Time Series Forecasting - Sales

A Project Report

Submitted by

Nivedha. A
CB.SC.I5DAS19023

In partial fulfilment of the requirements for the award of the degree of

Integrated Master of Science
in
Data Science

Amrita School of Physical Sciences
Amrita Vishwa Vidyapeetham
Coimbatore – 641112

June 2023

Amrita School of Physical Sciences
Amrita Vishwa Vidyapeetham, Coimbatore – 641112

BONAFIDE CERTIFICATE

This is to certify that the project report entitled "**Time series Forecasting - Sales**" submitted by **Nivedha. A (CB.SC.I5DAS19023)** In partial fulfilment of the requirements for the award of the degree of **Integrated Master of Science** in **Data Science** is a bonafide record of the work carried out under my guidance and supervision at Amrita School of Physical Sciences, Amrita Vishwa Vidyapeetham, Coimbatore.

Dr. Murali Krishna P
Project Coordinator

Dr. J Ravichandran
Chairperson, Deparment of Mathematics

The project was evaluated by is on:

Examiner

Amrita School of Physical Sciences
Amrita Vishwa Vidyapeetham, Coimbatore – 641112

DECLARTION

I, **Nivedha. A (CB.SC.I5DAS19023)**, hereby declare that the dissertation entitled "**Time Series Forecasting- Sales**", is the record of the original work done by me. To the best of knowledge, this work has not formed the basis of the award of any degree/diploma/associateship/fellowship/or a similar award to any candidate in any university.

Place: Coimbatore

Signature of the student

Date: June 7th 2023

COUNTERSIGNED
Dr. Murali Krishna P
Project Coordinator
Department of Mathematics

ACKNOWLEDGEMENTS

I would like to express my special thanks of gratitude to **Dr. Ravichandran J**, Chairperson, Department of Mathematics, Amrita School of Engineering, for giving me the golden opportunity to do this wonderful project on the topic of **Time Series Forecasting of Sales**, this project helped me to learn and discover about so many new things.

I would also like to thank my class advisor, **Dr. Murali Krishnan P**, Department of Mathematics, Amrita School of Engineering, for supporting me with this project idea and guiding me through out.

I would like to thank the faculty members of Department of Mathematics for allowing me to work for this Project.

My heartfelt thanks to all the staffs for their invaluable teachings over the years and constant inspiration for my project work.

I owe a special debt gratitude to my revered parents and family for their blessings and inspirations.

Coimbatore Nivedha. A

June 2023

Table Of Contents

1.ABSTRACT:

This project focuses on time series forecasting for sales prediction using multiple models, including Multilayer Perceptron (MLP), Convolutional Neural Network (CNN), Long Short-Term Memory (LSTM), CNN-LSTM hybrid, and Autoregressive Integrated Moving Average (ARIMA). The goal is to compare the performance of these models and identify the most effective approach for accurate sales forecasting. Various evaluation metrics, such as RMSE, MAE, MAPE, MSE, and R-squared score, are utilized to assess the accuracy of the models. The results provide insights into the strengths and limitations of each model, enabling businesses to make informed decisions regarding their sales forecasting strategies.

2.INTRODUCTION:

Effective sales forecasting is crucial for businesses to optimize inventory management, resource allocation, and overall decision-making. Time series forecasting techniques provide valuable insights into future sales trends based on historical data. In this project, we aim to explore and compare different models for sales prediction, evaluating their performance and accuracy.

The dataset used in this study contains historical sales data, including dates and corresponding sales values. Our objective is to develop models capable of capturing the temporal patterns and dependencies within the data, enabling accurate predictions of future sales. We consider a range of models, including MLP, CNN, LSTM, CNN-LSTM hybrid, and ARIMA, which have demonstrated effectiveness in time series forecasting tasks.

The methodology involves preprocessing the data, splitting it into training and validation sets, and training each model using appropriate techniques. We focus on understanding the architectural characteristics, training procedures, and hyperparameter tuning for each model to achieve optimal performance. To evaluate the accuracy of the models, we employ various evaluation metrics such as RMSE, MAE, MAPE, MSE, and R-squared score.

The results of our experiments provide valuable insights into the strengths and weaknesses of each model for sales forecasting. By comparing the performance of the different models, we aim to identify the most effective approach that can be employed by businesses for accurate sales predictions. The findings of this study contribute to the field of time series forecasting, offering practical guidance and recommendations for businesses seeking to enhance their sales forecasting capabilities.

In summary, this project focuses on evaluating multiple models for sales forecasting using historical sales data. By comparing their performance and accuracy, we aim to identify the most effective model for accurate sales predictions. The outcomes of this study have practical implications for businesses, enabling them to make informed decisions regarding their sales forecasting strategies and ultimately improve their overall business performance.

3.IMPLEMENTATION:

- 3.1 About the dataset:
  The dataset used in this project is obtained from Kaggle and consists of historical sales data for a period of 5 years. The dataset contains information about the date, store, item, and sales. This dataset gives information about the sales of 50 different items across 10 different stores for 5 years.

	date	store	item	sales
0	2013-01-01	1	1	13
1	2013-01-02	1	1	11
2	2013-01-03	1	1	14
3	2013-01-04	1	1	13
4	2013-01-05	1	1	10
...	...	...	...	...
912995	2017-12-27	10	50	63
912996	2017-12-28	10	50	59
912997	2017-12-29	10	50	74
912998	2017-12-30	10	50	62
912999	2017-12-31	10	50	82

913000 rows × 4 columns

Table 3.1: Original Dataset

The obtained dataset has 913000 rows and 4 columns. Our aim is forecast sales combining all stores and items using the information of 5 years sales.
So, we reduce the dataset which is grouped by date and summing up the corresponding sales. And the resulting dataset has 1826 rows and 2 columns namely date and sales

	date	sales
0	2013-01-01	13696
1	2013-01-02	13678
2	2013-01-03	14488
3	2013-01-04	15677
4	2013-01-05	16237
...	...	...
1821	2017-12-27	20378
1822	2017-12-28	21885
1823	2017-12-29	23535
1824	2017-12-30	24988
1825	2017-12-31	26420

1826 rows × 2 columns

Table 3.2: Transformed Dataset (Used for models)

With the transformed dataset, we have the opportunity to apply time series forecasting techniques to analyze the sales patterns and make predictions for future sales. The dataset provides a valuable opportunity to explore to apply one-step time forecasting and find underlying patterns in sales and their impact on sales. By leveraging this information, we aim to develop accurate and reliable forecasting models that can assist in making informed decisions.

- 3.2 Description and Information about the dataset:
  Description:
  Descriptive statistics of a dataset, specifically for numerical columns is given by describe () function. It computes various statistical measures such as count, mean, standard deviation, minimum, quartiles, and maximum for each numerical column in the dataset.

  The descriptive statistics of original dataset:

	store	item	sales
count	913000.000000	913000.000000	913000.000000
mean	5.500000	25.500000	52.250287
std	2.872283	14.430878	28.801144
min	1.000000	1.000000	0.000000
25%	3.000000	13.000000	30.000000
50%	5.500000	25.500000	47.000000
75%	8.000000	38.000000	70.000000
max	10.000000	50.000000	231.000000

Table 3.3: Descriptive Statistics of Original Dataset

The descriptive statistics of transformed dataset (used for models):

	sales
count	1826.000000
mean	26125.143483
std	6418.270181
min	11709.000000
25%	21195.000000
50%	25839.500000
75%	30779.500000
max	44936.000000

Table 3.4: Descriptive Statistics of Original Dataset

Information:

Information of the dataset is obtained using info () method in pandas provides a concise summary of a DataFrame. It gives information about the dataset, including the total number of rows, the number of non-null values in each column, the data type of each column, and the memory usage.

