

# Set labels and title
plt.xlabel('Date')
plt.ylabel('Index')

# Add gridlines
plt.grid(True)

# Rotate x-axis labels for better readability
plt.xticks(rotation=45)

# Add a legend
plt.legend()

# Display the plot
plt.show()
```


House Price Index
House Price Index
Actuals
Predicted
300
250
200
150
100
Index
01/01/2010
01/01/2015
01/01/2020
01/01/2025
01/01/2030
Date