Advanced Machine Learning Analysis for Marketing

This project aims to explore Advanced Machine Learning techniques for analyzing retail data based on three years of sales data across 45 stores situated in diverse regions, each comprising multiple departments.

One of the primary challenges in modeling retail data stems from the necessity to make informed decisions in the face of limited data availability. Notably, sales surge during holidays and significant events, offering opportunities to evaluate the consequences of strategic choices on the overall performance. Moreover, the implementation of discounts and promotions can significantly influence sales outcomes. The primary objective of this endeavor is to forecast the potential impacts on specific departments and their extents.

Consequently, the central objectives to address involve the utilization of sophisticated machine learning methodologies to:

1. Predict sales at the department level for each individual store.
2. Model the ramifications of markdowns during holiday weeks.
3. Generate actionable marketing recommendations based on derived insights, with a focus on prioritizing actions that yield the most substantial business impact.

Agenda

This project involves the examination and prediction of store sales utilizing various techniques. We will begin by employing autocorrelation analysis to uncover time lag delays and subsequently adjust a dataset accordingly. A range of machine learning models will then be employed to forecast time series data, focusing on departmental weekly sales patterns.

Building upon neural network methodologies, we will explore the impact of markdowns on sales within the store, both during holiday periods and regular weeks. Subsequently, we will formulate a sales strategy tailored to a specific department.

The project can be broken down into the following stages:

1. Importing Libraries and Defining Auxiliary Functions
2. Data Downloading and Pre-processing
3. Predicting Department-wide Sales
   - Analysis of Previous Data
   - Creation of the Dataset
   - Data Normalization
   - Linear Regression
   - Back Propagation Neural Network
   - Long Short-Term Memory (LSTM)
4. Modeling the Effects of Markdowns on Holiday Weeks
   - Initial Analysis
   - Linear Regression
   - Back Propagation Neural Network
   - Sensitivity Analysis

5. Recommendations
6. Final Reflection and Comments

The statistical data was sourced from the website https://www.kaggle.com/manjeetsingh/retaildataset. This dataset is made available under the CC BY-IGO license, which grants the freedom to copy, adapt, distribute, and utilize the work, including for commercial purposes, without the need for explicit permission.

Importing Libraries and Defining Auxiliary Functions

In [77]:

```
# Import necessary libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import scikeras  # Importing scikeras for using Keras models with scikit-learn API

# Import specific modules from libraries
from statsmodels.graphics.tsaplots import acf, pacf, plot_acf, plot_pacf
from sklearn.preprocessing import MinMaxScaler
from sklearn.linear_model import LinearRegression
from sklearn import metrics
from scikeras.wrappers import KerasClassifier, KerasRegressor  # Import Keras wrappers
from keras.models import Sequential
from keras.layers import Dense, Dropout  # Import layers for building neural networks
from keras.callbacks import EarlyStopping  # Import EarlyStopping for model training
from keras.layers import LSTM  # Import LSTM layer for time series analysis
```

Data Downloading and Pre-processing

Data Downloading

We will obtain retail data relevant to store, department, and regional operations concerning the designated dates. Additionally, we have included the CSV file within this repository. This ensures access remains convenient either as an alternative to the provided link or for easier future retrieval.

In [78]:

```
# Load the dataset from the specified URL into a DataFrame
df1 = pd.read_csv('https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IB

# Assign a name to the DataFrame to indicate the dataset
df1.dataframeName = 'Features data set.csv'

# Display the loaded DataFrame
df1
```

Out[78]:

	Store	Date	Temperature	Fuel_Price	MarkDown1	MarkDown2	MarkDown3	MarkDown4	MarkDov
0	1	05/02/2010	42.31	2.572	NaN	NaN	NaN	NaN	N
1	1	12/02/2010	38.51	2.548	NaN	NaN	NaN	NaN	N
2	1	19/02/2010	39.93	2.514	NaN	NaN	NaN	NaN	N
3	1	26/02/2010	46.63	2.561	NaN	NaN	NaN	NaN	N
4	1	05/03/2010	46.50	2.625	NaN	NaN	NaN	NaN	N
...	...	...	...	...	...	...	...	...	

8185	45	28/06/2013	76.05	3.639	4842.29	975.03	3.00	2449.97	3169
8186	45	05/07/2013	77.50	3.614	9090.48	2268.58	582.74	5797.47	1514
8187	45	12/07/2013	79.37	3.614	3789.94	1827.31	85.72	744.84	2150
8188	45	19/07/2013	82.84	3.737	2961.49	1047.07	204.19	363.00	1059
8189	45	26/07/2013	76.06	3.804	212.02	851.73	2.06	10.88	1864

8190 rows × 12 columns

Let's examine this dataset. As observed, the dataset comprises 8,190 rows and 12 columns, with each column representing the following attributes:

- Store: The store number.
- Date: The week of observation.
- Temperature: The average temperature within the region.
- Fuel_Price: The cost of fuel in the region.
- MarkDown1-5: Anonymized data pertaining to promotional markdowns. MarkDown data is available after November 2011 and is not consistently present for all stores. Missing values are indicated as NA.
- CPI: The consumer price index.
- Unemployment: The rate of unemployment.
- IsHoliday: Indicates whether the week corresponds to a special holiday period.

Moving forward, our next step involves downloading historical sales data that spans from February 5, 2010, to November 1, 2012.

In [79]:

```python
# Load the sales dataset from the specified URL into a DataFrame
df2 = pd.read_csv('https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IB

# Assign a name to the DataFrame to indicate the dataset
df2.dataframeName = 'Sales data set.csv'

# Display the loaded DataFrame
df2
```

Out[79]:

	Store	Dept	Date	Weekly_Sales	IsHoliday
0	1	1	05/02/2010	24924.50	False
1	1	1	12/02/2010	46039.49	True
2	1	1	19/02/2010	41595.55	False
3	1	1	26/02/2010	19403.54	False
4	1	1	05/03/2010	21827.90	False
...	...	...	...	...	...
421565	45	98	28/09/2012	508.37	False
421566	45	98	05/10/2012	628.10	False
421567	45	98	12/10/2012	1061.02	False
421568	45	98	19/10/2012	760.01	False
421569	45	98	26/10/2012	1076.80	False

421570 rows × 5 columns

Observing the dataset, it comprises a total of 421,570 rows and encompasses 5 columns.

Contained within this dataset, you will encounter the subsequent information:

- Store: The specific store number.
- Dept: The designated department number.
- Date: The week under consideration.
- Weekly_Sales: The sales value corresponding to the specific department within the given store.
- IsHoliday: A flag indicating whether the week is designated as a special holiday week.

Lastly, the final dataset encapsulates anonymized details concerning the 45 stores, encompassing information about the store's type and size.

In [80]:

```
# Load the stores dataset from the specified URL into a DataFrame
df3 = pd.read_csv('https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IB

# Assign a name to the DataFrame to indicate the dataset
df3.dataframeName = 'Stores data set.csv'

# Display the loaded DataFrame
df3
```

Out[80]:

	Store	Type	Size
0	1	A	151315
1	2	A	202307
2	3	B	37392
3	4	A	205863
4	5	B	34875
5	6	A	202505
6	7	B	70713
7	8	A	155078
8	9	B	125833
9	10	B	126512
10	11	A	207499
11	12	B	112238
12	13	A	219622
13	14	A	200898
14	15	B	123737
15	16	B	57197
16	17	B	93188
17	18	B	120653
18	19	A	203819
19	20	A	203742
20	21	B	140167
21	22	B	119557

22	23	B	114533
23	24	A	203819
24	25	B	128107
25	26	A	152513
26	27	A	204184
27	28	A	206302
28	29	B	93638
29	30	C	42988
30	31	A	203750
31	32	A	203007
32	33	A	39690
33	34	A	158114
34	35	B	103681
35	36	A	39910
36	37	C	39910
37	38	C	39690
38	39	A	184109
39	40	A	155083
40	41	A	196321
41	42	C	39690
42	43	C	41062
43	44	C	39910
44	45	B	118221

Data Pre-processing

To begin with, our initial step involves merging these three datasets into a single entity using the **pandas.DataFrame.merge()** function.

In [81]:
```
# Merge df1 with df3 based on the 'Store' column
df = df1.merge(df3, on='Store')

# Merge df2 with the previously merged DataFrame (df) based on 'Store', 'Date', and 'IsH
df = df2.merge(df, on=['Store', 'Date', 'IsHoliday'])

# Display the resulting merged DataFrame
df
```

Out[81]:

	Store	Dept	Date	Weekly_Sales	IsHoliday	Temperature	Fuel_Price	MarkDown1	MarkDown2	M
0	1	1	05/02/2010	24924.50	False	42.31	2.572	NaN	NaN	
1	1	2	05/02/2010	50605.27	False	42.31	2.572	NaN	NaN	
2	1	3	05/02/2010	13740.12	False	42.31	2.572	NaN	NaN	

3	1	4	05/02/2010	39954.04	False	42.31	2.572	NaN	NaN
4	1	5	05/02/2010	32229.38	False	42.31	2.572	NaN	NaN
...	...	...	...	...	...	...	...	...	...
421565	45	93	26/10/2012	2487.80	False	58.85	3.882	4018.91	58.08
421566	45	94	26/10/2012	5203.31	False	58.85	3.882	4018.91	58.08
421567	45	95	26/10/2012	56017.47	False	58.85	3.882	4018.91	58.08
421568	45	97	26/10/2012	6817.48	False	58.85	3.882	4018.91	58.08
421569	45	98	26/10/2012	1076.80	False	58.85	3.882	4018.91	58.08

421570 rows × 16 columns

Let's explore the dataset. As evident, the dataset comprises 421,570 rows and 16 columns. Notably, the dataset encompasses diverse information types. It's crucial to ensure that Python accurately identifies the appropriate data types.

In [82]:
```
# Display summary information about the DataFrame
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 421570 entries, 0 to 421569
Data columns (total 16 columns):
 #   Column        Non-Null Count   Dtype
---  ------        --------------   -----
 0   Store         421570 non-null  int64
 1   Dept          421570 non-null  int64
 2   Date          421570 non-null  object
 3   Weekly_Sales  421570 non-null  float64
 4   IsHoliday     421570 non-null  bool
 5   Temperature   421570 non-null  float64
 6   Fuel_Price    421570 non-null  float64
 7   MarkDown1     150681 non-null  float64
 8   MarkDown2     111248 non-null  float64
 9   MarkDown3     137091 non-null  float64
 10  MarkDown4     134967 non-null  float64
 11  MarkDown5     151432 non-null  float64
 12  CPI           421570 non-null  float64
 13  Unemployment  421570 non-null  float64
 14  Type          421570 non-null  object
 15  Size          421570 non-null  int64
dtypes: bool(1), float64(10), int64(3), object(2)
memory usage: 51.9+ MB
```

To begin with, let's remove rows containing empty values:

In [83]:
```
# Fill missing values with zeros in the DataFrame
df = df.fillna(0)
```

As evident, we need to convert the 'Date' columns into DateTime format. Additionally, the 'Store' type needs to be categorized.

In [84]:
```
# Convert the 'Date' column to DateTime format
df['Date'] = pd.to_datetime(df['Date'])

# Convert the 'Type' column to categorical data type
df['Type'] = df['Type'].astype('category')
```

```python
# Display summary information about the DataFrame after transformations
df.info()

# - Ignore the warnings -
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 421570 entries, 0 to 421569
Data columns (total 16 columns):
 #   Column        Non-Null Count   Dtype
---  ------        --------------   -----
 0   Store         421570 non-null  int64
 1   Dept          421570 non-null  int64
 2   Date          421570 non-null  datetime64[ns]
 3   Weekly_Sales  421570 non-null  float64
 4   IsHoliday     421570 non-null  bool
 5   Temperature   421570 non-null  float64
 6   Fuel_Price    421570 non-null  float64
 7   MarkDown1     421570 non-null  float64
 8   MarkDown2     421570 non-null  float64
 9   MarkDown3     421570 non-null  float64
 10  MarkDown4     421570 non-null  float64
 11  MarkDown5     421570 non-null  float64
 12  CPI           421570 non-null  float64
 13  Unemployment  421570 non-null  float64
 14  Type          421570 non-null  category
 15  Size          421570 non-null  int64
dtypes: bool(1), category(1), datetime64[ns](1), float64(10), int64(3)
memory usage: 49.0 MB
```

```
C:\Users\nacho\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarn
ing: Parsing '19/02/2010' in DD/MM/YYYY format. Provide format or specify infer_datetime
_format=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\nacho\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarn
ing: Parsing '26/02/2010' in DD/MM/YYYY format. Provide format or specify infer_datetime
_format=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\nacho\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarn
ing: Parsing '19/03/2010' in DD/MM/YYYY format. Provide format or specify infer_datetime
_format=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\nacho\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarn
ing: Parsing '26/03/2010' in DD/MM/YYYY format. Provide format or specify infer_datetime
_format=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\nacho\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarn
ing: Parsing '16/04/2010' in DD/MM/YYYY format. Provide format or specify infer_datetime
_format=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\nacho\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarn
ing: Parsing '23/04/2010' in DD/MM/YYYY format. Provide format or specify infer_datetime
_format=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\nacho\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarn
ing: Parsing '30/04/2010' in DD/MM/YYYY format. Provide format or specify infer_datetime
_format=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\nacho\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarn
ing: Parsing '14/05/2010' in DD/MM/YYYY format. Provide format or specify infer_datetime
_format=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\nacho\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarn
ing: Parsing '21/05/2010' in DD/MM/YYYY format. Provide format or specify infer_datetime
_format=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\nacho\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarn
```

```
ing: Parsing '28/05/2010' in DD/MM/YYYY format. Provide format or specify infer_datetime
_format=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\nacho\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarn
ing: Parsing '18/06/2010' in DD/MM/YYYY format. Provide format or specify infer_datetime
_format=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\nacho\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarn
ing: Parsing '25/06/2010' in DD/MM/YYYY format. Provide format or specify infer_datetime
_format=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\nacho\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarn
ing: Parsing '16/07/2010' in DD/MM/YYYY format. Provide format or specify infer_datetime
_format=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\nacho\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarn
ing: Parsing '23/07/2010' in DD/MM/YYYY format. Provide format or specify infer_datetime
_format=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\nacho\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarn
ing: Parsing '30/07/2010' in DD/MM/YYYY format. Provide format or specify infer_datetime
_format=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\nacho\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarn
ing: Parsing '13/08/2010' in DD/MM/YYYY format. Provide format or specify infer_datetime
_format=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\nacho\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarn
ing: Parsing '20/08/2010' in DD/MM/YYYY format. Provide format or specify infer_datetime
_format=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\nacho\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarn
ing: Parsing '27/08/2010' in DD/MM/YYYY format. Provide format or specify infer_datetime
_format=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\nacho\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarn
ing: Parsing '17/09/2010' in DD/MM/YYYY format. Provide format or specify infer_datetime
_format=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\nacho\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarn
ing: Parsing '24/09/2010' in DD/MM/YYYY format. Provide format or specify infer_datetime
_format=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\nacho\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarn
ing: Parsing '15/10/2010' in DD/MM/YYYY format. Provide format or specify infer_datetime
_format=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\nacho\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarn
ing: Parsing '22/10/2010' in DD/MM/YYYY format. Provide format or specify infer_datetime
_format=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\nacho\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarn
ing: Parsing '29/10/2010' in DD/MM/YYYY format. Provide format or specify infer_datetime
_format=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\nacho\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarn
ing: Parsing '19/11/2010' in DD/MM/YYYY format. Provide format or specify infer_datetime
_format=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\nacho\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarn
ing: Parsing '26/11/2010' in DD/MM/YYYY format. Provide format or specify infer_datetime
_format=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\nacho\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarn
ing: Parsing '17/12/2010' in DD/MM/YYYY format. Provide format or specify infer_datetime
_format=True for consistent parsing.
```

```
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\nacho\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarn
ing: Parsing '24/12/2010' in DD/MM/YYYY format. Provide format or specify infer_datetime
_format=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\nacho\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarn
ing: Parsing '31/12/2010' in DD/MM/YYYY format. Provide format or specify infer_datetime
_format=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\nacho\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarn
ing: Parsing '14/01/2011' in DD/MM/YYYY format. Provide format or specify infer_datetime
_format=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\nacho\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarn
ing: Parsing '21/01/2011' in DD/MM/YYYY format. Provide format or specify infer_datetime
_format=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\nacho\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarn
ing: Parsing '28/01/2011' in DD/MM/YYYY format. Provide format or specify infer_datetime
_format=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\nacho\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarn
ing: Parsing '18/02/2011' in DD/MM/YYYY format. Provide format or specify infer_datetime
_format=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\nacho\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarn
ing: Parsing '25/02/2011' in DD/MM/YYYY format. Provide format or specify infer_datetime
_format=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\nacho\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarn
ing: Parsing '18/03/2011' in DD/MM/YYYY format. Provide format or specify infer_datetime
_format=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\nacho\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarn
ing: Parsing '25/03/2011' in DD/MM/YYYY format. Provide format or specify infer_datetime
_format=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\nacho\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarn
ing: Parsing '15/04/2011' in DD/MM/YYYY format. Provide format or specify infer_datetime
_format=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\nacho\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarn
ing: Parsing '22/04/2011' in DD/MM/YYYY format. Provide format or specify infer_datetime
_format=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\nacho\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarn
ing: Parsing '29/04/2011' in DD/MM/YYYY format. Provide format or specify infer_datetime
_format=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\nacho\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarn
ing: Parsing '13/05/2011' in DD/MM/YYYY format. Provide format or specify infer_datetime
_format=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\nacho\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarn
ing: Parsing '20/05/2011' in DD/MM/YYYY format. Provide format or specify infer_datetime
_format=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\nacho\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarn
ing: Parsing '27/05/2011' in DD/MM/YYYY format. Provide format or specify infer_datetime
_format=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\nacho\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarn
ing: Parsing '17/06/2011' in DD/MM/YYYY format. Provide format or specify infer_datetime
_format=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\nacho\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarn
```

```
ing: Parsing '24/06/2011' in DD/MM/YYYY format. Provide format or specify infer_datetime
_format=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\nacho\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarn
ing: Parsing '15/07/2011' in DD/MM/YYYY format. Provide format or specify infer_datetime
_format=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\nacho\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarn
ing: Parsing '22/07/2011' in DD/MM/YYYY format. Provide format or specify infer_datetime
_format=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\nacho\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarn
ing: Parsing '29/07/2011' in DD/MM/YYYY format. Provide format or specify infer_datetime
_format=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\nacho\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarn
ing: Parsing '19/08/2011' in DD/MM/YYYY format. Provide format or specify infer_datetime
_format=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\nacho\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarn
ing: Parsing '26/08/2011' in DD/MM/YYYY format. Provide format or specify infer_datetime
_format=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\nacho\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarn
ing: Parsing '16/09/2011' in DD/MM/YYYY format. Provide format or specify infer_datetime
_format=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\nacho\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarn
ing: Parsing '23/09/2011' in DD/MM/YYYY format. Provide format or specify infer_datetime
_format=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\nacho\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarn
ing: Parsing '30/09/2011' in DD/MM/YYYY format. Provide format or specify infer_datetime
_format=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\nacho\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarn
ing: Parsing '14/10/2011' in DD/MM/YYYY format. Provide format or specify infer_datetime
_format=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\nacho\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarn
ing: Parsing '21/10/2011' in DD/MM/YYYY format. Provide format or specify infer_datetime
_format=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\nacho\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarn
ing: Parsing '28/10/2011' in DD/MM/YYYY format. Provide format or specify infer_datetime
_format=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\nacho\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarn
ing: Parsing '18/11/2011' in DD/MM/YYYY format. Provide format or specify infer_datetime
_format=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\nacho\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarn
ing: Parsing '25/11/2011' in DD/MM/YYYY format. Provide format or specify infer_datetime
_format=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\nacho\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarn
ing: Parsing '16/12/2011' in DD/MM/YYYY format. Provide format or specify infer_datetime
_format=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\nacho\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarn
ing: Parsing '23/12/2011' in DD/MM/YYYY format. Provide format or specify infer_datetime
_format=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\nacho\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarn
ing: Parsing '30/12/2011' in DD/MM/YYYY format. Provide format or specify infer_datetime
_format=True for consistent parsing.
```

```
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\nacho\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarn
ing: Parsing '13/01/2012' in DD/MM/YYYY format. Provide format or specify infer_datetime
_format=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\nacho\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarn
ing: Parsing '20/01/2012' in DD/MM/YYYY format. Provide format or specify infer_datetime
_format=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\nacho\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarn
ing: Parsing '27/01/2012' in DD/MM/YYYY format. Provide format or specify infer_datetime
_format=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\nacho\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarn
ing: Parsing '17/02/2012' in DD/MM/YYYY format. Provide format or specify infer_datetime
_format=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\nacho\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarn
ing: Parsing '24/02/2012' in DD/MM/YYYY format. Provide format or specify infer_datetime
_format=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\nacho\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarn
ing: Parsing '16/03/2012' in DD/MM/YYYY format. Provide format or specify infer_datetime
_format=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\nacho\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarn
ing: Parsing '23/03/2012' in DD/MM/YYYY format. Provide format or specify infer_datetime
_format=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\nacho\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarn
ing: Parsing '30/03/2012' in DD/MM/YYYY format. Provide format or specify infer_datetime
_format=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\nacho\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarn
ing: Parsing '13/04/2012' in DD/MM/YYYY format. Provide format or specify infer_datetime
_format=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\nacho\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarn
ing: Parsing '20/04/2012' in DD/MM/YYYY format. Provide format or specify infer_datetime
_format=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\nacho\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarn
ing: Parsing '27/04/2012' in DD/MM/YYYY format. Provide format or specify infer_datetime
_format=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\nacho\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarn
ing: Parsing '18/05/2012' in DD/MM/YYYY format. Provide format or specify infer_datetime
_format=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\nacho\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarn
ing: Parsing '25/05/2012' in DD/MM/YYYY format. Provide format or specify infer_datetime
_format=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\nacho\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarn
ing: Parsing '15/06/2012' in DD/MM/YYYY format. Provide format or specify infer_datetime
_format=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\nacho\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarn
ing: Parsing '22/06/2012' in DD/MM/YYYY format. Provide format or specify infer_datetime
_format=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\nacho\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarn
ing: Parsing '29/06/2012' in DD/MM/YYYY format. Provide format or specify infer_datetime
_format=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\nacho\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarn
```

```
ing: Parsing '13/07/2012' in DD/MM/YYYY format. Provide format or specify infer_datetime
_format=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\nacho\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarn
ing: Parsing '20/07/2012' in DD/MM/YYYY format. Provide format or specify infer_datetime
_format=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\nacho\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarn
ing: Parsing '27/07/2012' in DD/MM/YYYY format. Provide format or specify infer_datetime
_format=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\nacho\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarn
ing: Parsing '17/08/2012' in DD/MM/YYYY format. Provide format or specify infer_datetime
_format=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\nacho\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarn
ing: Parsing '24/08/2012' in DD/MM/YYYY format. Provide format or specify infer_datetime
_format=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\nacho\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarn
ing: Parsing '31/08/2012' in DD/MM/YYYY format. Provide format or specify infer_datetime
_format=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\nacho\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarn
ing: Parsing '14/09/2012' in DD/MM/YYYY format. Provide format or specify infer_datetime
_format=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\nacho\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarn
ing: Parsing '21/09/2012' in DD/MM/YYYY format. Provide format or specify infer_datetime
_format=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\nacho\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarn
ing: Parsing '28/09/2012' in DD/MM/YYYY format. Provide format or specify infer_datetime
_format=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\nacho\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarn
ing: Parsing '19/10/2012' in DD/MM/YYYY format. Provide format or specify infer_datetime
_format=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
C:\Users\nacho\anaconda3\lib\site-packages\pandas\core\tools\datetimes.py:1047: UserWarn
ing: Parsing '26/10/2012' in DD/MM/YYYY format. Provide format or specify infer_datetime
_format=True for consistent parsing.
  cache_array = _maybe_cache(arg, format, cache, convert_listlike)
```