Information of original dataset:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 913000 entries, 0 to 912999
Data columns (total 4 columns):
 #   Column  Non-Null Count   Dtype
---  ------  --------------   -----
 0   date    913000 non-null  datetime64[ns]
 1   store   913000 non-null  int64
 2   item    913000 non-null  int64
 3   sales   913000 non-null  int64
dtypes: datetime64[ns](1), int64(3)
memory usage: 27.9 MB
```

Table 3.5: Information of Original Dataset

Information of transformed dataset:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1826 entries, 0 to 1825
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   date    1826 non-null   datetime64[ns]
 1   sales   1826 non-null   int64
dtypes: datetime64[ns](1), int64(1)
memory usage: 28.7 KB
```

Table 3.6: Information of Transformed Dataset

- 3.3 Exploratory Data Analysis (EDA):
  - Daily sales: This plot is to visualize the daily sales over the years.

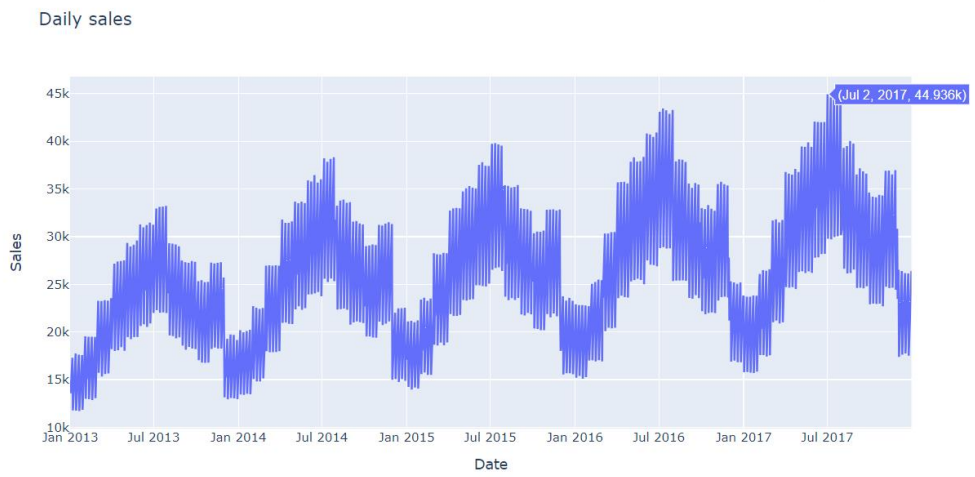


Fig 3.7: Daily sales plot

Inference: There is visible growth of sales over the years. For every year the peak sales time is around mid-year that is July. This pattern is followed for all the years

  - Sales of each store across the year: This plot is to visualize the sales of each store across 5 years. It helps in comparing 3 variables Date, Sales and Store.

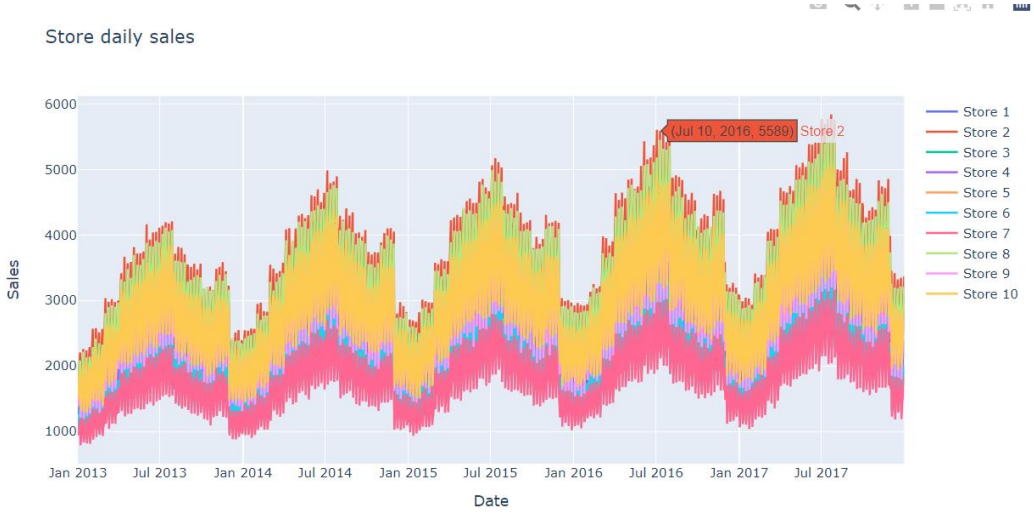


Fig 3.8: Sales of each store across the yea

Inference: There is visible growth for all the stores over the years. The highest sales are observed for the Store 2 and the lowest for Store 7. For all the stores the peak sales time is around mid-year (July)

- Daily Sales for Each item in the store: This plot is to visualize the sales of each items in all the stores over the years. It helps in comparing 3 variables Date, Sales and Items

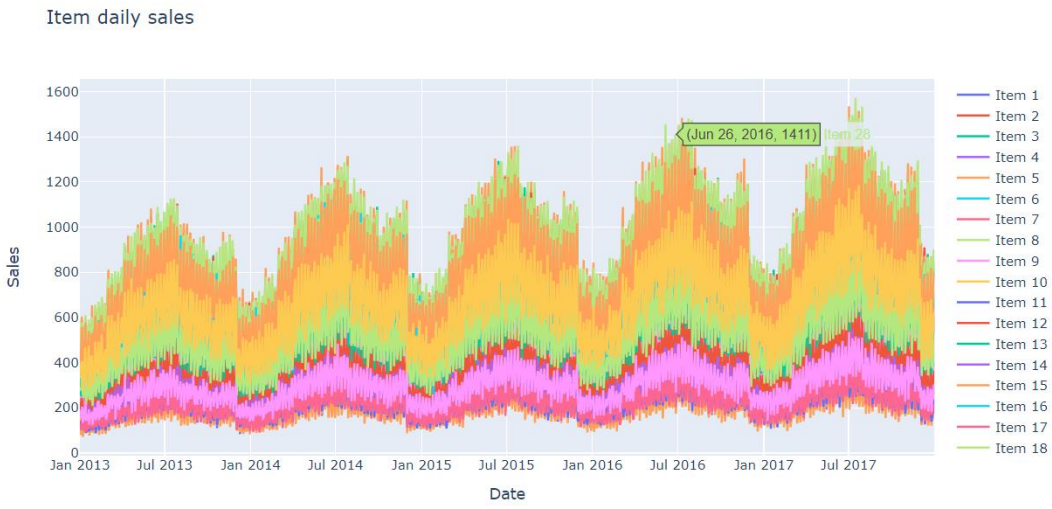


Fig 3.9: *Daily Sales for Each item in the store*

Inference: There is visible growth of sales for all the items over the years. The highest sales are observed for the Item 28 and Item 15 and lowest for Item 5 and Item 1

- Correlation: A correlation plot using a heatmap represents the correlation between variables in a dataset. It provides a visual representation of the pairwise correlations between different variables by using colors to indicate the strength and direction of the correlation.

Fig 3.10: *Heatmap for finding correlation*

Inference: We can clearly see there is no much correlation between variables Sales, Item and Store.

➢ Trend, Seasonal plot:

Trend plot: A trend plot shows the overall pattern or direction of change in a dataset over time. It helps identify the long-term behavior or trend exhibited by the data. By examining the trend plot, we can observe whether the variable's values are generally increasing, decreasing, or remaining stable over time. It provides insights into the overall direction and magnitude of the change.

Seasonal plot: A seasonal plot is a visualization technique that helps identify repeating patterns or seasonal variations in a time series dataset. It focuses on capturing the regular and predictable fluctuations that occur at fixed intervals within a given time period. By examining the seasonal plot, we can observe the recurring patterns or cycles exhibited by the data.

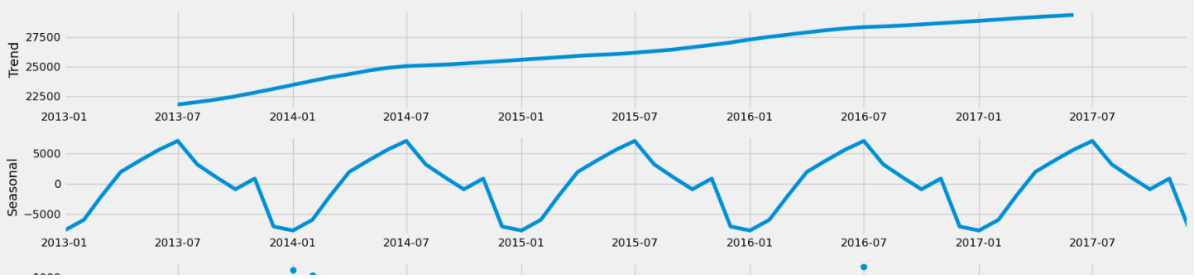


Fig 3.11: *Trend and Seasonal plot*

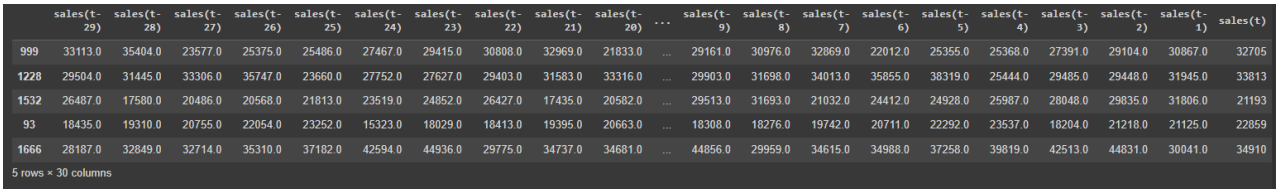
Inference:

1. By the Trend plot we can clearly see that the sales are gradually increased over the time.
2. By the Seasonal plot we can observe that there is repeated and consistent cycle/pattern followed for all the years for the sales. There is gradual increase in sales during early period of the year (January to June) and the peak is at mid-year i.e., July and again there is visible decrease of sales over the time. The next peak after the decrease in sales is observed during November and again the sales are decreasing.

- 3.4 Multistep Forecasting

➢ Series to supervised:
Transforms a time series dataset into a supervised learning problem. It takes the original dataset as input and generates a new dataset with shifted columns representing past time steps as input features and future time steps as target variables to be predicted. The function organizes the data in a tabular format suitable for training machine learning models. By specifying the window size and lag, the function determines the number of previous and future time steps to include. The transformed dataset enables the use of traditional supervised learning algorithms for time series forecasting tasks.

Preprocessed data for Multistep forecasting:



	sales(t-29)	sales(t-28)	sales(t-27)	sales(t-26)	sales(t-25)	sales(t-24)	sales(t-23)	sales(t-22)	sales(t-21)	sales(t-20)	...	sales(t-9)	sales(t-8)	sales(t-7)	sales(t-6)	sales(t-5)	sales(t-4)	sales(t-3)	sales(t-2)	sales(t-1)	sales(t)
999	33113.0	35404.0	23577.0	25375.0	25486.0	27467.0	29415.0	30808.0	32969.0	21833.0	...	29161.0	30976.0	32869.0	22012.0	25355.0	25368.0	27391.0	29104.0	30867.0	32705
1228	29504.0	31445.0	33306.0	35747.0	23660.0	27752.0	27627.0	29403.0	31583.0	33316.0	...	29903.0	31698.0	34013.0	35855.0	38319.0	25444.0	29485.0	29448.0	31945.0	33813
1532	26487.0	17580.0	20486.0	20568.0	21813.0	23519.0	24852.0	26427.0	17435.0	20582.0	...	29513.0	31693.0	21032.0	24412.0	24928.0	25987.0	28048.0	29835.0	31806.0	21193
93	18435.0	19310.0	20755.0	22054.0	23252.0	15323.0	18029.0	18413.0	19395.0	20663.0	...	18308.0	18276.0	19742.0	20711.0	22292.0	23537.0	18204.0	21218.0	21125.0	22859
1666	28187.0	32849.0	32714.0	35310.0	37182.0	42594.0	44936.0	29775.0	34737.0	34681.0	...	44856.0	29959.0	34615.0	34988.0	37258.0	39819.0	42513.0	44831.0	30041.0	34910

5 rows × 30 columns

Fig 3.4.1: Conversion of time series to supervised problem for multistep forecasting

- Processed using the series_to_supervised function with a window size of 29 and a lag of 90. The series_to_supervised function transforms the dataset into a supervised learning format by shifting the data to create input and target variables. The resulting dataset, series, contains columns representing past observations from time step t-29 to t, as well as future observations from time step t+1 to t+90. The head() function is then used to display the first few rows of the transformed dataset.

Multistep forecasting involves predicting multiple future time steps in a time series, beyond the immediate next step. It captures longer-term trends and dependencies. It uses historical observations as input and generates a sequence of future values. Evaluation metrics assess the accuracy of the entire predicted sequence. Various models like ARIMA, LSTM, or transformer-based models can be used. Multistep forecasting enables better planning and decision-making over an extended time horizon in domains like sales, stock market, and demand forecasting.

- 3.5 Model architecture:
  - MLP:

    Before hyperparameter tuning:
    Trainable parameters: 3,201
    Time taken: 7.311077833175659 seconds

    The MLP (Multi-Layer Perceptron) architecture used in this project is a simple neural network model. It consists of two dense layers. The first layer has 100 units with a rectified linear activation function. The second layer has a single unit, which provides the final output prediction. The model is compiled with the mean squared error loss function and optimized using the Adam optimizer. During training, the model is trained for 40 epochs using the training data and validated using the validation data. The objective is to minimize the mean squared error.

    After hyperparameter tuning:
    Trainable parameters: 32,601
    Time taken: 15.881219625473022 seconds

    The MLP architecture consists of an input layer followed by two hidden layers with 200 and 100 neurons respectively, using the ReLU activation function. Dropout layers with a rate of 0.2 are applied after each hidden layer to prevent overfitting. The output

layer has a single neuron for regression prediction. The model is trained using the mean squared error loss and the Adam optimizer for 100 epochs.