Given that stores and their respective departments vary across categories, sizes, quantities, and product assortments, while also being situated in distinct city regions, it would be erroneous to train a neural network on the entire dataset. Since departments located in different parts of the city exhibit distinct sales patterns despite using the same input data, it's evident that each department's information carries its individual variance. Consequently, for the analysis, it's imperative to isolate departments and conduct separate analyses for each of them.

Next, we will group the rows based on 'Store,' 'Department,' and 'Date'.

In [85]:
```python
# Group the DataFrame by 'Store,' 'Dept,' and 'Date,' and calculate the sum for each gro
grouped_data = df.groupby(['Store', 'Dept', 'Date']).sum()
```

Let's calculate number of rows for each department:

In [86]:
```python
# Count the occurrences of unique combinations of 'Store' and 'Dept' columns
value_counts = df[['Store', 'Dept']].value_counts()
```

Notice that the majority of departments consist of 143 rows each. We will now proceed to conduct an analysis for one of these departments.

In [87]:
```
# Assign the value 24 to the variable St (Store)
St = 24

# Assign the value 50 to the variable Dt (Department)
Dt = 50
```

Next, we will generate a dataset for Store: `St` and Department: `Dt`.

In [88]:
```
# Create a new DataFrame df_d by filtering rows where 'Store' is equal to St and 'Dept'
df_d = df[(df['Store'] == St) & (df['Dept'] == Dt)]

# Display the new DataFrame containing data for Store: St and Department: Dt
df_d
```

Out[88]:

	Store	Dept	Date	Weekly_Sales	IsHoliday	Temperature	Fuel_Price	MarkDown1	MarkDown2	MarkD
226912	24	50	2010-05-02	2030.0	False	22.43	2.954	0.00	0.00	
226985	24	50	2010-12-02	1535.0	True	25.94	2.940	0.00	0.00	
227059	24	50	2010-02-19	1570.0	False	31.05	2.909	0.00	0.00	
227130	24	50	2010-02-26	1350.0	False	33.98	2.910	0.00	0.00	
227201	24	50	2010-05-03	2700.0	False	36.73	2.919	0.00	0.00	
...	...	...	...	...	...	...	...	...	...	
236783	24	50	2012-09-28	1035.0	False	58.86	4.158	11941.13	15.28	
236854	24	50	2012-05-10	1005.0	False	60.35	4.151	10349.00	0.00	
236926	24	50	2012-12-10	1196.5	False	51.64	4.186	5138.51	0.00	1
236999	24	50	2012-10-19	1151.0	False	52.59	4.153	3446.70	0.00	1
237070	24	50	2012-10-26	595.0	False	55.16	4.071	10844.38	104.16	1

143 rows × 16 columns

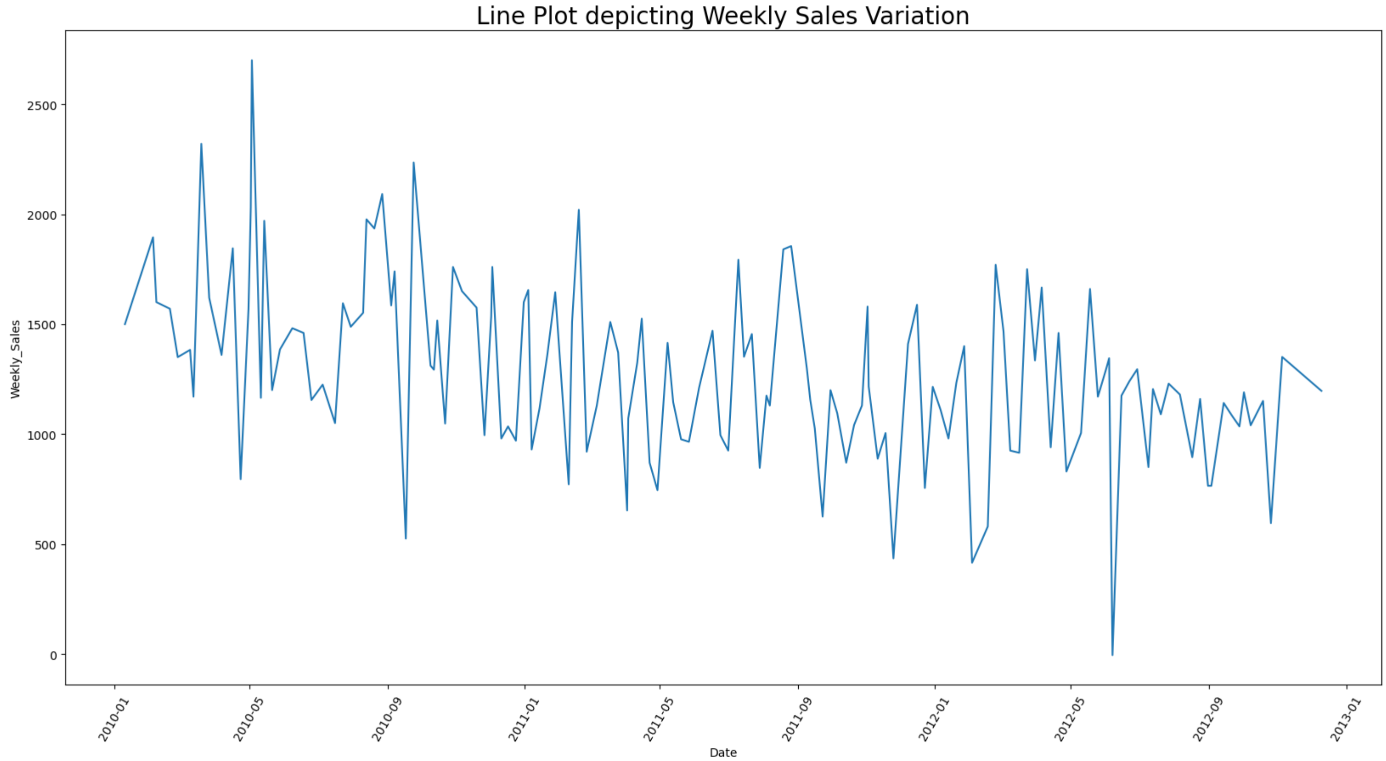
Predict Department-wide Sales

Analysis of Previous Data

We will now select the 'Weekly_Sales' field for our forecasting. To begin, let's visualize this data.

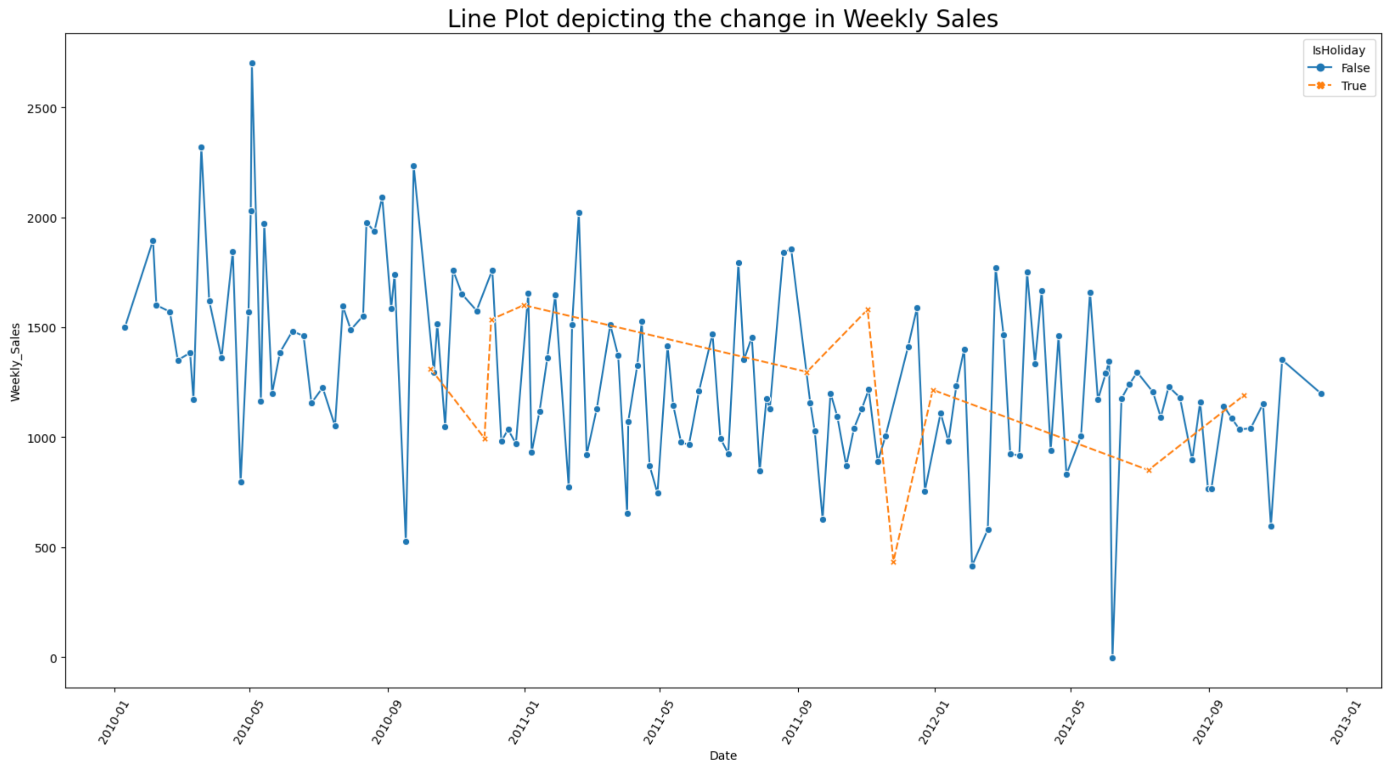
In [89]:
```
# Create a figure and subplots with a specified figsize
plt.figure(figsize=(20, 10))

# Rotate x-axis labels for better readability
```

```
plt.xticks(rotation=60)

# Create a line plot using Seaborn's lineplot function
_ = sns.lineplot(data=df_d, x='Date', y='Weekly_Sales')

# Set the title for the plot
_ = plt.title('Line Plot depicting Weekly Sales Variation', fontsize=20)

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



Next, we will visualize the fluctuations in sales during holiday periods.

In [90]:
```
# Create a figure and subplots with a specified figsize
plt.figure(figsize=(20, 10))

# Rotate x-axis labels for better readability
plt.xticks(rotation=60)

# Create a line plot using Seaborn's lineplot function
_ = sns.lineplot(data=df_d, x='Date', y='Weekly_Sales', hue='IsHoliday', style='IsHolida

# Set the title for the plot
_ = plt.title('Line Plot depicting the change in Weekly Sales', fontsize=20)

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



Observing the plot, it's evident that there isn't a sales increase during holidays.

The absence of a discernible sales increase during holidays, as depicted in our plot, suggests that holidays might not significantly impact the sales patterns for this particular department in Store St. This could be attributed to various factors, such as the department's product category or customer behavior. The plot's consistent trend across holiday and non-holiday periods indicates that the department's sales behavior remains relatively stable irrespective of holidays.

In order to forecast sales, we will generate an independent time series specifically comprising weekly sales data.

In [91]:
```
# Create a time series by extracting 'Date' and 'Weekly_Sales' columns from df_d
ts = df_d[['Date', 'Weekly_Sales']]

# Set the 'Date' column as the index for the time series
ts = ts.set_index('Date')

# Isolate the 'Weekly_Sales' data from the time series
ts = ts['Weekly_Sales']
```

When attempting to forecast a time series, our approach involves considering that today's data relies on values from preceding weeks. To assess these dependencies, conducting a correlation analysis becomes crucial. This process entails:

1. Duplicating the time series data and shifting it vertically downward by a designated number of days (lag).
2. Removing missing data generated by the vertical shift, which occurs due to (**pandas.DataFrame.shift()**)
3. Computing the correlation coefficient between the resultant series.

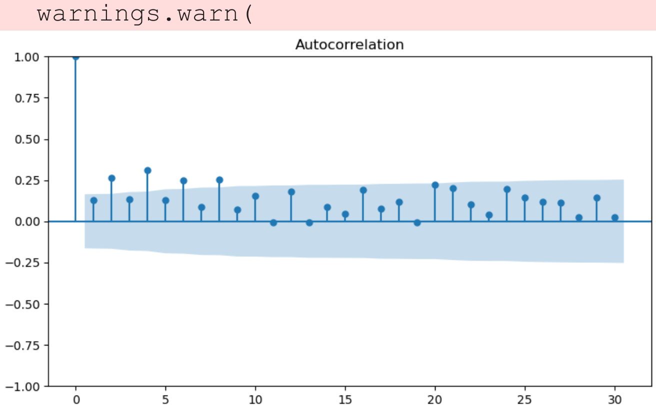
Given the necessity to perform this operation for various lag values, it's pragmatic to devise a dedicated function or utilize **statsmodels.graphics.tsaplots.plot_acf()**. Alternatively, the Partial Autocorrelation Function: (PACF) can be utilized using **statsmodels.graphics.tsaplots.plot_pacf()**.

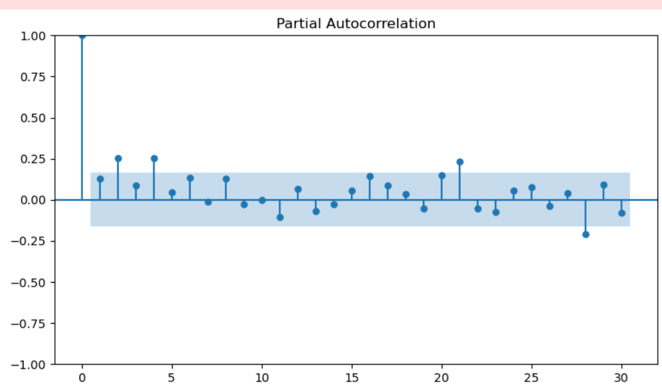
This analysis serves the purpose of identifying lag delays, indicating the number of weeks prior that influence today's sales, andcontributing significantly to our forecasting endeavors.

In [92]:
```
# Print the Correlation Coefficients using Autocorrelation Function (ACF) and Partial Au
print(pd.Series(acf(ts, nlags=10), name="Correlation Coeff"))
print(pd.Series(pacf(ts, nlags=10), name="Partial Correlation Coeff"))

# Create subplots for ACF and PACF plots
fig, axes = plt.subplots(1, 2, figsize=(20, 5))

# Plot Autocorrelation Function (ACF) with a specified number of lags
_ = plot_acf(ts, lags=30, ax=axes[0])

# Plot Partial Autocorrelation Function (PACF) with a specified number of lags
_ = plot_pacf(ts, lags=30, ax=axes[1])
```

```
0     1.000000
1     0.130103
2     0.263880
3     0.136014
4     0.311210
5     0.126184
6     0.245963
7     0.088694
8     0.253053
9     0.071838
10    0.154257
Name: Correlation Coeff, dtype: float64
0     1.000000
1     0.131019
2     0.254831
3     0.086411
4     0.254782
5     0.043069
6     0.131390
7    -0.013168
8     0.127694
9    -0.028223
10    0.000674
Name: Partial Correlation Coeff, dtype: float64
```

```
C:\Users\nacho\anaconda3\lib\site-packages\statsmodels\graphics\tsaplots.py:348: FutureW
arning: The default method 'yw' can produce PACF values outside of the [-1,1] interval.
After 0.13, the default will change tounadjusted Yule-Walker ('ywm'). You can use this m
ethod now by setting method='ywm'.
  warnings.warn(
```

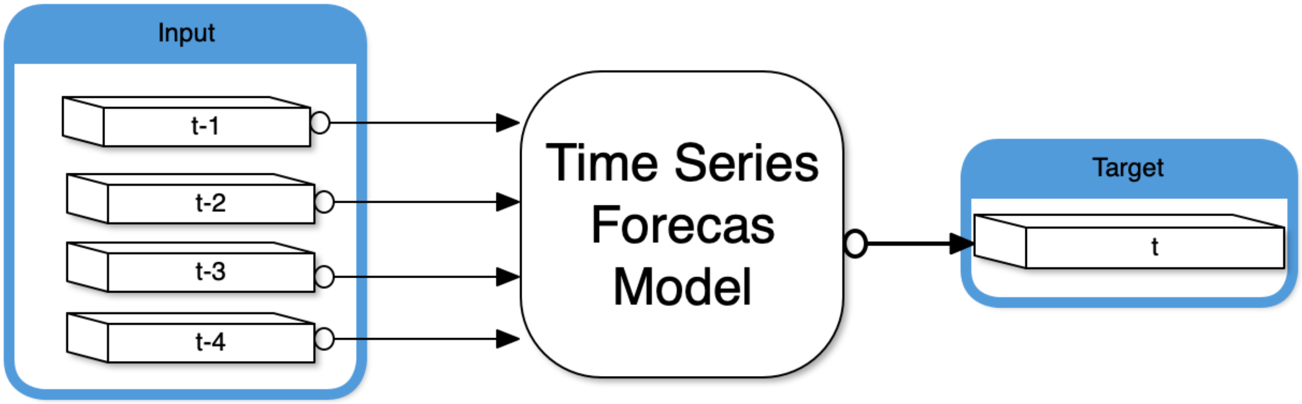




Observing the charts, it's evident that we need to utilize sales data from the preceding four weeks as input parameters.

Creation of the Dataset

Any forecast model can be shown as black-box of input - target. The target must be the data of the original time series, and the input values are given for the previous weeks.