- LSTM

  Before hyperparameter tuning
  Trainable parameters: 10,452
  Time taken: 20.531076669692993 seconds

  The LSTM (Long Short-Term Memory) architecture used in this project is a type of re current neural network (RNN) model. It consists of an LSTM layer with 50 units and a rectified linear activation function. The input shape of the LSTM layer is defined ba sed on the dimensions of the training data. A dense layer with a single unit is added af ter the LSTM layer to provide the final output prediction. The model is compiled with the mean squared error loss function and optimized using the Adam optimizer. The su mmary of the model shows the layers and the number of parameters. During training, the model is trained for 40 epochs using the training data and validated using the valid ation data. The goal is to minimize the mean squared error.
  After hyperparameter tuning:
  Trainable parameters: 32,601
  Time taken: 7.352673768997192 seconds

  The LSTM architecture consists of an input layer followed by two fully connected (dense) layers with 200 and 100 neurons respectively, using the ReLU activation function. The output layer has a single neuron for regression prediction. The model is trained using the mean squared error loss and the Adam optimizer with a learning rate of 0.0003. The model is trained for 50 epochs using a batch size of 256.

- CNN

  Before hyperparameter tuning:
  Trainable parameters: 45,093
  Time taken: 8.065096616744995 seconds

  The CNN (Convolutional Neural Network) architecture used in this project is designed for time series forecasting. The model consists of a convolutional layer with 64 filters and a kernel size of 2, followed by a rectified linear activation function. The input shape of the convolutional layer is defined based on the dimensions of the training data. A max pooling layer with a pool size of 2 is added to downsample the output. The flattened output is then passed through a dense layer with 50 units and a rectified linear activation function. Finally, a dense layer with a single unit is added for the final prediction. The model is compiled with the mean squared error loss function and optimized using the Adam optimizer. The summary of the model provides information about the layers and the number of parameters. During training, the model is trained for 40 epochs using the training data and validated using the validation data. The goal is to minimize the mean squared error.

After hyperparameter tuning:
Trainable parameters: 186,067
Time taken:9.190807580947876 seconds

The CNN architecture consists of a convolutional layer with 64 filters and a kernel size of 3, using the ReLU activation function. Max pooling with a pool size of 2 is applied to reduce the spatial dimensions. The output is then flattened and passed through a dense layer with 100 units and the ReLU activation function. Finally, a dense layer with a single neuron is used for regression prediction. The model is compiled with the mean squared error loss and the Adam optimizer with a learning rate of 0.01.
During training, the model is trained for 40 epochs with a batch size of 32. The training data (X_train_series) and corresponding labels (Y_train) are used for training, and the validation data (X_valid_series) and labels (Y_valid) are used for validation. The progress of the training is displayed with a verbosity level of 2.

- CNN-LSTM
  Before hyperparameter tuning: 99,979
  Time taken: 11.878673315048218 seconds

  The CNN-LSTM architecture combines both convolutional and LSTM layers for time series forecasting. The model begins with a TimeDistributed layer that applies a 1D convolutional operation with 64 filters and a kernel size of 1 to each time step of the input sequence. This is followed by a TimeDistributed max pooling layer to reduce the spatial dimension. The output is then flattened and passed through an LSTM layer with 50 units to capture temporal dependencies. Finally, a dense layer with a single unit is added for prediction. The model is trained using the mean squared error loss and the Adam optimizer for 40 epochs.

  After hyperparameter tuning:
  Trainable parameters: 219,829
  Time taken: 30.72464084625244 seconds

  The CNN-LSTM architecture combines a 1D convolutional neural network (CNN) and a long short-term memory (LSTM) model. It consists of a convolutional layer, pooling layer, and flattening layer followed by an LSTM layer with dropout. The output is passed through a dense layer for regression prediction. The model is trained using mean squared error (MSE) loss and the Adam optimizer.

Note: For all the neural network model the hyperparameter tuning is done by grid search method separately to find the best parameter and took those parameters alone to train the model. So, during any errors we don't have to re-run all combinations of parameters again, which is time consuming.

- ARIMA

  Time taken: 10.627931118011475 seconds

  The ARIMA architecture stands for Autoregressive Integrated Moving Average. In this code snippet, different combinations of parameters for the seasonal ARIMA model are evaluated to find the optimal model for the given time series data.
  The code initializes the ranges for the parameters p, d, and q, and creates a list of all possible combinations of these parameters using the itertools.product function. It also generates seasonal parameter combinations (seasonal_pdq) by combining the same ranges with a seasonal period of 12.
  The code then iterates over each parameter combination and tries to fit a SARIMAX model to the time series data. The SARIMAX model is created with the specified order and seasonal order parameters, while enforcing stationarity and invertibility constraints.
  For each parameter combination, the Akaike Information Criterion (AIC) is calculated as a measure of the model's goodness of fit. The AIC values are printed for each combination to evaluate the performance of different models.
  Finally, the execution time of the code is measured using the time module, starting before the parameter search loop and ending after the loop. The elapsed time is calculated and printed as the execution time in seconds.

4.RESULT:

- 4.1 Forecasting using MLP model:

  - Before hyperparameter tuning:

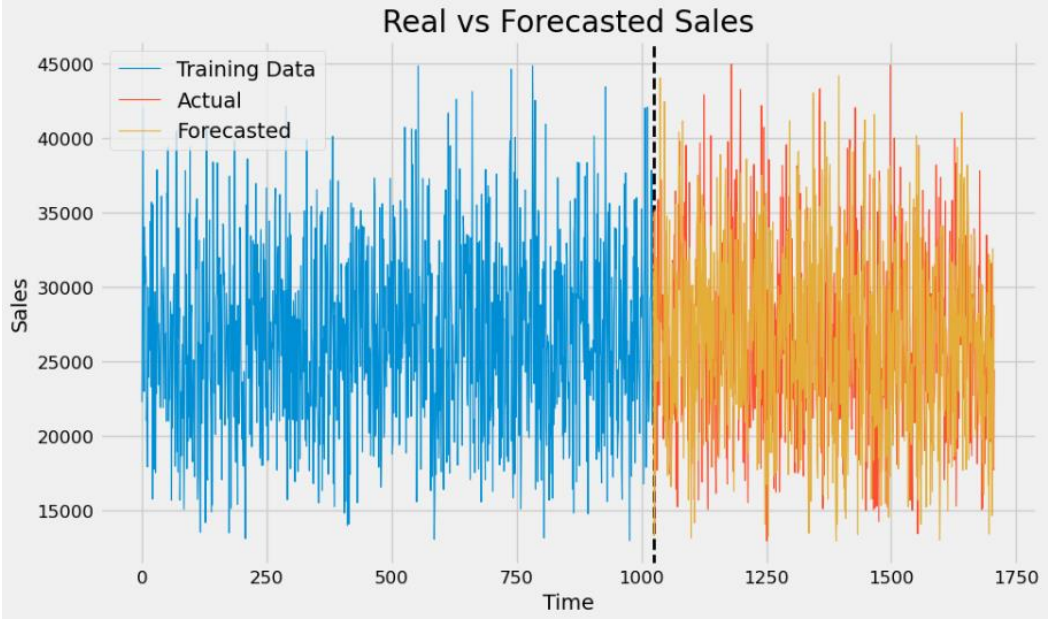


Fig 4.1.1: *Forecasting of sales using MLP model (before hyperparameter tuning)*

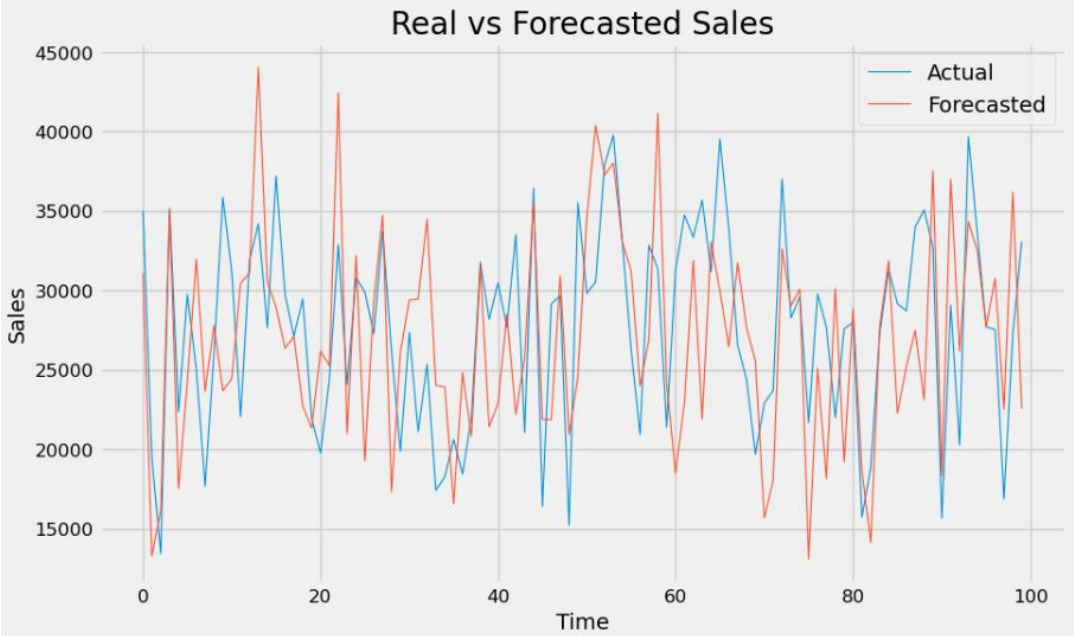


Fig 4.1.2: *Clear visualization of actual vs forecasted sales (1st 100 datapoints)*

- After hyperparameter tuning:

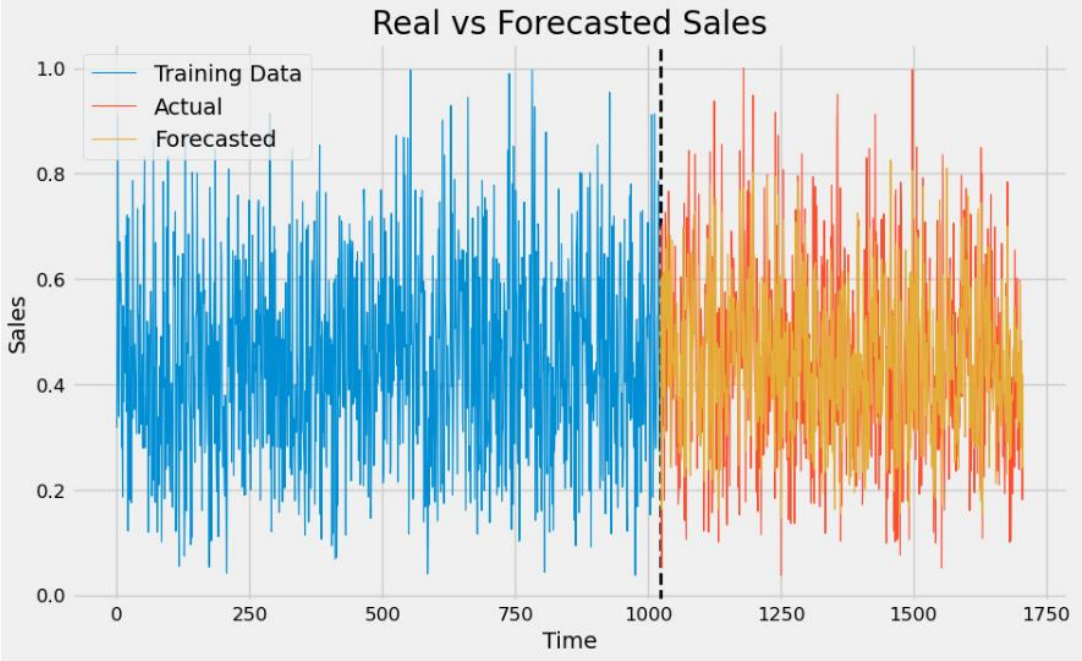


Fig 4.1.3: *Forecasting of sales using MLP model (after hyperparameter tuning)*

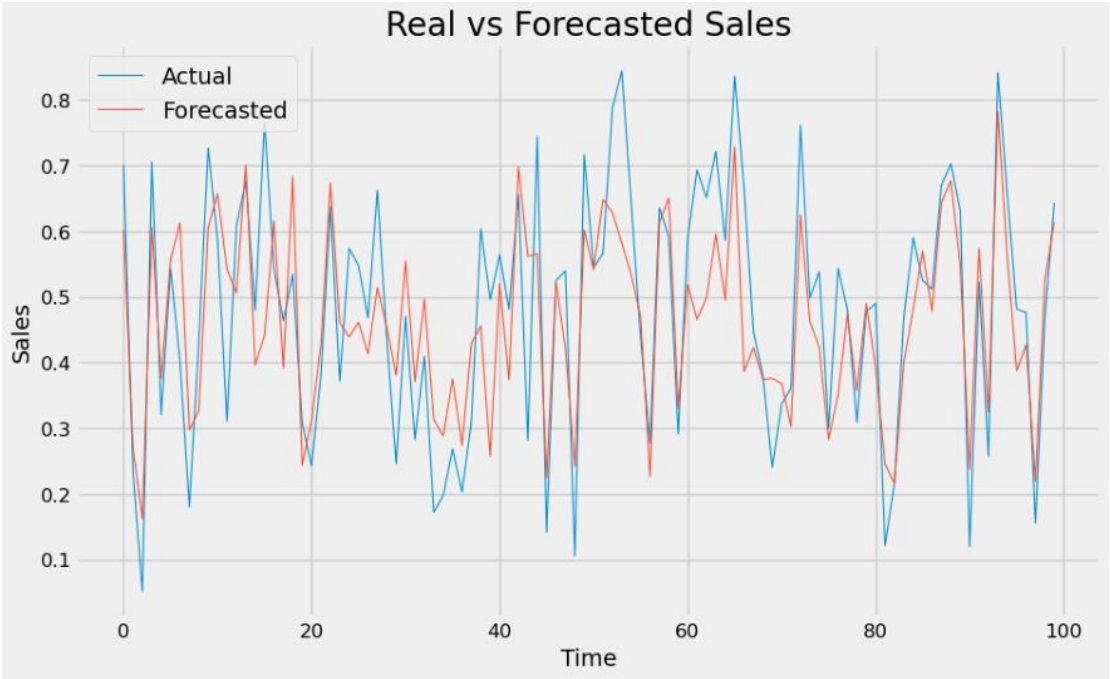


Fig 4.1.4: *Clear visualization of actual vs forecasted sales (1st 100 datapoints)*

- 4.2 Forecasting using LSTM:

  - Before hyperparameter tuning:

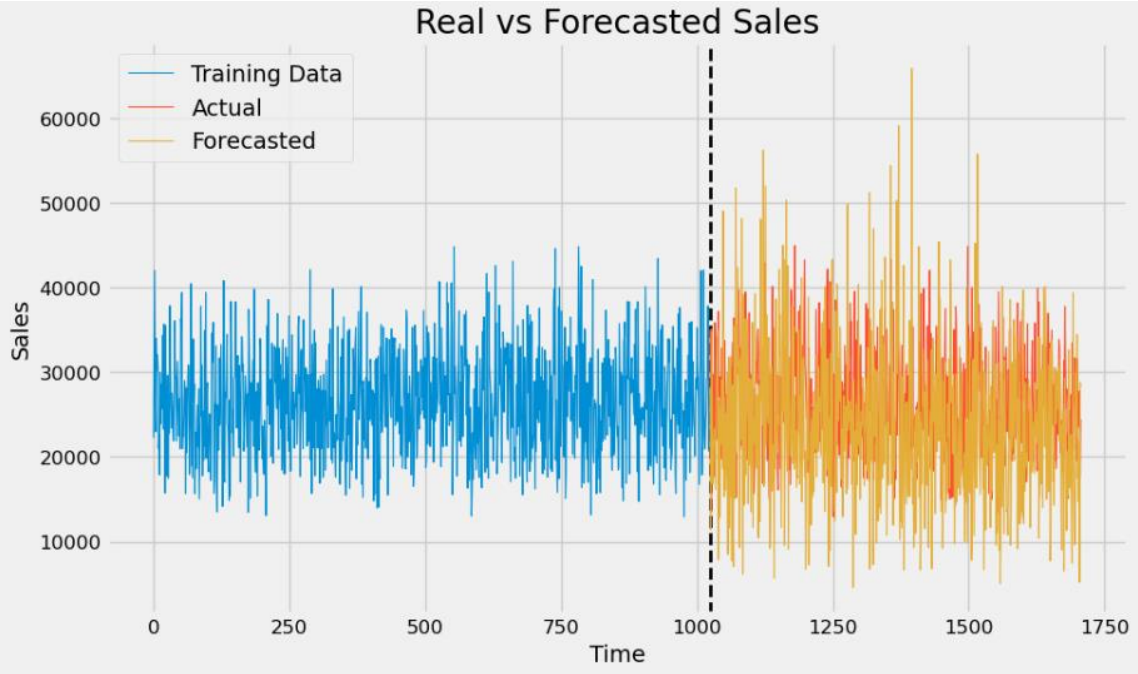


Fig 4.2.1: *Forecasting of sales using LSTM model (before hyperparameter tuning)*

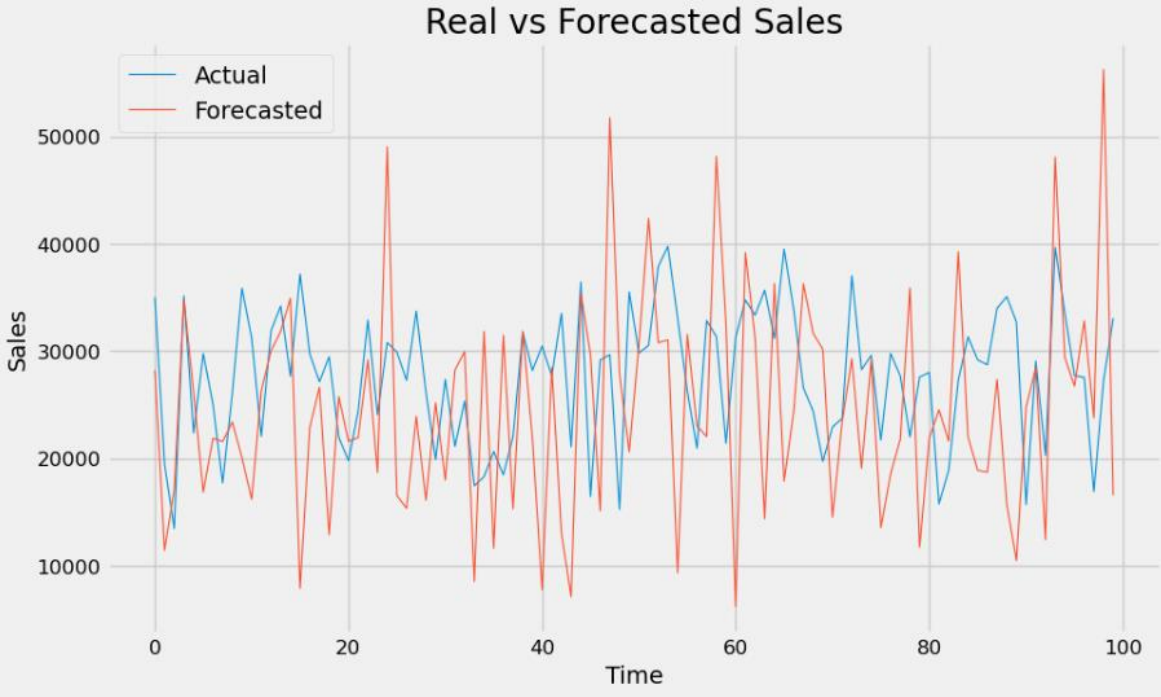


Fig 4.2.2: Clear visualization of actual vs forecasted sales (1st 100 datapoints)