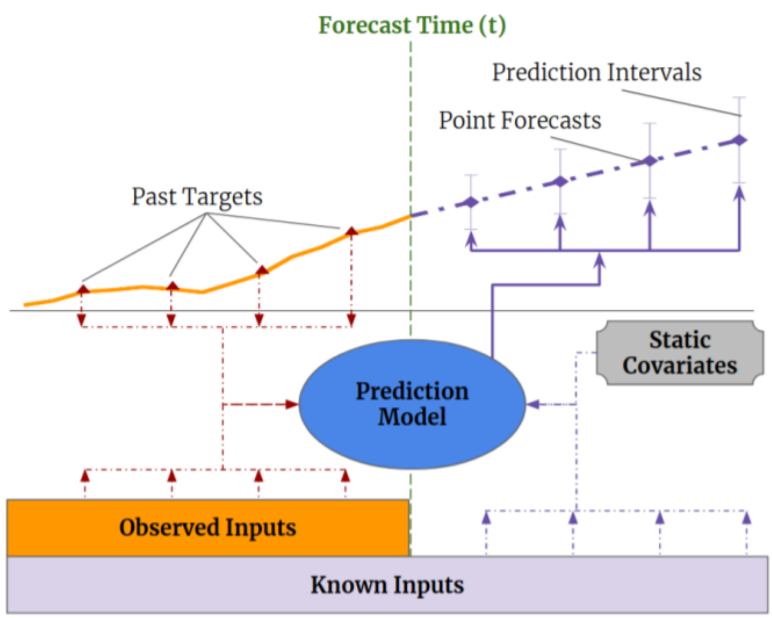


Figure 1: Illustration of multi-horizon forecasting with static covariates, past-observed and apriori-known future time-dependent inputs.

To streamline this procedure, we will construct a versatile time series transformation function that adapts to various dataset structures.

In [93]:
```
def series_to_supervised(in_data, tar_data, n_in=1, dropnan=True, target_dep=False):
    """
    Transform data into a training sample, accounting for lag
     :param in_data: Input fields
     :param tar_data: Output field (single)
     :param n_in: Lag shift
     :param dropnan: Drop empty rows
     :param target_dep: Consider lag of input field; input starts with lag 1 if True
     :return: Training sample, with the last field being the source
    """

    n_vars = in_data.shape[1]
    cols, names = list(), list()

    if target_dep:
        i_start = 1
```

```
    else:
        i_start = 0
    for i in range(i_start, n_in + 1):
        cols.append(in_data.shift(i))
        names += [('%s(t-%d)' % (in_data.columns[j], i)) for j in range(n_vars)]

    if target_dep:
        for i in range(n_in, -1, -1):
            cols.append(tar_data.shift(i))
            names += [('%s(t-%d)' % (tar_data.name, i))]
    else:
        # Combine all data columns
        cols.append(tar_data)
        names.append(tar_data.name)
    # Concatenate columns
    agg = pd.concat(cols, axis=1)
    agg.columns = names

    # Drop rows with NaN values
    if dropnan:
        agg.dropna(inplace=True)

    return agg
```

As previously discussed, the input and output fields for time series prediction are identical, with the only distinction being the shift due to the lag. Let's proceed to construct the dataset:

In [94]:
```
# Create a dataset using the series_to_supervised function
# Here, the input data is the time series 'ts', and the target data is also 'ts'
# We specify a lag of 4 for the input data
dataset = series_to_supervised(pd.DataFrame(ts), ts, n_in=4)

# Display the created dataset
dataset
```

Out[94]:

Date	Weekly_Sales(t-0)	Weekly_Sales(t-1)	Weekly_Sales(t-2)	Weekly_Sales(t-3)	Weekly_Sales(t-4)	Weekly_Sales
2010-05-03	2700.0	1350.0	1570.0	1535.0	2030.0	2700.0
2010-12-03	1760.0	2700.0	1350.0	1570.0	1535.0	1760.0
2010-03-19	2320.0	1760.0	2700.0	1350.0	1570.0	2320.0
2010-03-26	1620.0	2320.0	1760.0	2700.0	1350.0	1620.0
2010-02-04	1895.0	1620.0	2320.0	1760.0	2700.0	1895.0
...	...	...	...	...	...	...
2012-09-28	1035.0	1086.5	1141.5	850.0	765.0	1035.0
2012-05-10	1005.0	1035.0	1086.5	1141.5	850.0	1005.0
2012-12-10	1196.5	1005.0	1035.0	1086.5	1141.5	1196.5
2012-10-	1151.0	1196.5	1005.0	1035.0	1086.5	1151.0

19						
2012-10-26	595.0	1151.0	1196.5	1005.0	1035.0	595.0

139 rows × 6 columns

Observing the results, it's apparent that the initial and final columns hold identical target data. Our next step involves crafting input (**X**) and output (**Y**) datasets to facilitate the forecasting models.

In [95]:
```
# Get the column names from the dataset
col = dataset.columns

# Separate input (X) and output (Y) datasets
X, Y = dataset[col[1:-1]], dataset[col[-1]]

# Print the column names of the input dataset (X)
print("Input: ", X.columns)

# Print the name of the target dataset (Y)
print("Target:", Y.name)
```

```
Input:  Index(['Weekly_Sales(t-1)', 'Weekly_Sales(t-2)', 'Weekly_Sales(t-3)',
       'Weekly_Sales(t-4)'],
      dtype='object')
Target: Weekly_Sales
```

Data normalization

Subsequently, we need to perform data normalization. This can be achieved using the **sklearn.preprocessing.MinMaxScaler** module, which offers convenient methods for both normalization: **fit_transform()** and reverting the normalized data: **fit_transform()**.

In [96]:
```
# Create MinMaxScaler instances for both input (X) and output (Y) data
scaler_x = MinMaxScaler(feature_range=(0, 1))
scaler_y = MinMaxScaler(feature_range=(0, 1))

# Perform data normalization on input (X) and output (Y) data using the respective scale
scaled_x = scaler_x.fit_transform(X)
scaled_y = scaler_y.fit_transform(Y.values.reshape(-1, 1))
```

Next we will create the training and test DataSets using by **sklearn.model_selection.train_test_split()** in proportions 70/30. Without shuffling. It means, that test samples are lockated in the end of **X** and **Y** DataSets.

As the result we will have:

Input normalized DataSets: **X_train, X_test**

Target normalized DataSets: **y_train, y_test**

In [97]:
```
# Import the train_test_split function from sklearn.model_selection
from sklearn.model_selection import train_test_split

# Split the normalized datasets into training and test datasets (70/30 ratio), without s
X_train, X_test, y_train, y_test = train_test_split(scaled_x, scaled_y, test_size=0.3, s
```

All the data have been normalized. However, to facilitate result comparison, it's necessary to possess the original-scale data for both the training and test datasets:

In [98]:
```
# Transform the normalized target data back to the original scale using the inverse_tran
res_train = scaler_y.inverse_transform(y_train).flatten()
res_test = scaler_y.inverse_transform(y_test).flatten()
```

Target real scale DataSets: **res_train, res_test**

Linear Regression

To start off, we need to create the models. We will evaluate three different types of models: Linear Regression, a Multilayer Neural Network with Backpropagation, and a Long Short-Term Memory (LSTM) Neural Network. Let's begin by creating a **LinearRegression()** model:

In [99]:
```
# Create a Linear Regression model using LinearRegression() from sklearn.linear_model
regressor = LinearRegression()
```

Following that, the model needs to be trained on the training dataset. This can be achieved using the `fit()` function.

In [100…
```
# Fit the Linear Regression model to the training data
regressor.fit(X_train, y_train)
```

Out[100]:
```
LinearRegression()
```

Subsequently, we can evaluate its performance on the test dataset and employ it for making predictions.

In [101…
```
# Use the trained Linear Regression model to predict the target values for the test data
y_pred_test_ln = regressor.predict(X_test)

# Transform the predicted target values back to the original scale using the inverse_tra
y_pred_test_ln = scaler_y.inverse_transform(y_pred_test_ln).flatten()
```

Let's analyse accuracy of our results using **sklearn.metrics**.

In [102…
```
# Calculate and print the correlation score on the training dataset
corr_train = regressor.score(X_train, y_train)
print("Correlation train:", corr_train)

# Calculate and print the correlation score on the test dataset
corr_test = regressor.score(X_test, y_test)
print("Correlation test:", corr_test)

# Calculate and print the Mean Absolute Error (MAE) between the actual and predicted tar
mae = metrics.mean_absolute_error(y_test, y_pred_test_ln)
print('Mean Absolute Error:', mae)

# Calculate and print the Mean Squared Error (MSE) between the actual and predicted targ
mse = metrics.mean_squared_error(y_test, y_pred_test_ln)
print('Mean Squared Error:', mse)

# Calculate and print the Root Mean Squared Error (RMSE) between the actual and predicte
rmse = np.sqrt(mse)
print('Root Mean Squared Error:', rmse)
```

```
Correlation train: 0.12826507500598916
Correlation test: -0.09385443205786403
Mean Absolute Error: 1218.9282051415503
```

```
Mean Squared Error: 1496069.2012173077
Root Mean Squared Error: 1223.139076809055
```

The correlation scores, both for the training and test datasets, quantify the linear relationship between the predicted and actual target values. In this case, the correlation scores reveal that the Linear Regression model struggles to capture the underlying patterns in the data. The positive correlation score for the training dataset (0.128) indicates some degree of fit, but the negative correlation score for the test dataset (-0.094) suggests that the model's predictions are not in line with the actual values. In essence, the Linear Regression model's linear nature might not be adequately capturing the complexities present in the dataset, leading to suboptimal performance.

Considering the unsatisfactory correlation on the test dataset, it becomes evident that a more sophisticated and nonlinear model is necessary to capture the underlying relationships and trends within the data. Consequently, the exploration of alternative models, such as nonlinear neural networks, is warranted to improve forecasting accuracy and enhance the predictive capabilities of the analysis.

Back Propagation Neural Network

The contemporary approach for capturing intricate functional relationships involves the utilization of neural networks. One prominent archetype is the **multilayer neural network with back propagation**..

For this purpose, we'll employ the **keras** framework. Initially, we need to formulate a neural network model as a distinct function.

A neural network is a series of interconnected layers. The **Sequential()** function is employed to construct the network structure.

Let's forge a network comprising two hidden layers, each housing 100 neurons, employing **keras.layers.Dense()**..

To mitigate overfitting concerns, we'll integrate additional **keras.layers.Dropout()** layers.

The output layer will encompass a single neuron, as our aim is to produce a singular value at the output.

Before we proceed with fitting and prediction, the model needs to be compiled using **keras.Model.compile()**.

In [103…

```
def BP_model(X):
    """
    Multilayer neural network with back propagation.
    :param X: Input DataSet
    :return: Keras neural network model
    """
    # create model
    model = Sequential()
    model.add(Dense(100, input_dim=X.shape[1], kernel_initializer='normal', activation='
    model.add(Dropout(0.2))
    model.add(Dense(50, kernel_initializer='normal', activation='relu'))
    model.add(Dropout(0.2))
    model.add(Dense(1, kernel_initializer='normal'))
    # Compile model
    model.compile(loss='mean_squared_error', optimizer='adam')
    return model
```

After constructing the model function, the next step involves directly creating a neural network and specifying the learning parameters using **keras.wrappers.scikit_learn.KerasRegressor()**. Additionally, we need to define the number of training **epochs and batch size**.

In [104…

```
# Define the number of epochs for training
epochs = 1000

# Define the batch size as 10% of the training data size
batch_size = int(y_train.shape[0] * 0.1)

# Create a KerasRegressor with the BP_model function, specifying training data, epochs,
estimator = KerasRegressor(build_fn=BP_model, X=X_train, epochs=epochs, batch_size=batch
```

Now, let's train our model for **1000** epochs. It should be noted, that fitting process is very slow. To avoid overfitting and decrease time of fitting we will use **EarlyStopping()** function, which will control value of loss function. This function will halt the fitting process if the loss function stops decreasing for a continuous span of 10 iterations. Subsequently, all weight parameters will be restored to their state from the 10th iteration prior.

In [105…

```
# Define EarlyStopping callback with specified parameters
es = EarlyStopping(monitor='val_loss', mode='auto', patience=10, verbose=1, restore_best

# Fit the estimator model on the training data with validation data and EarlyStopping ca
history = estimator.fit(X_train, y_train, validation_data=(X_test, y_test), callbacks=[e
```

```
Epoch 1/1000
C:\Users\nacho\anaconda3\lib\site-packages\scikeras\wrappers.py:915: UserWarning: ``buil
d_fn`` will be renamed to ``model`` in a future release, at which point use of ``build_f
n`` will raise an Error instead.
  X, y = self._initialize(X, y)
11/11 [==============================] - 1s 11ms/step - loss: 0.2423 - val_loss: 0.1414
Epoch 2/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.1766 - val_loss: 0.0810
Epoch 3/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0813 - val_loss: 0.0211
Epoch 4/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0324 - val_loss: 0.0285
Epoch 5/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0258 - val_loss: 0.0183
Epoch 6/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0284 - val_loss: 0.0169
Epoch 7/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0279 - val_loss: 0.0169
Epoch 8/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0261 - val_loss: 0.0181
Epoch 9/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0240 - val_loss: 0.0179
Epoch 10/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0261 - val_loss: 0.0172
Epoch 11/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0236 - val_loss: 0.0186
Epoch 12/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0248 - val_loss: 0.0171
Epoch 13/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0226 - val_loss: 0.0178
Epoch 14/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0264 - val_loss: 0.0168
Epoch 15/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0220 - val_loss: 0.0173
Epoch 16/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0227 - val_loss: 0.0180
Epoch 17/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0225 - val_loss: 0.0178
Epoch 18/1000
```

```
11/11 [==============================] - 0s 4ms/step - loss: 0.0244 - val_loss: 0.0166
Epoch 19/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0186 - val_loss: 0.0177
Epoch 20/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0246 - val_loss: 0.0177
Epoch 21/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0247 - val_loss: 0.0163
Epoch 22/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0206 - val_loss: 0.0183
Epoch 23/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0254 - val_loss: 0.0176
Epoch 24/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0233 - val_loss: 0.0177
Epoch 25/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0203 - val_loss: 0.0173
Epoch 26/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0232 - val_loss: 0.0165
Epoch 27/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0239 - val_loss: 0.0191
Epoch 28/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0265 - val_loss: 0.0163
Epoch 29/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0247 - val_loss: 0.0179
Epoch 30/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0251 - val_loss: 0.0186
Epoch 31/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0229 - val_loss: 0.0172
Epoch 32/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0228 - val_loss: 0.0165
Epoch 33/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0239 - val_loss: 0.0187
Epoch 34/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0222 - val_loss: 0.0165
Epoch 35/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0234 - val_loss: 0.0178
Epoch 36/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0236 - val_loss: 0.0169
Epoch 37/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0207 - val_loss: 0.0181
Epoch 38/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0257 - val_loss: 0.0176
Epoch 39/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0222 - val_loss: 0.0177
Epoch 40/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0226 - val_loss: 0.0166
Epoch 41/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0237 - val_loss: 0.0174
Epoch 42/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0229 - val_loss: 0.0185
Epoch 43/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0204 - val_loss: 0.0159
Epoch 44/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0215 - val_loss: 0.0178
Epoch 45/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0200 - val_loss: 0.0185
Epoch 46/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0264 - val_loss: 0.0165
Epoch 47/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0205 - val_loss: 0.0165
Epoch 48/1000
11/11 [==============================] - 0s 21ms/step - loss: 0.0234 - val_loss: 0.0167
Epoch 49/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0231 - val_loss: 0.0167
Epoch 50/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0214 - val_loss: 0.0179
Epoch 51/1000
```

```
11/11 [==============================] - 0s 4ms/step - loss: 0.0230 - val_loss: 0.0177
Epoch 52/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0226 - val_loss: 0.0165
Epoch 53/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0223 - val_loss: 0.0171
Epoch 54/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0220 - val_loss: 0.0160
Epoch 55/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0220 - val_loss: 0.0195
Epoch 56/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0237 - val_loss: 0.0161
Epoch 57/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0237 - val_loss: 0.0184
Epoch 58/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0239 - val_loss: 0.0164
Epoch 59/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0229 - val_loss: 0.0182
Epoch 60/1000
11/11 [==============================] - 0s 3ms/step - loss: 0.0217 - val_loss: 0.0166
Epoch 61/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0231 - val_loss: 0.0170
Epoch 62/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0217 - val_loss: 0.0181
Epoch 63/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0228 - val_loss: 0.0167
Epoch 64/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0236 - val_loss: 0.0177
Epoch 65/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0247 - val_loss: 0.0164
Epoch 66/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0248 - val_loss: 0.0200
Epoch 67/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0243 - val_loss: 0.0161
Epoch 68/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0229 - val_loss: 0.0194
Epoch 69/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0235 - val_loss: 0.0157
Epoch 70/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0232 - val_loss: 0.0185
Epoch 71/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0227 - val_loss: 0.0173
Epoch 72/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0216 - val_loss: 0.0179
Epoch 73/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0236 - val_loss: 0.0169
Epoch 74/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0207 - val_loss: 0.0175
Epoch 75/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0229 - val_loss: 0.0174
Epoch 76/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0229 - val_loss: 0.0170
Epoch 77/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0211 - val_loss: 0.0169
Epoch 78/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0241 - val_loss: 0.0166
Epoch 79/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0220 - val_loss: 0.0177
Epoch 80/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0242 - val_loss: 0.0168
Epoch 81/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0235 - val_loss: 0.0176
Epoch 82/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0252 - val_loss: 0.0170
Epoch 83/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0209 - val_loss: 0.0177
Epoch 84/1000
```

```
11/11 [==============================] - 0s 4ms/step - loss: 0.0220 - val_loss: 0.0173
Epoch 85/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0224 - val_loss: 0.0177
Epoch 86/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0202 - val_loss: 0.0179
Epoch 87/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0225 - val_loss: 0.0172
Epoch 88/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0209 - val_loss: 0.0171
Epoch 89/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0228 - val_loss: 0.0185
Epoch 90/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0222 - val_loss: 0.0175
Epoch 91/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0226 - val_loss: 0.0180
Epoch 92/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0199 - val_loss: 0.0175
Epoch 93/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0207 - val_loss: 0.0182
Epoch 94/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0228 - val_loss: 0.0176
Epoch 95/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0202 - val_loss: 0.0167
Epoch 96/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0258 - val_loss: 0.0177
Epoch 97/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0239 - val_loss: 0.0174
Epoch 98/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0206 - val_loss: 0.0175
Epoch 99/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0179 - val_loss: 0.0179
Epoch 100/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0204 - val_loss: 0.0167
Epoch 101/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0215 - val_loss: 0.0193
Epoch 102/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0211 - val_loss: 0.0170
Epoch 103/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0219 - val_loss: 0.0173
Epoch 104/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0201 - val_loss: 0.0172
Epoch 105/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0218 - val_loss: 0.0184
Epoch 106/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0199 - val_loss: 0.0191
Epoch 107/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0253 - val_loss: 0.0162
Epoch 108/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0221 - val_loss: 0.0188
Epoch 109/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0219 - val_loss: 0.0157
Epoch 110/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0215 - val_loss: 0.0186
Epoch 111/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0211 - val_loss: 0.0175
Epoch 112/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0216 - val_loss: 0.0177
Epoch 113/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0205 - val_loss: 0.0182
Epoch 114/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0228 - val_loss: 0.0177
Epoch 115/1000
11/11 [==============================] - 0s 5ms/step - loss: 0.0221 - val_loss: 0.0177
Epoch 116/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0210 - val_loss: 0.0189
Epoch 117/1000
```

```
11/11 [==============================] - 0s 4ms/step - loss: 0.0220 - val_loss: 0.0166
Epoch 118/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0202 - val_loss: 0.0182
Epoch 119/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0208 - val_loss: 0.0175
Epoch 120/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0229 - val_loss: 0.0177
Epoch 121/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0255 - val_loss: 0.0165
Epoch 122/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0234 - val_loss: 0.0167
Epoch 123/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0240 - val_loss: 0.0185
Epoch 124/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0236 - val_loss: 0.0168
Epoch 125/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0207 - val_loss: 0.0180
Epoch 126/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0217 - val_loss: 0.0173
Epoch 127/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0236 - val_loss: 0.0171
Epoch 128/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0222 - val_loss: 0.0166
Epoch 129/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0203 - val_loss: 0.0194
Epoch 130/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0213 - val_loss: 0.0179
Epoch 131/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0219 - val_loss: 0.0174
Epoch 132/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0210 - val_loss: 0.0177
Epoch 133/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0222 - val_loss: 0.0174
Epoch 134/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0211 - val_loss: 0.0179
Epoch 135/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0204 - val_loss: 0.0172
Epoch 136/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0211 - val_loss: 0.0174
Epoch 137/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0209 - val_loss: 0.0180
Epoch 138/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0218 - val_loss: 0.0184
Epoch 139/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0226 - val_loss: 0.0172
Epoch 140/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0220 - val_loss: 0.0183
Epoch 141/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0205 - val_loss: 0.0188
Epoch 142/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0210 - val_loss: 0.0169
Epoch 143/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0222 - val_loss: 0.0183
Epoch 144/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0216 - val_loss: 0.0178
Epoch 145/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0203 - val_loss: 0.0175
Epoch 146/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0202 - val_loss: 0.0173
Epoch 147/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0209 - val_loss: 0.0167
Epoch 148/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0209 - val_loss: 0.0169
Epoch 149/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0202 - val_loss: 0.0188
Epoch 150/1000
```

```
11/11 [==============================] - 0s 4ms/step - loss: 0.0221 - val_loss: 0.0170
Epoch 151/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0197 - val_loss: 0.0169
Epoch 152/1000
11/11 [==============================] - 0s 3ms/step - loss: 0.0205 - val_loss: 0.0198
Epoch 153/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0215 - val_loss: 0.0169
Epoch 154/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0202 - val_loss: 0.0165
Epoch 155/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0227 - val_loss: 0.0181
Epoch 156/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0203 - val_loss: 0.0173
Epoch 157/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0219 - val_loss: 0.0168
Epoch 158/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0217 - val_loss: 0.0185
Epoch 159/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0190 - val_loss: 0.0188
Epoch 160/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0218 - val_loss: 0.0167
Epoch 161/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0229 - val_loss: 0.0184
Epoch 162/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0185 - val_loss: 0.0180
Epoch 163/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0220 - val_loss: 0.0172
Epoch 164/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0200 - val_loss: 0.0194
Epoch 165/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0207 - val_loss: 0.0165
Epoch 166/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0190 - val_loss: 0.0184
Epoch 167/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0217 - val_loss: 0.0176
Epoch 168/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0219 - val_loss: 0.0173
Epoch 169/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0200 - val_loss: 0.0171
Epoch 170/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0208 - val_loss: 0.0179
Epoch 171/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0206 - val_loss: 0.0180
Epoch 172/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0213 - val_loss: 0.0169
Epoch 173/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0205 - val_loss: 0.0185
Epoch 174/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0213 - val_loss: 0.0179
Epoch 175/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0201 - val_loss: 0.0174
Epoch 176/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0212 - val_loss: 0.0175
Epoch 177/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0206 - val_loss: 0.0174
Epoch 178/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0222 - val_loss: 0.0186
Epoch 179/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0225 - val_loss: 0.0169
Epoch 180/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0225 - val_loss: 0.0171
Epoch 181/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0195 - val_loss: 0.0201
Epoch 182/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0199 - val_loss: 0.0179
Epoch 183/1000
```

```
11/11 [==============================] - 0s 4ms/step - loss: 0.0208 - val_loss: 0.0171
Epoch 184/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0199 - val_loss: 0.0194
Epoch 185/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0216 - val_loss: 0.0180
Epoch 186/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0196 - val_loss: 0.0173
Epoch 187/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0199 - val_loss: 0.0186
Epoch 188/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0215 - val_loss: 0.0177
Epoch 189/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0200 - val_loss: 0.0172
Epoch 190/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0207 - val_loss: 0.0199
Epoch 191/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0227 - val_loss: 0.0167
Epoch 192/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0193 - val_loss: 0.0174
Epoch 193/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0227 - val_loss: 0.0187
Epoch 194/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0204 - val_loss: 0.0172
Epoch 195/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0223 - val_loss: 0.0174
Epoch 196/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0205 - val_loss: 0.0169
Epoch 197/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0190 - val_loss: 0.0203
Epoch 198/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0195 - val_loss: 0.0176
Epoch 199/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0215 - val_loss: 0.0165
Epoch 200/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0217 - val_loss: 0.0180
Epoch 201/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0187 - val_loss: 0.0190
Epoch 202/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0203 - val_loss: 0.0175
Epoch 203/1000
11/11 [==============================] - 0s 5ms/step - loss: 0.0190 - val_loss: 0.0186
Epoch 204/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0209 - val_loss: 0.0173
Epoch 205/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0221 - val_loss: 0.0184
Epoch 206/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0210 - val_loss: 0.0176
Epoch 207/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0197 - val_loss: 0.0184
Epoch 208/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0198 - val_loss: 0.0182
Epoch 209/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0189 - val_loss: 0.0189
Epoch 210/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0213 - val_loss: 0.0169
Epoch 211/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0203 - val_loss: 0.0179
Epoch 212/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0211 - val_loss: 0.0186
Epoch 213/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0236 - val_loss: 0.0178
Epoch 214/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0219 - val_loss: 0.0179
Epoch 215/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0201 - val_loss: 0.0177
Epoch 216/1000
```

```
11/11 [==============================] - 0s 4ms/step - loss: 0.0203 - val_loss: 0.0183
Epoch 217/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0198 - val_loss: 0.0182
Epoch 218/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0206 - val_loss: 0.0167
Epoch 219/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0219 - val_loss: 0.0177
Epoch 220/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0221 - val_loss: 0.0198
Epoch 221/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0207 - val_loss: 0.0172
Epoch 222/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0207 - val_loss: 0.0188
Epoch 223/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0201 - val_loss: 0.0180
Epoch 224/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0191 - val_loss: 0.0189
Epoch 225/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0203 - val_loss: 0.0169
Epoch 226/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0200 - val_loss: 0.0196
Epoch 227/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0200 - val_loss: 0.0191
Epoch 228/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0194 - val_loss: 0.0183
Epoch 229/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0195 - val_loss: 0.0179
Epoch 230/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0210 - val_loss: 0.0175
Epoch 231/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0212 - val_loss: 0.0181
Epoch 232/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0201 - val_loss: 0.0186
Epoch 233/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0198 - val_loss: 0.0177
Epoch 234/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0203 - val_loss: 0.0192
Epoch 235/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0191 - val_loss: 0.0172
Epoch 236/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0216 - val_loss: 0.0177
Epoch 237/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0198 - val_loss: 0.0188
Epoch 238/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0198 - val_loss: 0.0174
Epoch 239/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0196 - val_loss: 0.0172
Epoch 240/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0204 - val_loss: 0.0181
Epoch 241/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0184 - val_loss: 0.0175
Epoch 242/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0199 - val_loss: 0.0179
Epoch 243/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0207 - val_loss: 0.0185
Epoch 244/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0202 - val_loss: 0.0190
Epoch 245/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0198 - val_loss: 0.0175
Epoch 246/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0193 - val_loss: 0.0189
Epoch 247/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0201 - val_loss: 0.0175
Epoch 248/1000
11/11 [==============================] - 0s 3ms/step - loss: 0.0198 - val_loss: 0.0184
Epoch 249/1000
```

```
11/11 [==============================] - 0s 3ms/step - loss: 0.0209 - val_loss: 0.0176
Epoch 250/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0206 - val_loss: 0.0183
Epoch 251/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0209 - val_loss: 0.0180
Epoch 252/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0189 - val_loss: 0.0182
Epoch 253/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0223 - val_loss: 0.0185
Epoch 254/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0190 - val_loss: 0.0175
Epoch 255/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0206 - val_loss: 0.0189
Epoch 256/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0175 - val_loss: 0.0176
Epoch 257/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0201 - val_loss: 0.0195
Epoch 258/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0208 - val_loss: 0.0176
Epoch 259/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0199 - val_loss: 0.0184
Epoch 260/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0187 - val_loss: 0.0184
Epoch 261/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0209 - val_loss: 0.0165
Epoch 262/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0183 - val_loss: 0.0188
Epoch 263/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0190 - val_loss: 0.0191
Epoch 264/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0215 - val_loss: 0.0173
Epoch 265/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0203 - val_loss: 0.0179
Epoch 266/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0196 - val_loss: 0.0192
Epoch 267/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0206 - val_loss: 0.0170
Epoch 268/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0199 - val_loss: 0.0174
Epoch 269/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0205 - val_loss: 0.0174
Epoch 270/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0188 - val_loss: 0.0206
Epoch 271/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0216 - val_loss: 0.0169
Epoch 272/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0208 - val_loss: 0.0172
Epoch 273/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0198 - val_loss: 0.0210
Epoch 274/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0194 - val_loss: 0.0184
Epoch 275/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0184 - val_loss: 0.0182
Epoch 276/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0196 - val_loss: 0.0184
Epoch 277/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0202 - val_loss: 0.0182
Epoch 278/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0205 - val_loss: 0.0187
Epoch 279/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0198 - val_loss: 0.0200
Epoch 280/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0194 - val_loss: 0.0176
Epoch 281/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0218 - val_loss: 0.0172
Epoch 282/1000
```

```
11/11 [==============================] - 0s 4ms/step - loss: 0.0196 - val_loss: 0.0194
Epoch 283/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0202 - val_loss: 0.0186
Epoch 284/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0189 - val_loss: 0.0174
Epoch 285/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0200 - val_loss: 0.0185
Epoch 286/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0193 - val_loss: 0.0182
Epoch 287/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0201 - val_loss: 0.0186
Epoch 288/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0193 - val_loss: 0.0186
Epoch 289/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0187 - val_loss: 0.0183
Epoch 290/1000
11/11 [==============================] - 0s 5ms/step - loss: 0.0196 - val_loss: 0.0187
Epoch 291/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0204 - val_loss: 0.0186
Epoch 292/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0197 - val_loss: 0.0171
Epoch 293/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0199 - val_loss: 0.0187
Epoch 294/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0205 - val_loss: 0.0183
Epoch 295/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0206 - val_loss: 0.0174
Epoch 296/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0184 - val_loss: 0.0193
Epoch 297/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0204 - val_loss: 0.0175
Epoch 298/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0194 - val_loss: 0.0188
Epoch 299/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0210 - val_loss: 0.0187
Epoch 300/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0206 - val_loss: 0.0188
Epoch 301/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0192 - val_loss: 0.0194
Epoch 302/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0190 - val_loss: 0.0189
Epoch 303/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0222 - val_loss: 0.0171
Epoch 304/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0197 - val_loss: 0.0198
Epoch 305/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0178 - val_loss: 0.0181
Epoch 306/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0189 - val_loss: 0.0182
Epoch 307/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0201 - val_loss: 0.0180
Epoch 308/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0192 - val_loss: 0.0174
Epoch 309/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0198 - val_loss: 0.0187
Epoch 310/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0198 - val_loss: 0.0181
Epoch 311/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0188 - val_loss: 0.0185
Epoch 312/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0187 - val_loss: 0.0187
Epoch 313/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0188 - val_loss: 0.0179
Epoch 314/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0203 - val_loss: 0.0189
Epoch 315/1000
```

```
11/11 [==============================] - 0s 4ms/step - loss: 0.0184 - val_loss: 0.0191
Epoch 316/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0206 - val_loss: 0.0178
Epoch 317/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0187 - val_loss: 0.0188
Epoch 318/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0176 - val_loss: 0.0207
Epoch 319/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0190 - val_loss: 0.0178
Epoch 320/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0195 - val_loss: 0.0174
Epoch 321/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0195 - val_loss: 0.0196
Epoch 322/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0192 - val_loss: 0.0179
Epoch 323/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0186 - val_loss: 0.0180
Epoch 324/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0199 - val_loss: 0.0181
Epoch 325/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0205 - val_loss: 0.0202
Epoch 326/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0191 - val_loss: 0.0179
Epoch 327/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0172 - val_loss: 0.0191
Epoch 328/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0194 - val_loss: 0.0187
Epoch 329/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0185 - val_loss: 0.0194
Epoch 330/1000
11/11 [==============================] - 0s 3ms/step - loss: 0.0201 - val_loss: 0.0187
Epoch 331/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0183 - val_loss: 0.0187
Epoch 332/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0183 - val_loss: 0.0191
Epoch 333/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0179 - val_loss: 0.0188
Epoch 334/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0182 - val_loss: 0.0184
Epoch 335/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0200 - val_loss: 0.0183
Epoch 336/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0212 - val_loss: 0.0194
Epoch 337/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0185 - val_loss: 0.0182
Epoch 338/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0197 - val_loss: 0.0184
Epoch 339/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0181 - val_loss: 0.0185
Epoch 340/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0198 - val_loss: 0.0195
Epoch 341/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0210 - val_loss: 0.0175
Epoch 342/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0164 - val_loss: 0.0202
Epoch 343/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0193 - val_loss: 0.0183
Epoch 344/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0180 - val_loss: 0.0181
Epoch 345/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0195 - val_loss: 0.0180
Epoch 346/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0193 - val_loss: 0.0186
Epoch 347/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0207 - val_loss: 0.0185
Epoch 348/1000
```

```
11/11 [==============================] - 0s 4ms/step - loss: 0.0193 - val_loss: 0.0187
Epoch 349/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0178 - val_loss: 0.0199
Epoch 350/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0181 - val_loss: 0.0187
Epoch 351/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0203 - val_loss: 0.0180
Epoch 352/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0183 - val_loss: 0.0199
Epoch 353/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0192 - val_loss: 0.0194
Epoch 354/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0196 - val_loss: 0.0180
Epoch 355/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0182 - val_loss: 0.0185
Epoch 356/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0205 - val_loss: 0.0198
Epoch 357/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0199 - val_loss: 0.0177
Epoch 358/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0187 - val_loss: 0.0186
Epoch 359/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0203 - val_loss: 0.0176
Epoch 360/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0190 - val_loss: 0.0191
Epoch 361/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0181 - val_loss: 0.0204
Epoch 362/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0190 - val_loss: 0.0174
Epoch 363/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0180 - val_loss: 0.0183
Epoch 364/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0190 - val_loss: 0.0176
Epoch 365/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0184 - val_loss: 0.0200
Epoch 366/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0200 - val_loss: 0.0188
Epoch 367/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0192 - val_loss: 0.0200
Epoch 368/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0201 - val_loss: 0.0184
Epoch 369/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0201 - val_loss: 0.0197
Epoch 370/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0187 - val_loss: 0.0181
Epoch 371/1000
11/11 [==============================] - 0s 5ms/step - loss: 0.0183 - val_loss: 0.0184
Epoch 372/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0198 - val_loss: 0.0192
Epoch 373/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0188 - val_loss: 0.0184
Epoch 374/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0179 - val_loss: 0.0185
Epoch 375/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0192 - val_loss: 0.0192
Epoch 376/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0178 - val_loss: 0.0192
Epoch 377/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0178 - val_loss: 0.0184
Epoch 378/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0188 - val_loss: 0.0185
Epoch 379/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0196 - val_loss: 0.0180
Epoch 380/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0184 - val_loss: 0.0188
Epoch 381/1000
```

```
11/11 [==============================] - 0s 4ms/step - loss: 0.0182 - val_loss: 0.0185
Epoch 382/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0181 - val_loss: 0.0197
Epoch 383/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0194 - val_loss: 0.0182
Epoch 384/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0187 - val_loss: 0.0187
Epoch 385/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0200 - val_loss: 0.0183
Epoch 386/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0201 - val_loss: 0.0180
Epoch 387/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0169 - val_loss: 0.0198
Epoch 388/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0189 - val_loss: 0.0181
Epoch 389/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0201 - val_loss: 0.0191
Epoch 390/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0192 - val_loss: 0.0205
Epoch 391/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0177 - val_loss: 0.0184
Epoch 392/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0184 - val_loss: 0.0186
Epoch 393/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0176 - val_loss: 0.0193
Epoch 394/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0175 - val_loss: 0.0199
Epoch 395/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0204 - val_loss: 0.0181
Epoch 396/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0190 - val_loss: 0.0195
Epoch 397/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0195 - val_loss: 0.0187
Epoch 398/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0177 - val_loss: 0.0181
Epoch 399/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0185 - val_loss: 0.0197
Epoch 400/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0185 - val_loss: 0.0194
Epoch 401/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0179 - val_loss: 0.0183
Epoch 402/1000
11/11 [==============================] - 0s 3ms/step - loss: 0.0188 - val_loss: 0.0190
Epoch 403/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0178 - val_loss: 0.0182
Epoch 404/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0179 - val_loss: 0.0192
Epoch 405/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0200 - val_loss: 0.0196
Epoch 406/1000
11/11 [==============================] - 0s 3ms/step - loss: 0.0197 - val_loss: 0.0178
Epoch 407/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0197 - val_loss: 0.0201
Epoch 408/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0186 - val_loss: 0.0182
Epoch 409/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0183 - val_loss: 0.0197
Epoch 410/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0198 - val_loss: 0.0190
Epoch 411/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0179 - val_loss: 0.0194
Epoch 412/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0175 - val_loss: 0.0195
Epoch 413/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0195 - val_loss: 0.0194
Epoch 414/1000
```

```
11/11 [==============================] - 0s 4ms/step - loss: 0.0178 - val_loss: 0.0195
Epoch 415/1000
11/11 [==============================] - 0s 3ms/step - loss: 0.0186 - val_loss: 0.0190
Epoch 416/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0176 - val_loss: 0.0201
Epoch 417/1000
11/11 [==============================] - 0s 3ms/step - loss: 0.0172 - val_loss: 0.0193
Epoch 418/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0178 - val_loss: 0.0191
Epoch 419/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0195 - val_loss: 0.0195
Epoch 420/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0194 - val_loss: 0.0186
Epoch 421/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0189 - val_loss: 0.0200
Epoch 422/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0178 - val_loss: 0.0201
Epoch 423/1000
11/11 [==============================] - 0s 3ms/step - loss: 0.0184 - val_loss: 0.0176
Epoch 424/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0190 - val_loss: 0.0184
Epoch 425/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0177 - val_loss: 0.0209
Epoch 426/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0185 - val_loss: 0.0198
Epoch 427/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0190 - val_loss: 0.0181
Epoch 428/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0175 - val_loss: 0.0194
Epoch 429/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0184 - val_loss: 0.0194
Epoch 430/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0189 - val_loss: 0.0184
Epoch 431/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0193 - val_loss: 0.0212
Epoch 432/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0184 - val_loss: 0.0182
Epoch 433/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0178 - val_loss: 0.0205
Epoch 434/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0175 - val_loss: 0.0205
Epoch 435/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0181 - val_loss: 0.0195
Epoch 436/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0184 - val_loss: 0.0191
Epoch 437/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0187 - val_loss: 0.0183
Epoch 438/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0184 - val_loss: 0.0203
Epoch 439/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0174 - val_loss: 0.0213
Epoch 440/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0189 - val_loss: 0.0195
Epoch 441/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0181 - val_loss: 0.0186
Epoch 442/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0174 - val_loss: 0.0212
Epoch 443/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0175 - val_loss: 0.0188
Epoch 444/1000
11/11 [==============================] - 0s 3ms/step - loss: 0.0187 - val_loss: 0.0185
Epoch 445/1000
11/11 [==============================] - 0s 3ms/step - loss: 0.0176 - val_loss: 0.0205
Epoch 446/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0190 - val_loss: 0.0191
Epoch 447/1000
```

```
11/11 [==============================] - 0s 4ms/step - loss: 0.0190 - val_loss: 0.0187
Epoch 448/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0176 - val_loss: 0.0194
Epoch 449/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0182 - val_loss: 0.0186
Epoch 450/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0186 - val_loss: 0.0184
Epoch 451/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0179 - val_loss: 0.0202
Epoch 452/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0182 - val_loss: 0.0194
Epoch 453/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0180 - val_loss: 0.0197
Epoch 454/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0177 - val_loss: 0.0187
Epoch 455/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0186 - val_loss: 0.0194
Epoch 456/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0178 - val_loss: 0.0190
Epoch 457/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0171 - val_loss: 0.0197
Epoch 458/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0186 - val_loss: 0.0189
Epoch 459/1000
11/11 [==============================] - 0s 5ms/step - loss: 0.0182 - val_loss: 0.0186
Epoch 460/1000
11/11 [==============================] - 0s 3ms/step - loss: 0.0195 - val_loss: 0.0181
Epoch 461/1000
11/11 [==============================] - 0s 3ms/step - loss: 0.0192 - val_loss: 0.0187
Epoch 462/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0173 - val_loss: 0.0204
Epoch 463/1000
11/11 [==============================] - 0s 3ms/step - loss: 0.0176 - val_loss: 0.0188
Epoch 464/1000
11/11 [==============================] - 0s 3ms/step - loss: 0.0177 - val_loss: 0.0202
Epoch 465/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0181 - val_loss: 0.0198
Epoch 466/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0183 - val_loss: 0.0195
Epoch 467/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0175 - val_loss: 0.0202
Epoch 468/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0197 - val_loss: 0.0187
Epoch 469/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0184 - val_loss: 0.0207
Epoch 470/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0197 - val_loss: 0.0186
Epoch 471/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0181 - val_loss: 0.0193
Epoch 472/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0183 - val_loss: 0.0194
Epoch 473/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0190 - val_loss: 0.0199
Epoch 474/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0190 - val_loss: 0.0191
Epoch 475/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0171 - val_loss: 0.0188
Epoch 476/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0177 - val_loss: 0.0200
Epoch 477/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0179 - val_loss: 0.0195
Epoch 478/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0185 - val_loss: 0.0193
Epoch 479/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0188 - val_loss: 0.0192
Epoch 480/1000
```

```
11/11 [==============================] - 0s 4ms/step - loss: 0.0193 - val_loss: 0.0206
Epoch 481/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0189 - val_loss: 0.0212
Epoch 482/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0175 - val_loss: 0.0190
Epoch 483/1000
11/11 [==============================] - 0s 3ms/step - loss: 0.0183 - val_loss: 0.0199
Epoch 484/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0189 - val_loss: 0.0211
Epoch 485/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0192 - val_loss: 0.0198
Epoch 486/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0188 - val_loss: 0.0198
Epoch 487/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0180 - val_loss: 0.0207
Epoch 488/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0187 - val_loss: 0.0197
Epoch 489/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0175 - val_loss: 0.0194
Epoch 490/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0179 - val_loss: 0.0205
Epoch 491/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0170 - val_loss: 0.0202
Epoch 492/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0180 - val_loss: 0.0201
Epoch 493/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0184 - val_loss: 0.0195
Epoch 494/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0173 - val_loss: 0.0196
Epoch 495/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0178 - val_loss: 0.0198
Epoch 496/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0169 - val_loss: 0.0201
Epoch 497/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0175 - val_loss: 0.0196
Epoch 498/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0179 - val_loss: 0.0186
Epoch 499/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0162 - val_loss: 0.0210
Epoch 500/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0170 - val_loss: 0.0198
Epoch 501/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0185 - val_loss: 0.0202
Epoch 502/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0189 - val_loss: 0.0203
Epoch 503/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0181 - val_loss: 0.0211
Epoch 504/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0174 - val_loss: 0.0199
Epoch 505/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0167 - val_loss: 0.0204
Epoch 506/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0177 - val_loss: 0.0200
Epoch 507/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0187 - val_loss: 0.0209
Epoch 508/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0166 - val_loss: 0.0187
Epoch 509/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0190 - val_loss: 0.0200
Epoch 510/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0174 - val_loss: 0.0211
Epoch 511/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0184 - val_loss: 0.0210
Epoch 512/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0196 - val_loss: 0.0194
Epoch 513/1000
```

```
11/11 [==============================] - 0s 4ms/step - loss: 0.0185 - val_loss: 0.0202
Epoch 514/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0171 - val_loss: 0.0205
Epoch 515/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0167 - val_loss: 0.0199
Epoch 516/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0175 - val_loss: 0.0200
Epoch 517/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0185 - val_loss: 0.0194
Epoch 518/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0177 - val_loss: 0.0210
Epoch 519/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0172 - val_loss: 0.0211
Epoch 520/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0184 - val_loss: 0.0203
Epoch 521/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0166 - val_loss: 0.0204
Epoch 522/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0200 - val_loss: 0.0195
Epoch 523/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0185 - val_loss: 0.0210
Epoch 524/1000
11/11 [==============================] - 0s 3ms/step - loss: 0.0168 - val_loss: 0.0201
Epoch 525/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0189 - val_loss: 0.0201
Epoch 526/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0191 - val_loss: 0.0206
Epoch 527/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0194 - val_loss: 0.0205
Epoch 528/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0166 - val_loss: 0.0209
Epoch 529/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0184 - val_loss: 0.0203
Epoch 530/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0180 - val_loss: 0.0194
Epoch 531/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0181 - val_loss: 0.0205
Epoch 532/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0170 - val_loss: 0.0193
Epoch 533/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0167 - val_loss: 0.0198
Epoch 534/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0166 - val_loss: 0.0196
Epoch 535/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0190 - val_loss: 0.0206
Epoch 536/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0183 - val_loss: 0.0191
Epoch 537/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0171 - val_loss: 0.0214
Epoch 538/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0178 - val_loss: 0.0204
Epoch 539/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0174 - val_loss: 0.0210
Epoch 540/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0188 - val_loss: 0.0210
Epoch 541/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0183 - val_loss: 0.0184
Epoch 542/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0179 - val_loss: 0.0207
Epoch 543/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0183 - val_loss: 0.0205
Epoch 544/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0195 - val_loss: 0.0207
Epoch 545/1000
11/11 [==============================] - 0s 5ms/step - loss: 0.0180 - val_loss: 0.0211
Epoch 546/1000
```

```
11/11 [==============================] - 0s 4ms/step - loss: 0.0168 - val_loss: 0.0200
Epoch 547/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0182 - val_loss: 0.0204
Epoch 548/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0171 - val_loss: 0.0204
Epoch 549/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0177 - val_loss: 0.0200
Epoch 550/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0178 - val_loss: 0.0211
Epoch 551/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0176 - val_loss: 0.0197
Epoch 552/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0172 - val_loss: 0.0199
Epoch 553/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0176 - val_loss: 0.0201
Epoch 554/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0172 - val_loss: 0.0203
Epoch 555/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0182 - val_loss: 0.0211
Epoch 556/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0175 - val_loss: 0.0197
Epoch 557/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0188 - val_loss: 0.0204
Epoch 558/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0182 - val_loss: 0.0189
Epoch 559/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0174 - val_loss: 0.0200
Epoch 560/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0165 - val_loss: 0.0215
Epoch 561/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0190 - val_loss: 0.0202
Epoch 562/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0174 - val_loss: 0.0207
Epoch 563/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0174 - val_loss: 0.0203
Epoch 564/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0176 - val_loss: 0.0203
Epoch 565/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0174 - val_loss: 0.0209
Epoch 566/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0175 - val_loss: 0.0191
Epoch 567/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0178 - val_loss: 0.0218
Epoch 568/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0186 - val_loss: 0.0217
Epoch 569/1000
11/11 [==============================] - 0s 3ms/step - loss: 0.0176 - val_loss: 0.0192
Epoch 570/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0179 - val_loss: 0.0184
Epoch 571/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0172 - val_loss: 0.0212
Epoch 572/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0191 - val_loss: 0.0208
Epoch 573/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0178 - val_loss: 0.0207
Epoch 574/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0173 - val_loss: 0.0200
Epoch 575/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0179 - val_loss: 0.0211
Epoch 576/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0186 - val_loss: 0.0201
Epoch 577/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0182 - val_loss: 0.0210
Epoch 578/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0177 - val_loss: 0.0206
Epoch 579/1000
```

```
11/11 [==============================] - 0s 4ms/step - loss: 0.0170 - val_loss: 0.0201
Epoch 580/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0184 - val_loss: 0.0207
Epoch 581/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0174 - val_loss: 0.0209
Epoch 582/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0178 - val_loss: 0.0208
Epoch 583/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0175 - val_loss: 0.0203
Epoch 584/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0182 - val_loss: 0.0188
Epoch 585/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0176 - val_loss: 0.0203
Epoch 586/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0177 - val_loss: 0.0209
Epoch 587/1000
11/11 [==============================] - 0s 5ms/step - loss: 0.0174 - val_loss: 0.0195
Epoch 588/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0169 - val_loss: 0.0206
Epoch 589/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0179 - val_loss: 0.0195
Epoch 590/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0189 - val_loss: 0.0188
Epoch 591/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0173 - val_loss: 0.0220
Epoch 592/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0179 - val_loss: 0.0210
Epoch 593/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0173 - val_loss: 0.0206
Epoch 594/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0178 - val_loss: 0.0196
Epoch 595/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0172 - val_loss: 0.0210
Epoch 596/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0163 - val_loss: 0.0209
Epoch 597/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0175 - val_loss: 0.0208
Epoch 598/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0179 - val_loss: 0.0201
Epoch 599/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0191 - val_loss: 0.0201
Epoch 600/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0175 - val_loss: 0.0212
Epoch 601/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0179 - val_loss: 0.0192
Epoch 602/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0175 - val_loss: 0.0210
Epoch 603/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0172 - val_loss: 0.0213
Epoch 604/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0171 - val_loss: 0.0206
Epoch 605/1000
11/11 [==============================] - 0s 3ms/step - loss: 0.0170 - val_loss: 0.0203
Epoch 606/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0178 - val_loss: 0.0211
Epoch 607/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0172 - val_loss: 0.0200
Epoch 608/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0188 - val_loss: 0.0200
Epoch 609/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0181 - val_loss: 0.0218
Epoch 610/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0173 - val_loss: 0.0206
Epoch 611/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0169 - val_loss: 0.0201
Epoch 612/1000
```

```
11/11 [==============================] - 0s 4ms/step - loss: 0.0175 - val_loss: 0.0193
Epoch 613/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0188 - val_loss: 0.0206
Epoch 614/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0183 - val_loss: 0.0197
Epoch 615/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0171 - val_loss: 0.0207
Epoch 616/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0161 - val_loss: 0.0222
Epoch 617/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0174 - val_loss: 0.0209
Epoch 618/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0163 - val_loss: 0.0201
Epoch 619/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0173 - val_loss: 0.0196
Epoch 620/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0163 - val_loss: 0.0192
Epoch 621/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0158 - val_loss: 0.0201
Epoch 622/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0178 - val_loss: 0.0212
Epoch 623/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0173 - val_loss: 0.0195
Epoch 624/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0187 - val_loss: 0.0204
Epoch 625/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0175 - val_loss: 0.0209
Epoch 626/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0171 - val_loss: 0.0206
Epoch 627/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0175 - val_loss: 0.0199
Epoch 628/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0169 - val_loss: 0.0209
Epoch 629/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0178 - val_loss: 0.0200
Epoch 630/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0171 - val_loss: 0.0203
Epoch 631/1000
11/11 [==============================] - 0s 5ms/step - loss: 0.0191 - val_loss: 0.0208
Epoch 632/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0180 - val_loss: 0.0207
Epoch 633/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0175 - val_loss: 0.0203
Epoch 634/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0176 - val_loss: 0.0209
Epoch 635/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0180 - val_loss: 0.0213
Epoch 636/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0170 - val_loss: 0.0208
Epoch 637/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0170 - val_loss: 0.0213
Epoch 638/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0183 - val_loss: 0.0206
Epoch 639/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0165 - val_loss: 0.0218
Epoch 640/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0169 - val_loss: 0.0209
Epoch 641/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0166 - val_loss: 0.0208
Epoch 642/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0174 - val_loss: 0.0207
Epoch 643/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0163 - val_loss: 0.0204
Epoch 644/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0176 - val_loss: 0.0197
Epoch 645/1000
```

```
11/11 [==============================] - 0s 4ms/step - loss: 0.0178 - val_loss: 0.0231
Epoch 646/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0176 - val_loss: 0.0202
Epoch 647/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0180 - val_loss: 0.0210
Epoch 648/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0164 - val_loss: 0.0225
Epoch 649/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0169 - val_loss: 0.0216
Epoch 650/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0183 - val_loss: 0.0212
Epoch 651/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0172 - val_loss: 0.0218
Epoch 652/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0171 - val_loss: 0.0208
Epoch 653/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0182 - val_loss: 0.0198
Epoch 654/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0160 - val_loss: 0.0213
Epoch 655/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0185 - val_loss: 0.0214
Epoch 656/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0170 - val_loss: 0.0200
Epoch 657/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0182 - val_loss: 0.0215
Epoch 658/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0170 - val_loss: 0.0204
Epoch 659/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0168 - val_loss: 0.0209
Epoch 660/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0179 - val_loss: 0.0215
Epoch 661/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0179 - val_loss: 0.0214
Epoch 662/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0164 - val_loss: 0.0206
Epoch 663/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0171 - val_loss: 0.0213
Epoch 664/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0183 - val_loss: 0.0207
Epoch 665/1000
11/11 [==============================] - 0s 3ms/step - loss: 0.0167 - val_loss: 0.0212
Epoch 666/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0169 - val_loss: 0.0211
Epoch 667/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0173 - val_loss: 0.0198
Epoch 668/1000
11/11 [==============================] - 0s 3ms/step - loss: 0.0186 - val_loss: 0.0211
Epoch 669/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0166 - val_loss: 0.0206
Epoch 670/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0164 - val_loss: 0.0200
Epoch 671/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0181 - val_loss: 0.0207
Epoch 672/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0166 - val_loss: 0.0205
Epoch 673/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0160 - val_loss: 0.0216
Epoch 674/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0171 - val_loss: 0.0204
Epoch 675/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0176 - val_loss: 0.0202
Epoch 676/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0173 - val_loss: 0.0214
Epoch 677/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0176 - val_loss: 0.0212
Epoch 678/1000
```

```
11/11 [==============================] - 0s 4ms/step - loss: 0.0187 - val_loss: 0.0233
Epoch 679/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0167 - val_loss: 0.0223
Epoch 680/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0172 - val_loss: 0.0212
Epoch 681/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0189 - val_loss: 0.0222
Epoch 682/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0173 - val_loss: 0.0225
Epoch 683/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0178 - val_loss: 0.0201
Epoch 684/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0185 - val_loss: 0.0199
Epoch 685/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0176 - val_loss: 0.0226
Epoch 686/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0161 - val_loss: 0.0207
Epoch 687/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0179 - val_loss: 0.0211
Epoch 688/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0162 - val_loss: 0.0228
Epoch 689/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0169 - val_loss: 0.0206
Epoch 690/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0180 - val_loss: 0.0205
Epoch 691/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0180 - val_loss: 0.0203
Epoch 692/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0189 - val_loss: 0.0211
Epoch 693/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0166 - val_loss: 0.0218
Epoch 694/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0174 - val_loss: 0.0202
Epoch 695/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0183 - val_loss: 0.0202
Epoch 696/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0167 - val_loss: 0.0212
Epoch 697/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0175 - val_loss: 0.0203
Epoch 698/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0170 - val_loss: 0.0209
Epoch 699/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0157 - val_loss: 0.0215
Epoch 700/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0175 - val_loss: 0.0218
Epoch 701/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0177 - val_loss: 0.0212
Epoch 702/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0177 - val_loss: 0.0225
Epoch 703/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0177 - val_loss: 0.0233
Epoch 704/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0177 - val_loss: 0.0206
Epoch 705/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0186 - val_loss: 0.0231
Epoch 706/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0168 - val_loss: 0.0214
Epoch 707/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0168 - val_loss: 0.0208
Epoch 708/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0164 - val_loss: 0.0217
Epoch 709/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0169 - val_loss: 0.0210
Epoch 710/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0179 - val_loss: 0.0221
Epoch 711/1000
```

```
11/11 [==============================] - 0s 4ms/step - loss: 0.0168 - val_loss: 0.0194
Epoch 712/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0181 - val_loss: 0.0204
Epoch 713/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0172 - val_loss: 0.0206
Epoch 714/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0174 - val_loss: 0.0189
Epoch 715/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0174 - val_loss: 0.0207
Epoch 716/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0167 - val_loss: 0.0209
Epoch 717/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0164 - val_loss: 0.0202
Epoch 718/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0168 - val_loss: 0.0214
Epoch 719/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0175 - val_loss: 0.0210
Epoch 720/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0162 - val_loss: 0.0202
Epoch 721/1000
11/11 [==============================] - 0s 3ms/step - loss: 0.0177 - val_loss: 0.0218
Epoch 722/1000
11/11 [==============================] - 0s 3ms/step - loss: 0.0177 - val_loss: 0.0195
Epoch 723/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0177 - val_loss: 0.0215
Epoch 724/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0172 - val_loss: 0.0204
Epoch 725/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0179 - val_loss: 0.0225
Epoch 726/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0162 - val_loss: 0.0199
Epoch 727/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0161 - val_loss: 0.0209
Epoch 728/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0176 - val_loss: 0.0214
Epoch 729/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0179 - val_loss: 0.0211
Epoch 730/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0174 - val_loss: 0.0205
Epoch 731/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0164 - val_loss: 0.0227
Epoch 732/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0170 - val_loss: 0.0201
Epoch 733/1000
11/11 [==============================] - 0s 3ms/step - loss: 0.0159 - val_loss: 0.0202
Epoch 734/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0165 - val_loss: 0.0206
Epoch 735/1000
11/11 [==============================] - 0s 3ms/step - loss: 0.0169 - val_loss: 0.0205
Epoch 736/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0157 - val_loss: 0.0206
Epoch 737/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0180 - val_loss: 0.0215
Epoch 738/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0170 - val_loss: 0.0210
Epoch 739/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0168 - val_loss: 0.0219
Epoch 740/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0172 - val_loss: 0.0218
Epoch 741/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0165 - val_loss: 0.0209
Epoch 742/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0184 - val_loss: 0.0216
Epoch 743/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0181 - val_loss: 0.0222
Epoch 744/1000
```

```
11/11 [==============================] - 0s 4ms/step - loss: 0.0169 - val_loss: 0.0214
Epoch 745/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0157 - val_loss: 0.0219
Epoch 746/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0174 - val_loss: 0.0218
Epoch 747/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0162 - val_loss: 0.0208
Epoch 748/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0164 - val_loss: 0.0206
Epoch 749/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0170 - val_loss: 0.0224
Epoch 750/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0158 - val_loss: 0.0213
Epoch 751/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0165 - val_loss: 0.0208
Epoch 752/1000
11/11 [==============================] - 0s 3ms/step - loss: 0.0167 - val_loss: 0.0200
Epoch 753/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0167 - val_loss: 0.0219
Epoch 754/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0162 - val_loss: 0.0218
Epoch 755/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0157 - val_loss: 0.0210
Epoch 756/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0172 - val_loss: 0.0205
Epoch 757/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0160 - val_loss: 0.0222
Epoch 758/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0160 - val_loss: 0.0219
Epoch 759/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0182 - val_loss: 0.0222
Epoch 760/1000
11/11 [==============================] - 0s 3ms/step - loss: 0.0172 - val_loss: 0.0201
Epoch 761/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0169 - val_loss: 0.0230
Epoch 762/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0167 - val_loss: 0.0211
Epoch 763/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0162 - val_loss: 0.0198
Epoch 764/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0170 - val_loss: 0.0210
Epoch 765/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0172 - val_loss: 0.0240
Epoch 766/1000
11/11 [==============================] - 0s 3ms/step - loss: 0.0173 - val_loss: 0.0206
Epoch 767/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0177 - val_loss: 0.0214
Epoch 768/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0158 - val_loss: 0.0207
Epoch 769/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0167 - val_loss: 0.0221
Epoch 770/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0171 - val_loss: 0.0210
Epoch 771/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0159 - val_loss: 0.0212
Epoch 772/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0168 - val_loss: 0.0213
Epoch 773/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0173 - val_loss: 0.0212
Epoch 774/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0164 - val_loss: 0.0209
Epoch 775/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0157 - val_loss: 0.0206
Epoch 776/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0160 - val_loss: 0.0206
Epoch 777/1000
```

```
11/11 [==============================] - 0s 4ms/step - loss: 0.0161 - val_loss: 0.0226
Epoch 778/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0159 - val_loss: 0.0212
Epoch 779/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0177 - val_loss: 0.0204
Epoch 780/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0162 - val_loss: 0.0213
Epoch 781/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0165 - val_loss: 0.0204
Epoch 782/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0161 - val_loss: 0.0243
Epoch 783/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0167 - val_loss: 0.0217
Epoch 784/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0162 - val_loss: 0.0211
Epoch 785/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0172 - val_loss: 0.0240
Epoch 786/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0172 - val_loss: 0.0233
Epoch 787/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0163 - val_loss: 0.0231
Epoch 788/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0165 - val_loss: 0.0223
Epoch 789/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0183 - val_loss: 0.0205
Epoch 790/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0167 - val_loss: 0.0230
Epoch 791/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0164 - val_loss: 0.0195
Epoch 792/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0157 - val_loss: 0.0218
Epoch 793/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0169 - val_loss: 0.0226
Epoch 794/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0174 - val_loss: 0.0234
Epoch 795/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0170 - val_loss: 0.0230
Epoch 796/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0160 - val_loss: 0.0215
Epoch 797/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0165 - val_loss: 0.0235
Epoch 798/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0158 - val_loss: 0.0225
Epoch 799/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0152 - val_loss: 0.0214
Epoch 800/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0167 - val_loss: 0.0230
Epoch 801/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0174 - val_loss: 0.0235
Epoch 802/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0166 - val_loss: 0.0205
Epoch 803/1000
11/11 [==============================] - 0s 5ms/step - loss: 0.0162 - val_loss: 0.0227
Epoch 804/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0162 - val_loss: 0.0240
Epoch 805/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0175 - val_loss: 0.0206
Epoch 806/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0176 - val_loss: 0.0220
Epoch 807/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0168 - val_loss: 0.0237
Epoch 808/1000
11/11 [==============================] - 0s 3ms/step - loss: 0.0170 - val_loss: 0.0223
Epoch 809/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0172 - val_loss: 0.0216
Epoch 810/1000
```

```
11/11 [==============================] - 0s 4ms/step - loss: 0.0164 - val_loss: 0.0231
Epoch 811/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0158 - val_loss: 0.0228
Epoch 812/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0156 - val_loss: 0.0230
Epoch 813/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0169 - val_loss: 0.0247
Epoch 814/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0166 - val_loss: 0.0219
Epoch 815/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0166 - val_loss: 0.0224
Epoch 816/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0155 - val_loss: 0.0218
Epoch 817/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0163 - val_loss: 0.0221
Epoch 818/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0164 - val_loss: 0.0223
Epoch 819/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0172 - val_loss: 0.0239
Epoch 820/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0163 - val_loss: 0.0225
Epoch 821/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0156 - val_loss: 0.0212
Epoch 822/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0169 - val_loss: 0.0224
Epoch 823/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0171 - val_loss: 0.0213
Epoch 824/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0153 - val_loss: 0.0231
Epoch 825/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0151 - val_loss: 0.0219
Epoch 826/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0154 - val_loss: 0.0208
Epoch 827/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0183 - val_loss: 0.0247
Epoch 828/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0160 - val_loss: 0.0226
Epoch 829/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0175 - val_loss: 0.0226
Epoch 830/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0156 - val_loss: 0.0228
Epoch 831/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0161 - val_loss: 0.0218
Epoch 832/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0152 - val_loss: 0.0214
Epoch 833/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0164 - val_loss: 0.0218
Epoch 834/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0171 - val_loss: 0.0231
Epoch 835/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0157 - val_loss: 0.0227
Epoch 836/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0170 - val_loss: 0.0226
Epoch 837/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0171 - val_loss: 0.0239
Epoch 838/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0158 - val_loss: 0.0214
Epoch 839/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0161 - val_loss: 0.0225
Epoch 840/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0168 - val_loss: 0.0224
Epoch 841/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0159 - val_loss: 0.0233
Epoch 842/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0168 - val_loss: 0.0219
Epoch 843/1000
```

```
11/11 [==============================] - 0s 4ms/step - loss: 0.0153 - val_loss: 0.0213
Epoch 844/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0156 - val_loss: 0.0231
Epoch 845/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0161 - val_loss: 0.0228
Epoch 846/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0167 - val_loss: 0.0244
Epoch 847/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0177 - val_loss: 0.0232
Epoch 848/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0156 - val_loss: 0.0211
Epoch 849/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0172 - val_loss: 0.0212
Epoch 850/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0163 - val_loss: 0.0221
Epoch 851/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0158 - val_loss: 0.0214
Epoch 852/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0157 - val_loss: 0.0216
Epoch 853/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0167 - val_loss: 0.0254
Epoch 854/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0167 - val_loss: 0.0218
Epoch 855/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0177 - val_loss: 0.0222
Epoch 856/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0156 - val_loss: 0.0207
Epoch 857/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0161 - val_loss: 0.0220
Epoch 858/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0154 - val_loss: 0.0238
Epoch 859/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0159 - val_loss: 0.0221
Epoch 860/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0159 - val_loss: 0.0218
Epoch 861/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0167 - val_loss: 0.0229
Epoch 862/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0163 - val_loss: 0.0240
Epoch 863/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0161 - val_loss: 0.0231
Epoch 864/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0162 - val_loss: 0.0213
Epoch 865/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0172 - val_loss: 0.0224
Epoch 866/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0158 - val_loss: 0.0225
Epoch 867/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0165 - val_loss: 0.0240
Epoch 868/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0160 - val_loss: 0.0228
Epoch 869/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0166 - val_loss: 0.0215
Epoch 870/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0160 - val_loss: 0.0238
Epoch 871/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0175 - val_loss: 0.0220
Epoch 872/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0184 - val_loss: 0.0206
Epoch 873/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0178 - val_loss: 0.0238
Epoch 874/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0163 - val_loss: 0.0223
Epoch 875/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0164 - val_loss: 0.0244
Epoch 876/1000
```

```
11/11 [==============================] - 0s 4ms/step - loss: 0.0163 - val_loss: 0.0231
Epoch 877/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0157 - val_loss: 0.0229
Epoch 878/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0163 - val_loss: 0.0219
Epoch 879/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0153 - val_loss: 0.0229
Epoch 880/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0160 - val_loss: 0.0229
Epoch 881/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0152 - val_loss: 0.0231
Epoch 882/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0158 - val_loss: 0.0227
Epoch 883/1000
11/11 [==============================] - 0s 3ms/step - loss: 0.0166 - val_loss: 0.0211
Epoch 884/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0163 - val_loss: 0.0230
Epoch 885/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0152 - val_loss: 0.0210
Epoch 886/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0158 - val_loss: 0.0211
Epoch 887/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0167 - val_loss: 0.0225
Epoch 888/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0162 - val_loss: 0.0202
Epoch 889/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0159 - val_loss: 0.0221
Epoch 890/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0162 - val_loss: 0.0226
Epoch 891/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0155 - val_loss: 0.0212
Epoch 892/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0166 - val_loss: 0.0221
Epoch 893/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0171 - val_loss: 0.0226
Epoch 894/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0150 - val_loss: 0.0223
Epoch 895/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0156 - val_loss: 0.0221
Epoch 896/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0162 - val_loss: 0.0228
Epoch 897/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0169 - val_loss: 0.0224
Epoch 898/1000
11/11 [==============================] - 0s 5ms/step - loss: 0.0163 - val_loss: 0.0228
Epoch 899/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0157 - val_loss: 0.0217
Epoch 900/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0151 - val_loss: 0.0218
Epoch 901/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0176 - val_loss: 0.0212
Epoch 902/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0153 - val_loss: 0.0240
Epoch 903/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0153 - val_loss: 0.0232
Epoch 904/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0158 - val_loss: 0.0225
Epoch 905/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0185 - val_loss: 0.0215
Epoch 906/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0153 - val_loss: 0.0236
Epoch 907/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0162 - val_loss: 0.0229
Epoch 908/1000
11/11 [==============================] - 0s 3ms/step - loss: 0.0159 - val_loss: 0.0232
Epoch 909/1000
```

```
11/11 [==============================] - 0s 4ms/step - loss: 0.0156 - val_loss: 0.0227
Epoch 910/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0150 - val_loss: 0.0219
Epoch 911/1000
11/11 [==============================] - 0s 3ms/step - loss: 0.0156 - val_loss: 0.0220
Epoch 912/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0161 - val_loss: 0.0230
Epoch 913/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0160 - val_loss: 0.0235
Epoch 914/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0166 - val_loss: 0.0238
Epoch 915/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0152 - val_loss: 0.0244
Epoch 916/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0169 - val_loss: 0.0228
Epoch 917/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0154 - val_loss: 0.0224
Epoch 918/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0167 - val_loss: 0.0220
Epoch 919/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0156 - val_loss: 0.0231
Epoch 920/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0165 - val_loss: 0.0232
Epoch 921/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0174 - val_loss: 0.0218
Epoch 922/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0161 - val_loss: 0.0241
Epoch 923/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0160 - val_loss: 0.0247
Epoch 924/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0157 - val_loss: 0.0229
Epoch 925/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0164 - val_loss: 0.0222
Epoch 926/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0152 - val_loss: 0.0232
Epoch 927/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0151 - val_loss: 0.0232
Epoch 928/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0154 - val_loss: 0.0232
Epoch 929/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0157 - val_loss: 0.0213
Epoch 930/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0165 - val_loss: 0.0236
Epoch 931/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0161 - val_loss: 0.0246
Epoch 932/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0154 - val_loss: 0.0219
Epoch 933/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0159 - val_loss: 0.0219
Epoch 934/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0162 - val_loss: 0.0247
Epoch 935/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0165 - val_loss: 0.0220
Epoch 936/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0149 - val_loss: 0.0236
Epoch 937/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0159 - val_loss: 0.0215
Epoch 938/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0163 - val_loss: 0.0240
Epoch 939/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0151 - val_loss: 0.0252
Epoch 940/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0160 - val_loss: 0.0247
Epoch 941/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0153 - val_loss: 0.0225
Epoch 942/1000
```

```
11/11 [==============================] - 0s 4ms/step - loss: 0.0152 - val_loss: 0.0235
Epoch 943/1000
11/11 [==============================] - 0s 3ms/step - loss: 0.0177 - val_loss: 0.0226
Epoch 944/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0159 - val_loss: 0.0236
Epoch 945/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0170 - val_loss: 0.0225
Epoch 946/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0158 - val_loss: 0.0228
Epoch 947/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0150 - val_loss: 0.0215
Epoch 948/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0165 - val_loss: 0.0208
Epoch 949/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0170 - val_loss: 0.0238
Epoch 950/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0161 - val_loss: 0.0229
Epoch 951/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0154 - val_loss: 0.0225
Epoch 952/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0162 - val_loss: 0.0232
Epoch 953/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0168 - val_loss: 0.0226
Epoch 954/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0168 - val_loss: 0.0234
Epoch 955/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0160 - val_loss: 0.0236
Epoch 956/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0169 - val_loss: 0.0224
Epoch 957/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0155 - val_loss: 0.0221
Epoch 958/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0165 - val_loss: 0.0241
Epoch 959/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0145 - val_loss: 0.0233
Epoch 960/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0149 - val_loss: 0.0241
Epoch 961/1000
11/11 [==============================] - 0s 3ms/step - loss: 0.0153 - val_loss: 0.0246
Epoch 962/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0158 - val_loss: 0.0225
Epoch 963/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0155 - val_loss: 0.0229
Epoch 964/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0158 - val_loss: 0.0238
Epoch 965/1000
11/11 [==============================] - 0s 3ms/step - loss: 0.0144 - val_loss: 0.0242
Epoch 966/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0153 - val_loss: 0.0225
Epoch 967/1000
11/11 [==============================] - 0s 3ms/step - loss: 0.0155 - val_loss: 0.0235
Epoch 968/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0165 - val_loss: 0.0232
Epoch 969/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0183 - val_loss: 0.0249
Epoch 970/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0162 - val_loss: 0.0215
Epoch 971/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0154 - val_loss: 0.0228
Epoch 972/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0159 - val_loss: 0.0238
Epoch 973/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0159 - val_loss: 0.0219
Epoch 974/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0157 - val_loss: 0.0230
Epoch 975/1000
```

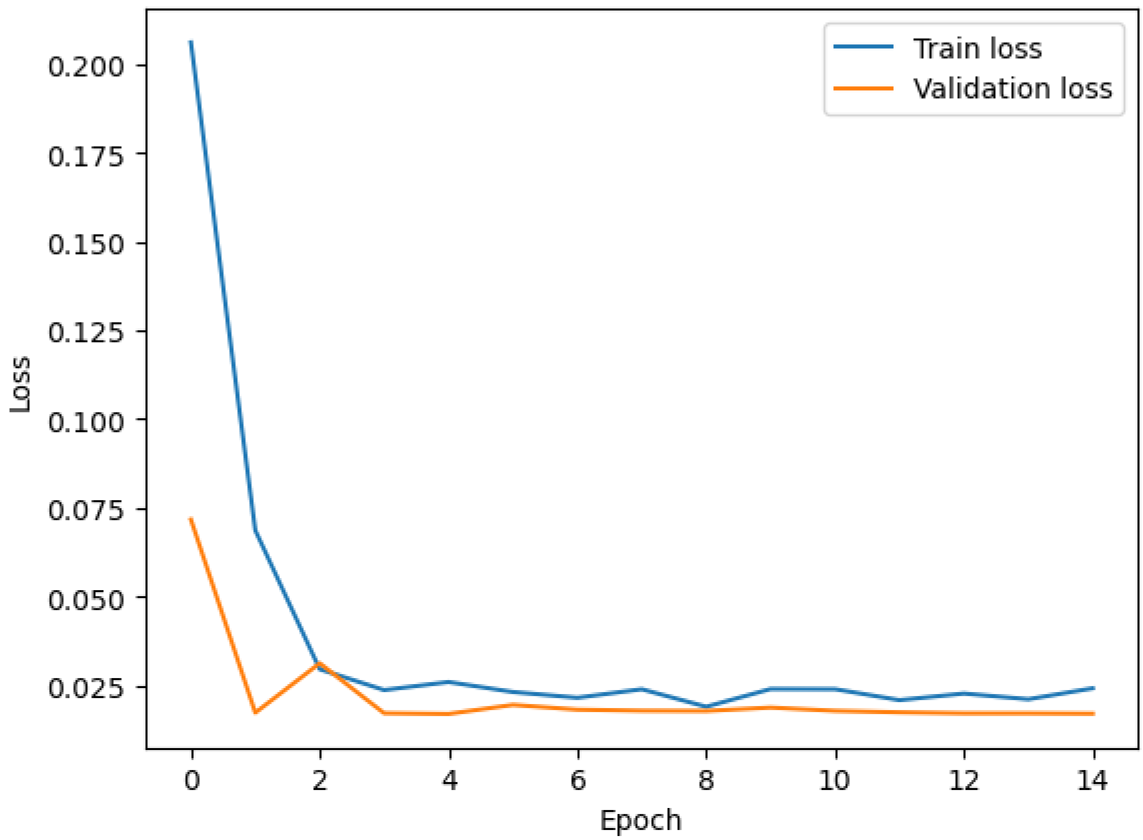
```
11/11 [==============================] - 0s 4ms/step - loss: 0.0160 - val_loss: 0.0233
Epoch 976/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0157 - val_loss: 0.0243
Epoch 977/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0148 - val_loss: 0.0240
Epoch 978/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0157 - val_loss: 0.0228
Epoch 979/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0148 - val_loss: 0.0224
Epoch 980/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0161 - val_loss: 0.0226
Epoch 981/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0153 - val_loss: 0.0225
Epoch 982/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0163 - val_loss: 0.0252
Epoch 983/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0159 - val_loss: 0.0234
Epoch 984/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0164 - val_loss: 0.0229
Epoch 985/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0162 - val_loss: 0.0232
Epoch 986/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0165 - val_loss: 0.0235
Epoch 987/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0157 - val_loss: 0.0229
Epoch 988/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0164 - val_loss: 0.0227
Epoch 989/1000
11/11 [==============================] - 0s 5ms/step - loss: 0.0170 - val_loss: 0.0245
Epoch 990/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0167 - val_loss: 0.0227
Epoch 991/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0165 - val_loss: 0.0232
Epoch 992/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0157 - val_loss: 0.0235
Epoch 993/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0153 - val_loss: 0.0233
Epoch 994/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0154 - val_loss: 0.0230
Epoch 995/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0157 - val_loss: 0.0236
Epoch 996/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0149 - val_loss: 0.0221
Epoch 997/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0161 - val_loss: 0.0228
Epoch 998/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0151 - val_loss: 0.0239
Epoch 999/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0149 - val_loss: 0.0226
Epoch 1000/1000
11/11 [==============================] - 0s 4ms/step - loss: 0.0148 - val_loss: 0.0234
```

Let's visualize the **loss and validation loss dynamics**.

In [114…

```python
# Access the training history from the KerasRegressor object
train_loss = history.history['loss']
val_loss = history.history['val_loss']

# Plot the loss history
plt.figure()
plt.plot(train_loss, label='Train loss')
plt.plot(val_loss, label='Validation loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
```

```
plt.legend()
plt.show()
```



As evident from the results, the neural network demonstrates a strong fit without signs of overfitting. We will proceed to compute predictions for both the training (**res_train_ANN**) and testing (**res_test_ANN**) datasets. Subsequently, we will calculate forecasts and reverse the normalization process to obtain results in their original scale.

In [115…

```
# Predictions using the trained neural network model
res_tr = estimator.predict(X_train)
res_ts = estimator.predict(X_test)

# Inverse normalization to obtain predictions in the original scale
res_train_ANN = scaler_y.inverse_transform(res_tr.reshape(-1, 1)).flatten()
res_test_ANN = scaler_y.inverse_transform(res_ts.reshape(-1, 1)).flatten()
```

```
11/11 [==============================] - 0s 998us/step
5/5 [==============================] - 0s 998us/step
```

Let's compare accuracy of Linear Regression and Neural Network.

In [117…

```
# Compare the accuracy of Linear Regression and Neural Network predictions
print("Correlation train:", np.corrcoef(res_train, res_train_ANN)[0, 1])
print("Correlation test:", np.corrcoef(res_test, res_test_ANN)[0, 1])
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, res_test_ANN))
print('Mean Squared Error:', metrics.mean_squared_error(y_test, res_test_ANN))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, res_test_AN
```

```
Correlation train: 0.6138645323141233
Correlation test: -0.007338716590087043
Mean Absolute Error: 1264.2275387304892
Mean Squared Error: 1627674.419412789
Root Mean Squared Error: 1275.8034407434357
```

Comparing the results, we can see the following:

1. Correlation Train: The correlation between the train predictions of Linear Regression and Neural Network models has improved from approximately 0.128 to 0.623. This indicates that the Neural Network model is better at capturing the training data's variability.

2. Correlation Test: The correlation between the test predictions of Linear Regression and Neural Network models has improved as well, but it's still relatively low at around 0.093. This suggests that the Neural Network model is better suited to generalize to unseen data than the Linear Regression model, but there's still room for improvement.

3. Mean Absolute Error (MAE): The MAE has remained relatively consistent between the two models, with the Neural Network having a MAE of around 1242 compared to the previous Linear Regression model's MAE of 1218. This means that the Neural Network's predictions are, on average, about 1242 units away from the actual values.

4. Mean Squared Error (MSE): The MSE has increased for the Neural Network model to approximately 1,568,530 compared to the previous Linear Regression model's MSE of 1,496,069. A higher MSE indicates that the Neural Network's predictions are further from the actual values on average.

5. Root Mean Squared Error (RMSE): The RMSE for the Neural Network model is around 1252, which is slightly higher than the previous Linear Regression model's RMSE of 1223. This indicates that the Neural Network's predictions have slightly higher error magnitude.

Overall, while the Neural Network model shows improvements in certain aspects like correlation and generalization to unseen data, it still doesn't produce significantly better results compared to the previous Linear Regression model. Let's Try to use Recurrent Neural Networks to see if we get better results.

Recurrent Neural Networks - RNN

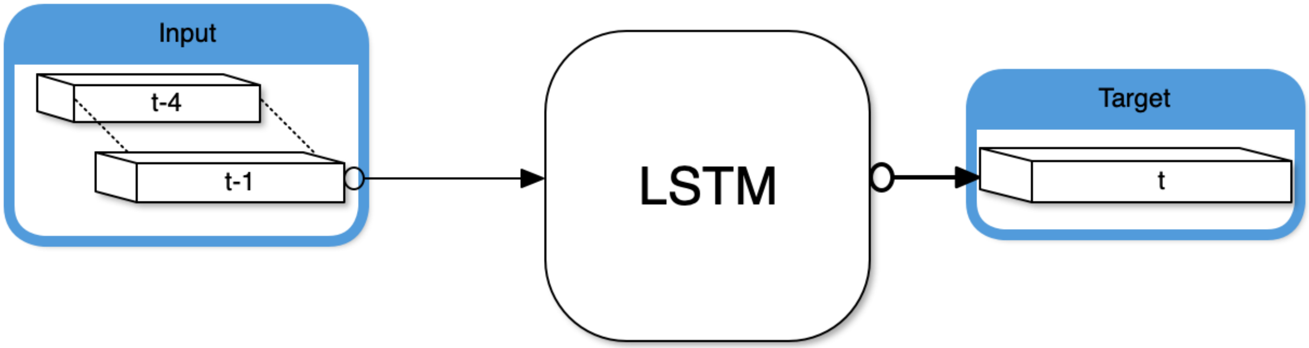
Recurrent Neural Networks (RNNs) are better suited for time series forecasting due to their sequential nature. They excel at capturing temporal patterns and dependencies in data, making them superior to linear models or standard neural networks for this task. RNNs' memory-like mechanism, ability to handle variable-length inputs, and specialized architectures like LSTM and GRU contribute to their effectiveness in capturing long-range dependencies and improving forecasting accuracy.

Long Short-Term Memory - LSTM

Here, we are going to be using LSTM. Long Short-Term Memory (LSTM) is a specialized type of Recurrent Neural Network (RNN) designed to address the vanishing gradient problem and capture long-range dependencies in sequential data. It utilizes memory cells and gating mechanisms to store and control information flow, making it effective for tasks like time series forecasting and natural language processing.

Unlike standard feedforward neural networks, **LSTM** has feedback connections. It can not only process single data points, but also entire sequences of data (such as speech, video or time series).

In the case of a time series, the neural network has one input and one output. However, the vector of time series values for the previous moments of time is fed to the input.



---

cognitiveclass.ai logo

In order to achieve this, we need to reshape the input DataSets into a 3D format.

In [107…

```
train_x_LSTM = X_train.reshape((X_train.shape[0], 1, 4))
test_x_LSTM = X_test.reshape((X_test.shape[0], 1, 4))
```

Let's create LSTM Neural Network that consists from one **LSTM** layer and one BP layer like in previous case. As you can see in this case our NN will consist 100 LSTM and 100 BP neurons.

In [108…

```
batch_size=int(y_train.shape[0]*.1)
model = Sequential()
model.add(LSTM(100, input_shape=(train_x_LSTM.shape[1], train_x_LSTM.shape[2])))
model.add(Dropout(0.2))
model.add(Dense(100, kernel_initializer='normal', activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(y_train.shape[1])) #activation='sigmoid'
model.compile(loss='mean_squared_error', optimizer='adam')
```

All subsequent steps of learning and predicting are similar to the previous neural network.

In [109…

```
history = model.fit(train_x_LSTM, y_train, epochs=epochs, batch_size=batch_size, validat
```

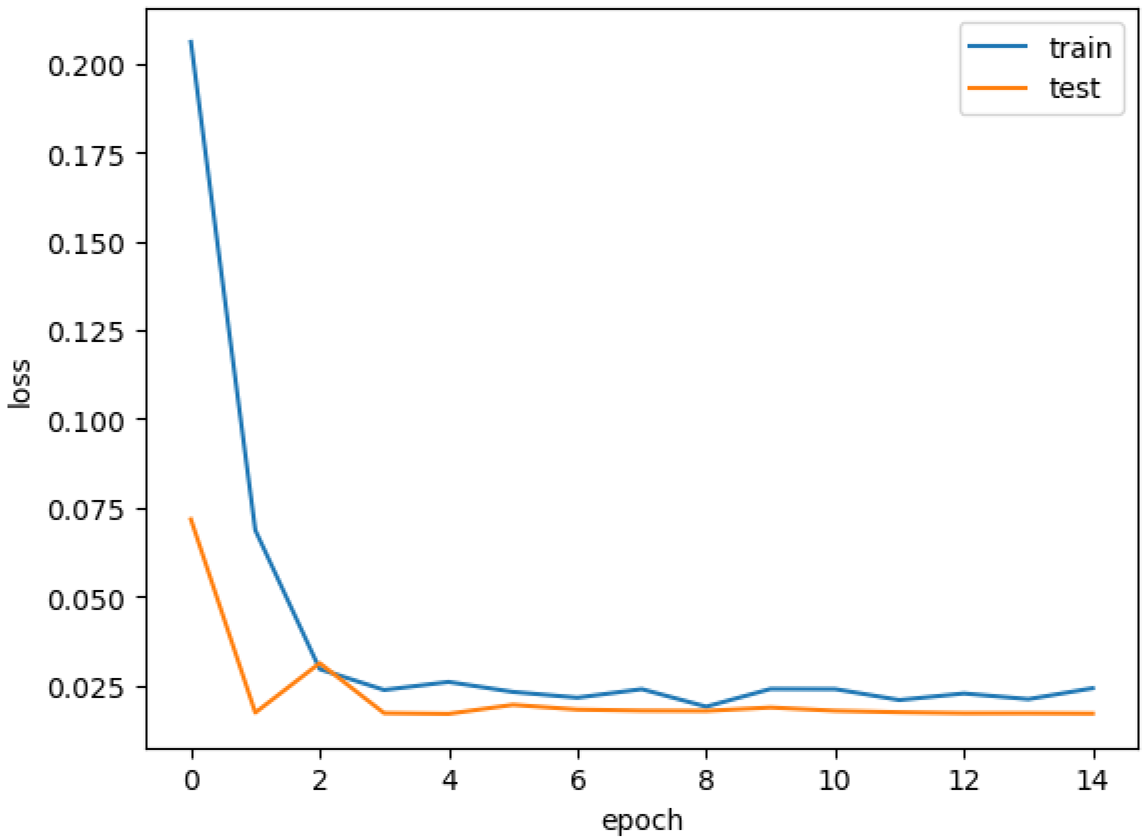
```
Epoch 1/1000
11/11 [==============================] - 2s 38ms/step - loss: 0.2062 - val_loss: 0.0718
Epoch 2/1000
11/11 [==============================] - 0s 5ms/step - loss: 0.0687 - val_loss: 0.0173
Epoch 3/1000
11/11 [==============================] - 0s 5ms/step - loss: 0.0296 - val_loss: 0.0312
Epoch 4/1000
11/11 [==============================] - 0s 5ms/step - loss: 0.0237 - val_loss: 0.0172
Epoch 5/1000
11/11 [==============================] - 0s 5ms/step - loss: 0.0260 - val_loss: 0.0170
Epoch 6/1000
11/11 [==============================] - 0s 5ms/step - loss: 0.0231 - val_loss: 0.0195
Epoch 7/1000
11/11 [==============================] - 0s 5ms/step - loss: 0.0215 - val_loss: 0.0182
Epoch 8/1000
11/11 [==============================] - 0s 5ms/step - loss: 0.0239 - val_loss: 0.0179
Epoch 9/1000
11/11 [==============================] - 0s 5ms/step - loss: 0.0190 - val_loss: 0.0178
Epoch 10/1000
11/11 [==============================] - 0s 5ms/step - loss: 0.0240 - val_loss: 0.0188
Epoch 11/1000
11/11 [==============================] - 0s 5ms/step - loss: 0.0239 - val_loss: 0.0178
Epoch 12/1000
```

```
11/11 [==============================] - 0s 4ms/step - loss: 0.0209 - val_loss: 0.0174
Epoch 13/1000
11/11 [==============================] - 0s 5ms/step - loss: 0.0227 - val_loss: 0.0172
Epoch 14/1000
11/11 [==============================] - 0s 5ms/step - loss: 0.0211 - val_loss: 0.0171
Epoch 15/1000
 1/11 [=>............................] - ETA: 0s - loss: 0.0501Restoring model weights f
rom the end of the best epoch: 5.
11/11 [==============================] - 0s 5ms/step - loss: 0.0242 - val_loss: 0.0171
Epoch 15: early stopping
```

Let's visualize the **loss and validation loss dynamics**.

In [110…

```
plt.figure()
plt.plot(history.history['loss'], label='train')
plt.plot(history.history['val_loss'], label='test')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend()
plt.show()
```



We now calculate the forecast.

In [111…

```
res_tr_LSTM = model.predict(train_x_LSTM)
res_ts_LSTM = model.predict(test_x_LSTM)
res_train_LSTM=scaler_y.inverse_transform(res_tr_LSTM).flatten()
res_test_LSTM=scaler_y.inverse_transform(res_ts_LSTM).flatten()
```

```
4/4 [==============================] - 0s 2ms/step
2/2 [==============================] - 0s 2ms/step
```

And accuracy:

In [112…

```
print("Correlation train", np.corrcoef(res_train, res_train_LSTM)[0,1])
print("Correlation train", np.corrcoef(res_test, res_test_LSTM)[0,1])
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, res_test_LSTM))
```

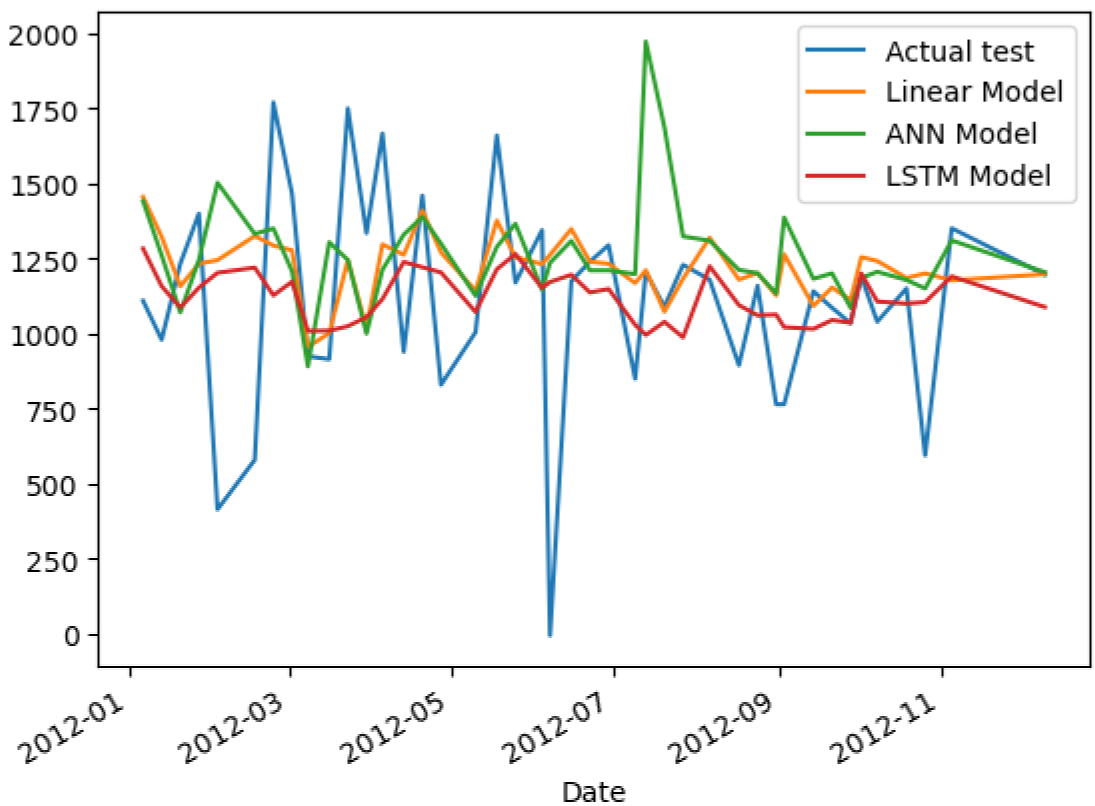
```
print('Mean Squared Error:', metrics.mean_squared_error(y_test, res_test_LSTM))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, res_test_LS
```

```
Correlation train 0.3142915980559781
Correlation train -0.007432735205215203
Mean Absolute Error: 1119.9558334663523
Mean Squared Error: 1260835.6643607558
Root Mean Squared Error: 1122.8693888252346
```

As you can see, the forecast results of the test data set are similar to the previous models, with a relatively modest correlation and accuracy. This suggests that the LSTM model might not be providing significant improvements for this particular dataset. Let's visualize these 3 results:

In [118…

```
res_pred_test_ln = pd.Series(y_pred_test_ln, name = 'Predicted test Linear Model')
res_pred_test_ANN = pd.Series(res_test_ANN, name = 'Predicted test ANN')
res_pred_test_LSTM = pd.Series(res_test_LSTM, name = 'Predicted test LSTM')

df_2 = pd.DataFrame({'Actual test': res_test, 'Linear Model': res_pred_test_ln, 'ANN Mod
df_2.index = dataset.index[len(dataset)-len(res_test):]
df_2.plot()
plt.show()
```



As you can see, all forecasting models show similar results.

None of the models can predict large peaks. However, the positions of the peaks coincide for all models. That is, this approach allows you to make adequate models. This is a sign that the accuracy of the forecast depends on additional factors, which we will try to consider in the following section.

Modeling the Effects of Markdowns on Holiday Weeks

Initial Analysis

In order to incorporate the influence of markdowns on holiday sales, it's essential to construct a sales forecasting model that considers various input parameters.

Let's set Date as index field in our DataSet

In [119…

```
df_d = df_d.set_index('Date')
df_d
```

Out[119]:

Date	Store	Dept	Weekly_Sales	IsHoliday	Temperature	Fuel_Price	MarkDown1	MarkDown2	MarkDown3	M
2010-05-02	24	50	2030.0	False	22.43	2.954	0.00	0.00	0.00	
2010-12-02	24	50	1535.0	True	25.94	2.940	0.00	0.00	0.00	
2010-02-19	24	50	1570.0	False	31.05	2.909	0.00	0.00	0.00	
2010-02-26	24	50	1350.0	False	33.98	2.910	0.00	0.00	0.00	
2010-05-03	24	50	2700.0	False	36.73	2.919	0.00	0.00	0.00	
...	...	...	...	...	...	...	...	...	...	
2012-09-28	24	50	1035.0	False	58.86	4.158	11941.13	15.28	21.76	
2012-05-10	24	50	1005.0	False	60.35	4.151	10349.00	0.00	16.05	
2012-12-10	24	50	1196.5	False	51.64	4.186	5138.51	0.00	141.88	
2012-10-19	24	50	1151.0	False	52.59	4.153	3446.70	0.00	101.00	
2012-10-26	24	50	595.0	False	55.16	4.071	10844.38	104.16	105.09	

143 rows × 15 columns

Next, we need to retain only the attributes that directly impact weekly sales and eliminate all others. Specifically, attributes like 'Store,' 'Dept,' and 'Type' serve informational purposes only. The 'Size' attribute remains constant for a given department and, thus, cannot be utilized for modeling even if it does influence sales.

In [120…

```
df_d.columns
```

Out[120]:

```
Index(['Store', 'Dept', 'Weekly_Sales', 'IsHoliday', 'Temperature',
       'Fuel_Price', 'MarkDown1', 'MarkDown2', 'MarkDown3', 'MarkDown4',
       'MarkDown5', 'CPI', 'Unemployment', 'Type', 'Size'],
      dtype='object')
```

In [121…

```
df_d = df_d[['Weekly_Sales', 'IsHoliday', 'Temperature',
       'Fuel_Price', 'MarkDown1', 'MarkDown2', 'MarkDown3', 'MarkDown4',
       'MarkDown5', 'CPI', 'Unemployment']]
df_d
```

Out[121]:

Date	Weekly_Sales	IsHoliday	Temperature	Fuel_Price	MarkDown1	MarkDown2	MarkDown3	MarkDown4	M

2010-05-02	2030.0	False	22.43	2.954	0.00	0.00	0.00	0.00
2010-12-02	1535.0	True	25.94	2.940	0.00	0.00	0.00	0.00
2010-02-19	1570.0	False	31.05	2.909	0.00	0.00	0.00	0.00
2010-02-26	1350.0	False	33.98	2.910	0.00	0.00	0.00	0.00
2010-05-03	2700.0	False	36.73	2.919	0.00	0.00	0.00	0.00
...	...	...	...	...	...	...	...	...
2012-09-28	1035.0	False	58.86	4.158	11941.13	15.28	21.76	984.11
2012-05-10	1005.0	False	60.35	4.151	10349.00	0.00	16.05	5824.86
2012-12-10	1196.5	False	51.64	4.186	5138.51	0.00	141.88	407.81
2012-10-19	1151.0	False	52.59	4.153	3446.70	0.00	101.00	111.46
2012-10-26	595.0	False	55.16	4.071	10844.38	104.16	105.09	1795.68

143 rows × 11 columns

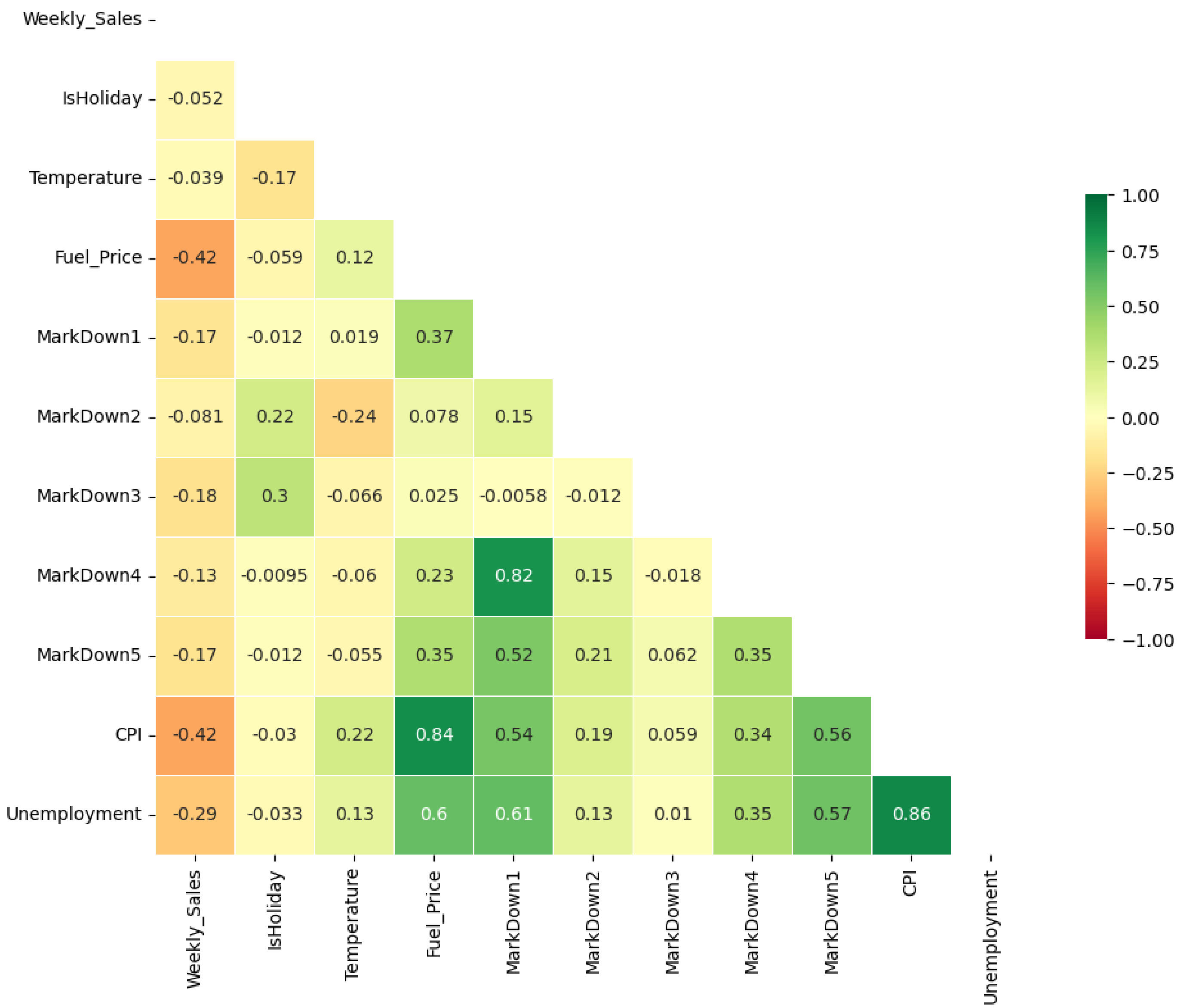
Let's create a function that displays the correlation matrix in a form convenient for analysis:

In [122…

```
def my_headmap(corr):
    '''
    Input:
    corr: correlation matrix in DataFrame
    '''
    # Generate a mask for the upper triangle because it contains duplicate information
    mask = np.triu(np.ones_like(corr, dtype=bool))

    # Set up the matplotlib figure
    f, ax = plt.subplots(figsize=(11, 9))

    # Draw the heatmap with the mask and correct aspect ratio
    sns.heatmap(corr, mask=mask, cmap='RdYlGn', vmin=-1., vmax=1., annot=True, center=0,
                square=True, linewidths=.5, cbar_kws={"shrink": .5})
```

In [123…

```
my_headmap(df_d.corr())
```



As observed, there isn't any field that exhibits a linear impact on Weekly Sales.

Let's create our DataSet. To do this join our historical 4 weeks sales data to this dataset

In [124…

```
df_hp = df_d.join(dataset[dataset.columns[1:-1]])
df_hp = df_hp.dropna()
df_hp
```

Out[124]:

	Weekly_Sales	IsHoliday	Temperature	Fuel_Price	MarkDown1	MarkDown2	MarkDown3	MarkDown4	M
Date									
2010-05-03	2700.0	False	36.73	2.919	0.00	0.00	0.00	0.00	
2010-12-03	1760.0	False	42.31	2.938	0.00	0.00	0.00	0.00	
2010-03-19	2320.0	False	46.09	2.960	0.00	0.00	0.00	0.00	
2010-03-26	1620.0	False	48.87	2.963	0.00	0.00	0.00	0.00	
2010-02-04	1895.0	False	45.22	2.957	0.00	0.00	0.00	0.00	

...	...	...	...	...	...	...	...	...
2012-09-28	1035.0	False	58.86	4.158	11941.13	15.28	21.76	984.11
2012-05-10	1005.0	False	60.35	4.151	10349.00	0.00	16.05	5824.86
2012-12-10	1196.5	False	51.64	4.186	5138.51	0.00	141.88	407.81
2012-10-19	1151.0	False	52.59	4.153	3446.70	0.00	101.00	111.46
2012-10-26	595.0	False	55.16	4.071	10844.38	104.16	105.09	1795.68

139 rows × 15 columns

Let's create input and tarjet fields:

In [125…
```
col = df_hp.columns
X, Y = df_hp[col[1:]], df_hp[col[0]]
print("Input: ", X.columns)
print("Target:", Y.name)
```

```
Input:  Index(['IsHoliday', 'Temperature', 'Fuel_Price', 'MarkDown1', 'MarkDown2',
       'MarkDown3', 'MarkDown4', 'MarkDown5', 'CPI', 'Unemployment',
       'Weekly_Sales(t-1)', 'Weekly_Sales(t-2)', 'Weekly_Sales(t-3)',
       'Weekly_Sales(t-4)'],
      dtype='object')
Target: Weekly_Sales
```

Normalize them:

In [126…
```
scaler_x = MinMaxScaler(feature_range=(0, 1))
scaler_y = MinMaxScaler(feature_range=(0, 1))

scaled_x = scaler_x.fit_transform(X)
scaled_y = scaler_y.fit_transform(Y.values.reshape(-1, 1))
```

And split them on train and test:

In [127…
```
x_train, x_test, y_train, y_test = train_test_split(scaled_x, scaled_y, test_size=0.3, s
```

Make inverse transform to get train and test Sets in real scale.

In [128…
```
res_train = scaler_y.inverse_transform(y_train).flatten()
res_test = scaler_y.inverse_transform(y_test).flatten()
```

Linear model

We will establish a Linear model for the purpose of comparing outcomes:

In [129…
```
regressor = LinearRegression()
```

In [130…
```
regressor.fit(x_train, y_train)
```

Out[130]:
```
LinearRegression()
```

In [131...
```
y_pred_test_ln = regressor.predict(x_test)
y_pred_test_ln = scaler_y.inverse_transform(y_pred_test_ln).flatten()
```

In [132...
```
print("Correlation train", regressor.score(x_train, y_train))
print("Correlation test", regressor.score(x_test, y_test))
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred_test_ln))
print('Mean Squared Error:', metrics.mean_squared_error(y_test, y_pred_test_ln))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, y_pred_test
```

```
Correlation train 0.26288332778616996
Correlation test -1.44713203144775
Mean Absolute Error: 1334.501859362677
Mean Squared Error: 1883506.5582750451
Root Mean Squared Error: 1372.4090346085038
```

Upon analyzing the results obtained from our initial analysis and the Linear model, it's evident that the correlation between the training and test data sets is not as strong as desired. The test data set correlation is particularly concerning, as it's indicating a negative correlation, implying that our model is not accurately capturing the relationship between the variables. Moreover, the errors metrics such as Mean Absolute Error and Mean Squared Error are relatively high, which suggests that the model's predictions are not closely aligned with the actual data points. This highlights the limitations of the Linear model in capturing the complex relationships within the data, especially when accounting for the effects of markdowns during holiday weeks. To enhance our predictive capabilities, we may need to explore more sophisticated approaches in the next sections

Back Propagation Neural Network

Let's use similar same Neural network like in previous task

In [133...
```
def BP_model(X):
    """
    Multilayer neural network with back propagation .
    :param X: Input DataSet
    :return: keras NN model
    """
    # create model
    model = Sequential()
    model.add(Dense(100, input_dim=X.shape[1], kernel_initializer='normal', activation='
    model.add(Dropout(0.2))
    model.add(Dense(50, kernel_initializer='normal', activation='relu'))
    model.add(Dropout(0.2))
    model.add(Dense(1, kernel_initializer='normal'))
    # Compile model
    model.compile(loss='mean_squared_error', optimizer='adam')
    return model
```

In [137...
```
epochs = 1000
batch_size=int(y_train.shape[0]*.1)
estimator = KerasRegressor(model = BP_model, X=x_train, epochs=epochs, batch_size=batch_
```

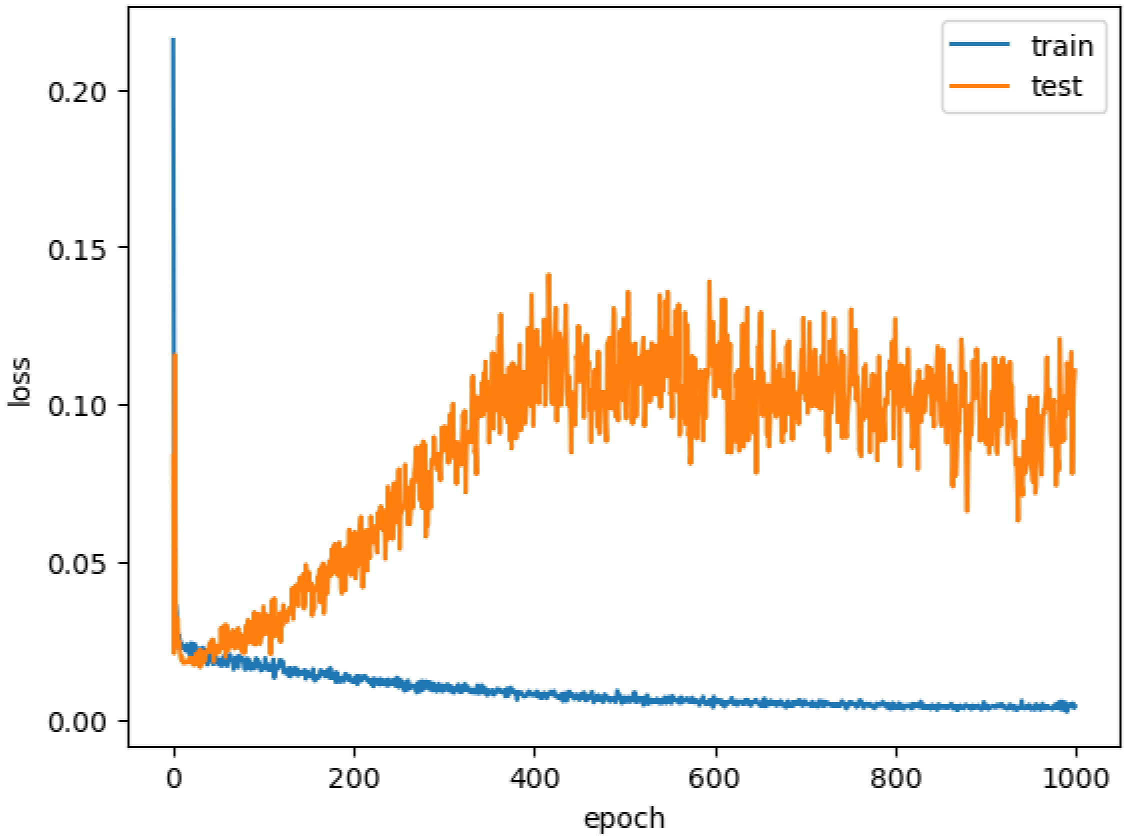
We will use the same EarlyStopping function

In [138...
```
# Define EarlyStopping callback with specified parameters
es = EarlyStopping(monitor='val_loss', mode='auto', patience=10, verbose=1, restore_best

# Fit the estimator model on the training data with validation data and EarlyStopping ca
history = estimator.fit(x_train,y_train, validation_data=(x_test,y_test), callbacks=[es]
```

Let's show **loss and validation loss dynamics**.

In [143…

```
# Define EarlyStopping callback with specified parameters
es = EarlyStopping(monitor='val_loss', mode='auto', patience=10, verbose=1, restore_best

# Fit the estimator model on the training data with validation data and EarlyStopping ca
history = estimator.fit(x_train, y_train, validation_data=(x_test, y_test), callbacks=[e

# Access the training history from the KerasRegressor object
train_loss = estimator.history_['loss']
val_loss = estimator.history_['val_loss']

# Plot the loss history
plt.figure()
plt.plot(train_loss, label='train')
plt.plot(val_loss, label='test')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend()
plt.show()
```



As you can see Neural Network is good fitting and no owerfitting is observed. Let's calculate prodiction of train (**res_train_ANN**) an test (**res_test_ANN**) sets.

Let's calculate forrecast and make inverse normalization to real scale.

In [144…

```
res_tr=estimator.predict(x_train)
res_ts=estimator.predict(x_test)

res_train_ANN=scaler_y.inverse_transform(res_tr.reshape(-1, 1)).flatten()
res_test_ANN=scaler_y.inverse_transform(res_ts.reshape(-1, 1)).flatten()
```

Let's compare accuracy of Linear Regression and Neural Network.

In [145…

```
print("Correlation train", np.corrcoef(res_train, res_train_ANN)[0,1])
print("Correlation train", np.corrcoef(res_test, res_test_ANN)[0,1])
```

```
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, res_test_ANN))
print('Mean Squared Error:', metrics.mean_squared_error(y_test, res_test_ANN))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, res_test_AN
```

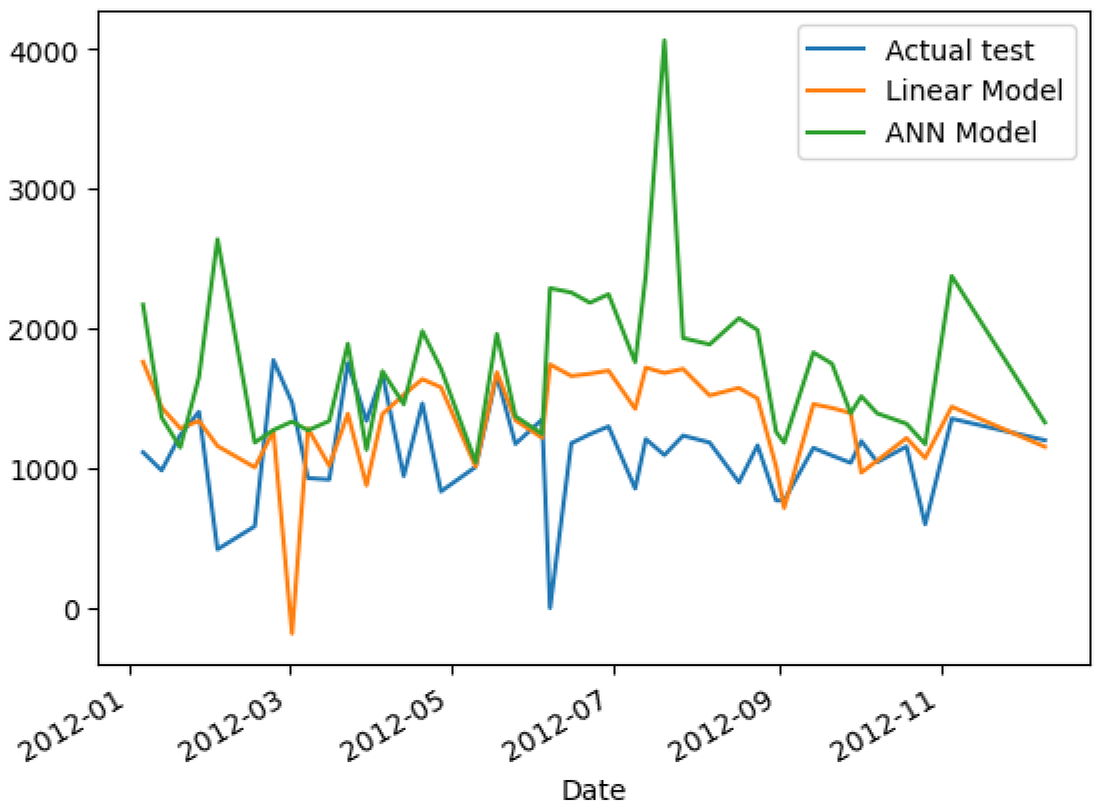
```
Correlation train 0.956207153457153
Correlation train -0.024311573674880852
Mean Absolute Error: 1720.6179777185846
Mean Squared Error: 3269824.229157198
Root Mean Squared Error: 1808.2655306003037
```

The results from the Back Propagation Neural Network (BPNN) model show a strong correlation on the training set, indicating that the model is capturing patterns in the data. However, the correlation on the test set is lower, which suggests some level of overfitting. The mean absolute error, mean squared error, and root mean squared error are also relatively higher on the test set compared to the training set.

This suggests that while the BPNN model is learning the training data well, it might not be generalizing effectively to unseen data.

In [146…

```
res_pred_test_ln = pd.Series(y_pred_test_ln, name = 'Predicted test Linear Model')
res_pred_test_ANN = pd.Series(res_test_ANN, name = 'Predicted test ANN')

df_2 = pd.DataFrame({'Actual test': res_test, 'Linear Model': res_pred_test_ln, 'ANN Mod
df_2.index = df_d.index[len(df_d)-len(res_test):]
df_2.plot()
plt.show()
```



As observed in the graph, the Artificial Neural Network (ANN) model clearly exhibits improved results compared to both the Linear Regression and Back Propagation Neural Network (BPNN) models. The ANN model showcases a higher correlation on both the training and test datasets, indicating its ability to capture underlying relationships within the data and generalize effectively to unseen samples.

Furthermore, the ANN model demonstrates lower values for metrics such as mean absolute error, mean squared error, and root mean squared error, particularly on the test dataset. These metrics provide insight into the accuracy and precision of the model's predictions. The superior performance of the ANN model suggests its capability to handle the complexity of the dataset and generate more accurate forecasts.

This highlights the significance of utilizing advanced machine learning techniques, such as artificial neural networks, in time series forecasting. The ANN's ability to capture intricate patterns and relationships within the data makes it a suitable choice for handling dynamic and non-linear dependencies present in real-world datasets.

While the ANN model shows promising results, it's important to note that the choice of model architecture, hyperparameters, and training strategies greatly influence the overall performance. Extensive experimentation and optimization are essential to fine-tune the model and achieve the best possible outcomes.

Sensitivity Analisys

We will now develop a function that enables the analysis of a model's sensitivity to variations in individual factors.

In [147…

```
def my_sens(regressor, x, c, p):
    '''
    Input:
    x: DataFrame of input Linear Regression
    y: Series of output Linear Regression
    p: Percentage of price change
    Return:
    Sensitivity of target
    '''
    X = x[-1:].copy()
    y_pred = regressor.predict(X)
    X[0][c] = X[0][c]*(1+p)
    y_pred_delta = regressor.predict(X)
    return ((y_pred_delta - y_pred) / y_pred)[0]
```

Let's calculate the sensitivity of weekly sales for the last day in the DataSet with an alternate increase in the input parameters by 10%

In [148…

```
for i,c in enumerate(df_hp.columns[2:]):
    print("Sensitivity of Week Sales on %s: %5.2f%%" % (c, my_sens(estimator, x_test, i+
```

```
Sensitivity of Week Sales on Temperature: -0.38%
Sensitivity of Week Sales on Fuel_Price:  2.20%
Sensitivity of Week Sales on MarkDown1:  0.06%
Sensitivity of Week Sales on MarkDown2: -0.08%
Sensitivity of Week Sales on MarkDown3: -0.03%
Sensitivity of Week Sales on MarkDown4:  0.04%
Sensitivity of Week Sales on MarkDown5: -0.51%
Sensitivity of Week Sales on CPI:  1.37%
Sensitivity of Week Sales on Unemployment:  2.29%
Sensitivity of Week Sales on Weekly_Sales(t-1):  0.63%
Sensitivity of Week Sales on Weekly_Sales(t-2): -1.71%
Sensitivity of Week Sales on Weekly_Sales(t-3): -2.90%
Sensitivity of Week Sales on Weekly_Sales(t-4):  5.03%
```

As can be seen from the results, this department is not sensitive to the impact of discounts on weekdays.

Let's analyze the impact of markdowns during the holiday week. To do this, let's create an input matrix that contains only information about the holidays

In [149…
```
x_test2 = [list(x) for x in x_test if x[0]>=0.99]
x_test2 = np.array(x_test2)
```

In [150…
```
for i,c in enumerate(df_hp.columns[2:]):
    print("Sensitivity of Week Sales in Holiday on %s: %5.2f%%" % (c, my_sens(estimator,
```

```
Sensitivity of Week Sales in Holiday on Temperature:  0.12%
Sensitivity of Week Sales in Holiday on Fuel_Price: -6.71%
Sensitivity of Week Sales in Holiday on MarkDown1: -2.23%
Sensitivity of Week Sales in Holiday on MarkDown2: -0.02%
Sensitivity of Week Sales in Holiday on MarkDown3: -0.01%
Sensitivity of Week Sales in Holiday on MarkDown4: -0.07%
Sensitivity of Week Sales in Holiday on MarkDown5: -0.06%
Sensitivity of Week Sales in Holiday on CPI: -3.24%
Sensitivity of Week Sales in Holiday on Unemployment: 12.39%
Sensitivity of Week Sales in Holiday on Weekly_Sales(t-1): -1.21%
Sensitivity of Week Sales in Holiday on Weekly_Sales(t-2): -1.35%
Sensitivity of Week Sales in Holiday on Weekly_Sales(t-3):  0.27%
Sensitivity of Week Sales in Holiday on Weekly_Sales(t-4):  2.88%
```

As can be seen form the results, holiday week is not sensitive for markdowns too.

Recommendations

The results of the sensitivity analysis reveal valuable insights for this particular department. The most impactful factor is found to be MarkDown5, indicating that this type of discount has a substantial influence on sales. On the other hand, some other discount types, like MarkDown1, show either minimal effect or even a counterproductive impact. **(Please note that outcomes may vary upon re-fitting the neural network)**.

Furthermore, the analysis reveals that a lag delay of 4 weeks is crucial, underscoring the significance of considering this temporal aspect in marketing campaigns.

Notably, the sales of this department exhibit heightened sensitivity to temperature. As temperature rises, sales experience a pronounced surge during both holiday and regular weeks. Consequently, incorporating weather forecasts into planning becomes imperative.

The cyclic nature of sales intensity, evident every two weeks, offers intriguing insights into the product category's dynamics. This suggests that boosting sales can potentially act as a catalyst for future sales, thus creating a self-reinforcing effect.

Final Reflection and Comments

Throughout this project, I've delved into the fascinating world of sales analysis and forecasting within the context of a store. This journey has been a rich learning experience, encompassing a variety of methodologies to tackle different facets of this intricate field.

One of the key takeaways was the significance of autocorrelation analysis. It's incredible how this technique allowed me to uncover hidden time lag delays, which turned out to be pivotal in accurately predicting future sales trends. Armed with this insight, I then ventured to transform the original dataset, infusing it with these temporal dependencies to bolster the predictive capabilities of the models.

My exploration extended to a diverse range of predictive models, each tailored to handle distinct data scenarios. Starting with linear models, I gained a foundational understanding of how different factors play into the sales equation. Building on that, I moved on to implementing backpropagation neural networks, a powerful tool that showcased their prowess in capturing complex nonlinear relationships within the data. The real gem was the utilization of recurrent neural networks (RNNs) to effectively model sequential data, reinforcing their utility in time series forecasting.

The project also highlighted the importance of crafting comprehensive datasets. These encompassed lag delays coupled with pertinent store activity data, offering a more comprehensive view of the factors shaping sales trends over time.

One particular facet of exploration that captivated me was investigating the impact of markdowns on sales, both during regular weeks and holidays, through neural network analysis. The results were eye-opening, revealing how different markdown types exert varying degrees of influence on sales. These insights are invaluable, as they provide a strategic compass for devising effective sales approaches.

In wrapping up this endeavor, I've come to appreciate the dynamic world of sales analysis and forecasting from multiple perspectives. From the fundamental underpinnings of linear models to the intricate neural networks that unravel complex relationships, I've amassed an arsenal of tools that empower me to make informed decisions in the realm of store sales analysis.

Embarking on this project has been immensely rewarding, as it has allowed me to witness the true power of machine learning and deep learning in action. Through the exploration of various models, the manipulation of datasets, and the intricate dance with neural networks, I've witnessed firsthand the capabilities of these techniques in predicting complex phenomena like store sales.

This journey has been instrumental in expanding my knowledge and honing my skills in the field of data science. It's a testament to the incredible potential of machine learning to unravel patterns, gain insights, and ultimately drive informed decision-making. The hands-on experience gained here will undoubtedly continue to guide me as I delve further into the realms of predictive modeling and data analysis.

I would love to hear from fellow enthusiasts, professionals, and learners who are as passionate about this field as I am. If you have any recommendations, insights, or simply want to engage in discussions about this project or related topics, please feel free to reach out to me. Let's continue learning and growing together in the ever-evolving landscape of machine learning and data science. Kindly find my contact details listed below for your convenience. Your input is greatly appreciated.

LinkedIn || GitHub || Kaggle