- After hyperparameter tuning

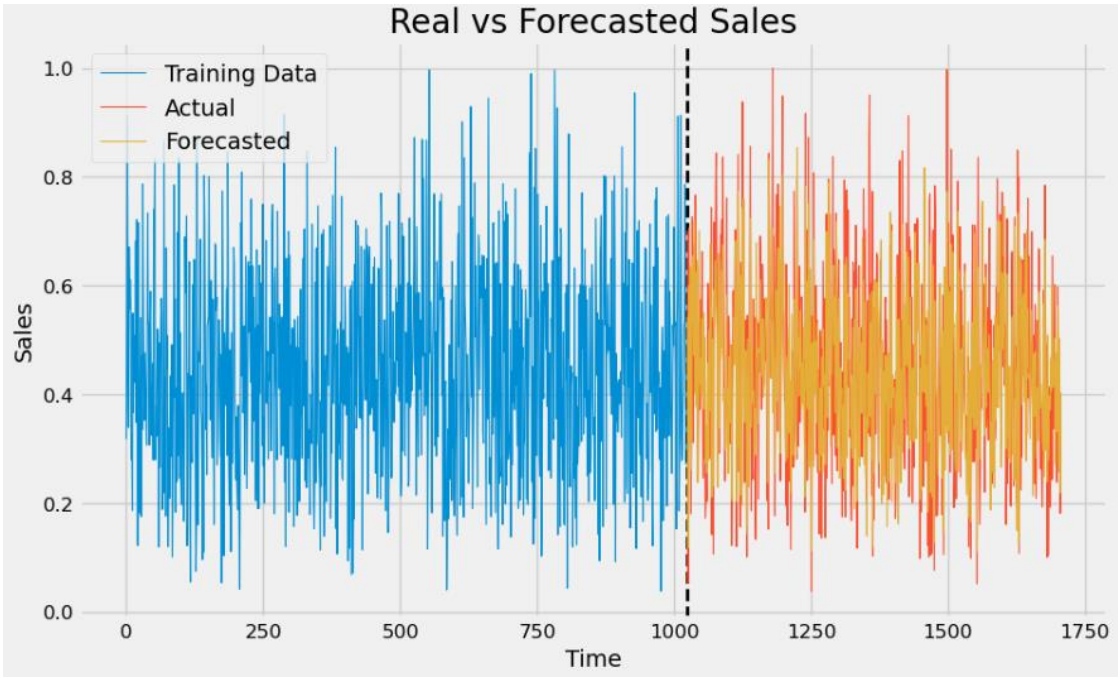


Fig 4.2.3: *Forecasting of sales using LSTM model (after hyperparameter tuning)*

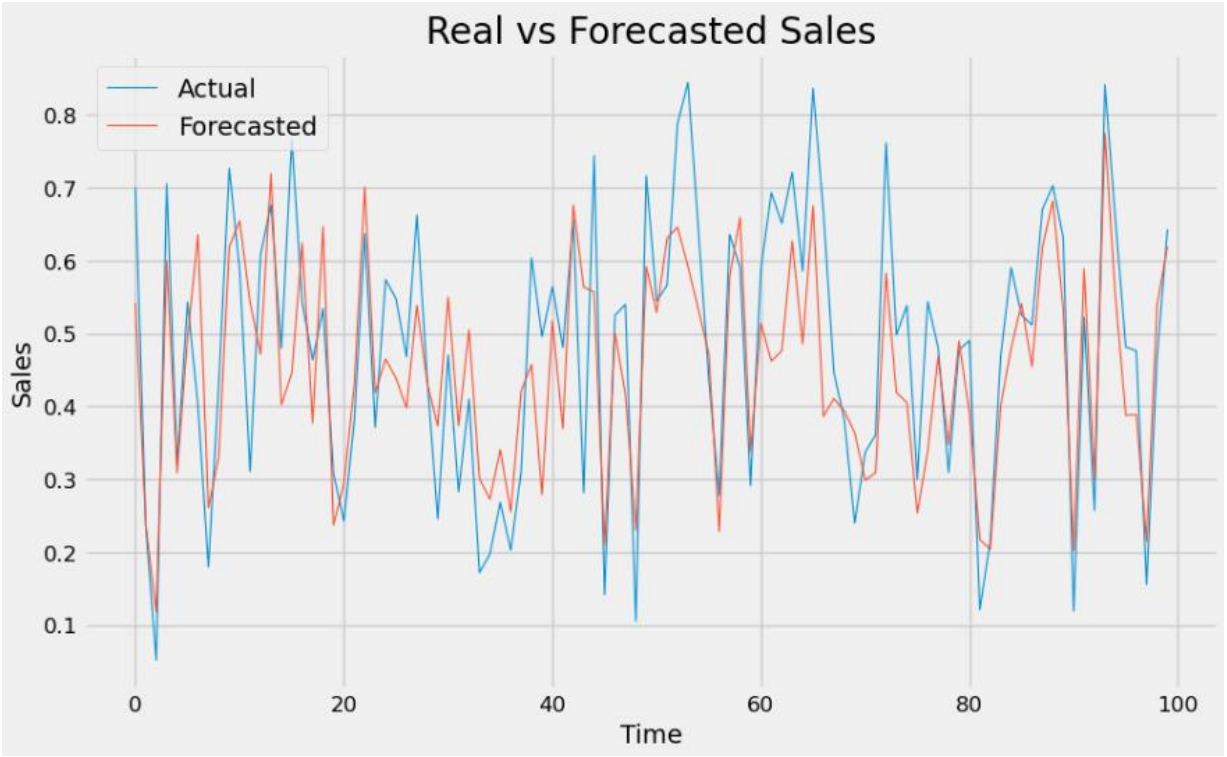


Fig 4.2.4: *Clear visualization of actual vs forecasted sales (1st 100 datapoints)*

- 4.3 Forecasting using CNN:

  - Before hyperparameter tuning:

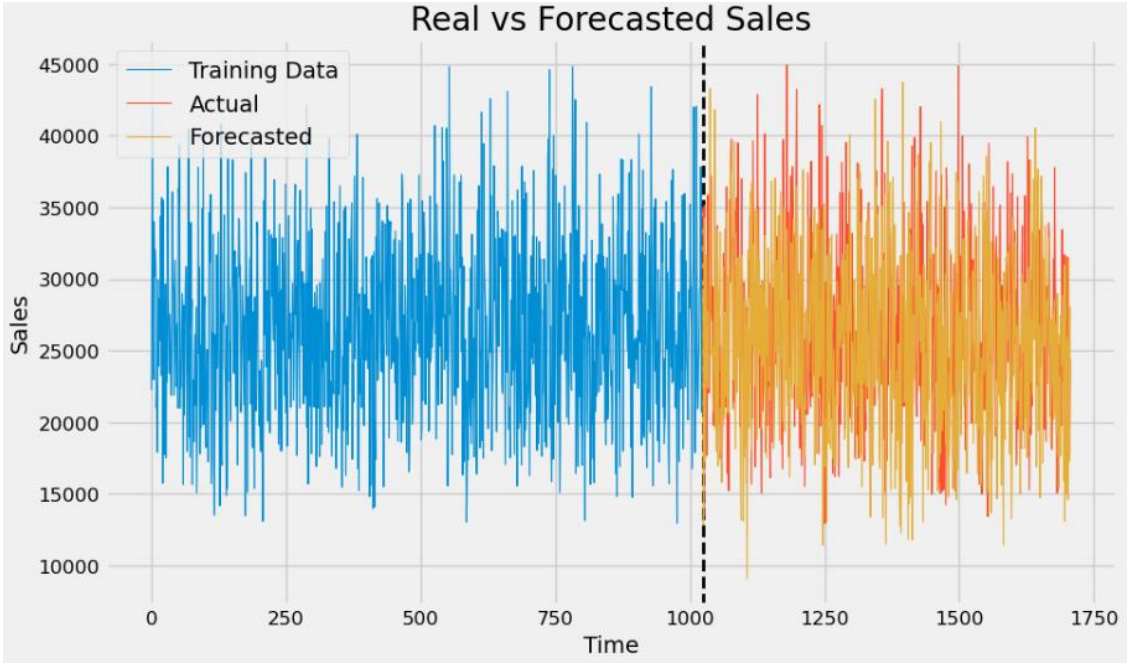


Fig 4.3.1: *Forecasting of sales using CNN model (before hyperparameter tuning)*

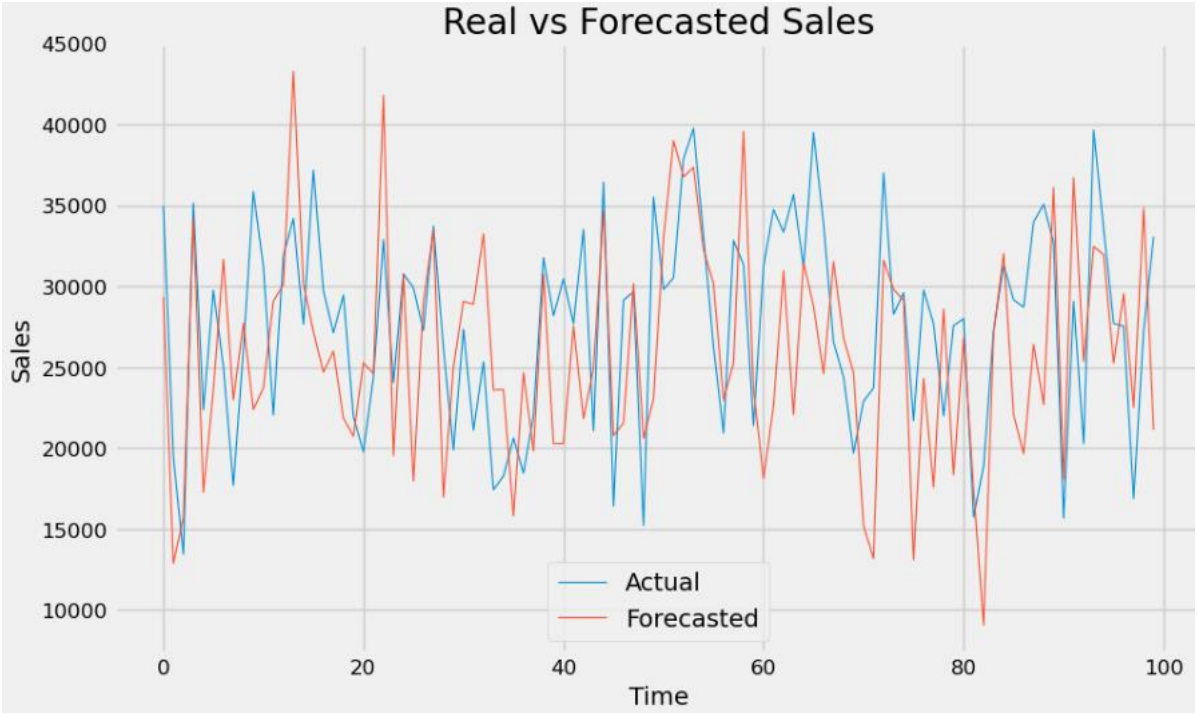


Fig 4.3.2: *Clear visualization of actual vs forecasted sales (1st 100 datapoints)*

- After hyperparameter tuning

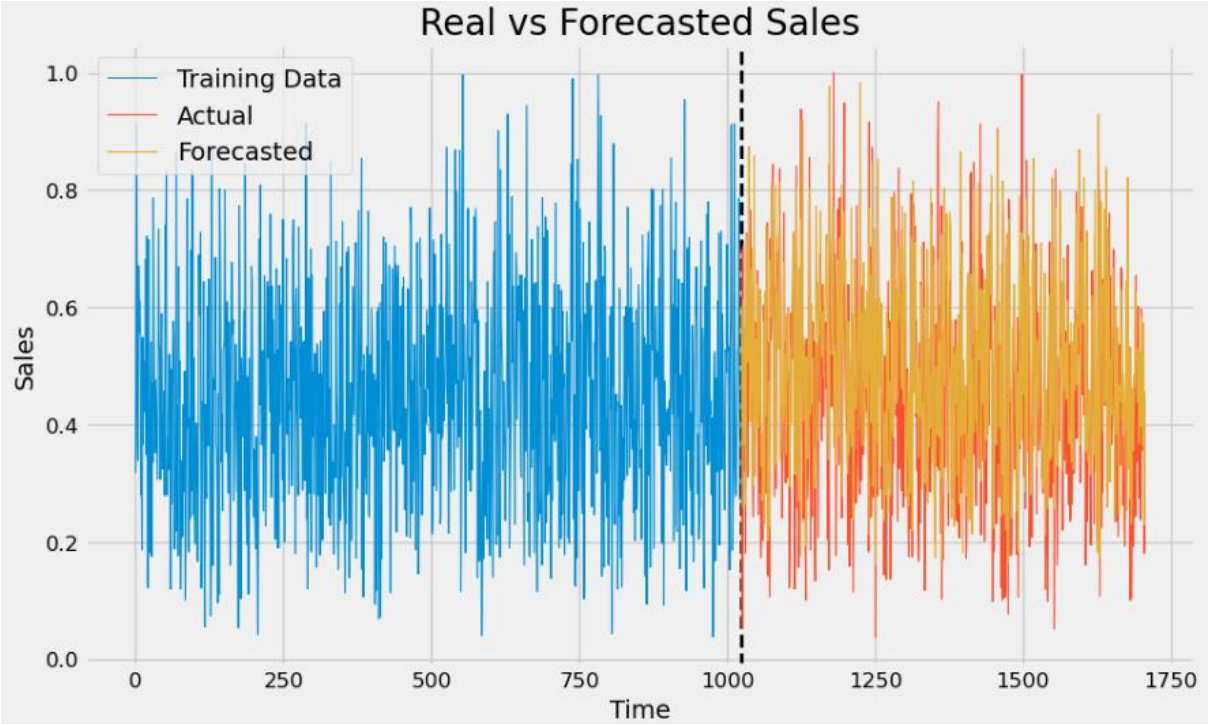


Fig 4.3.3: *Forecasting of sales using CNN model (after hyperparameter tuning)*

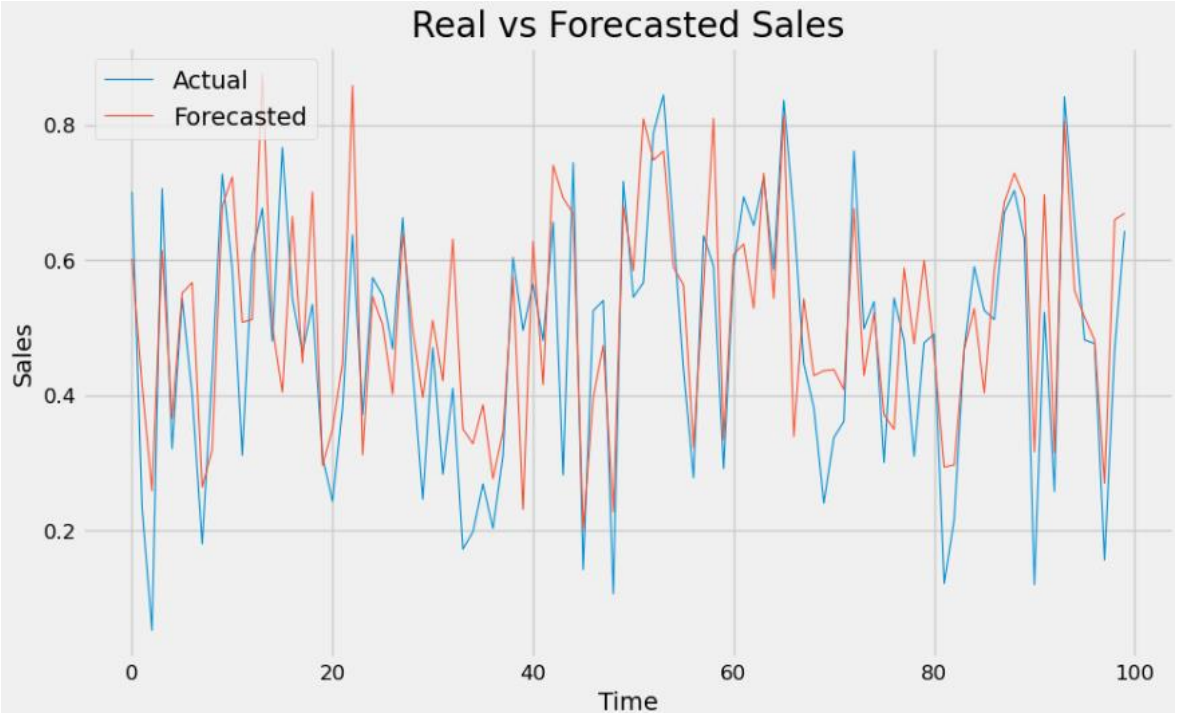


Fig 4.3.4: *Clear visualization of actual vs forecasted sales (1st 100 datapoints)*

- 4.4 Forecasting using CNN-LSTM

  - Before hyperparameter tuning

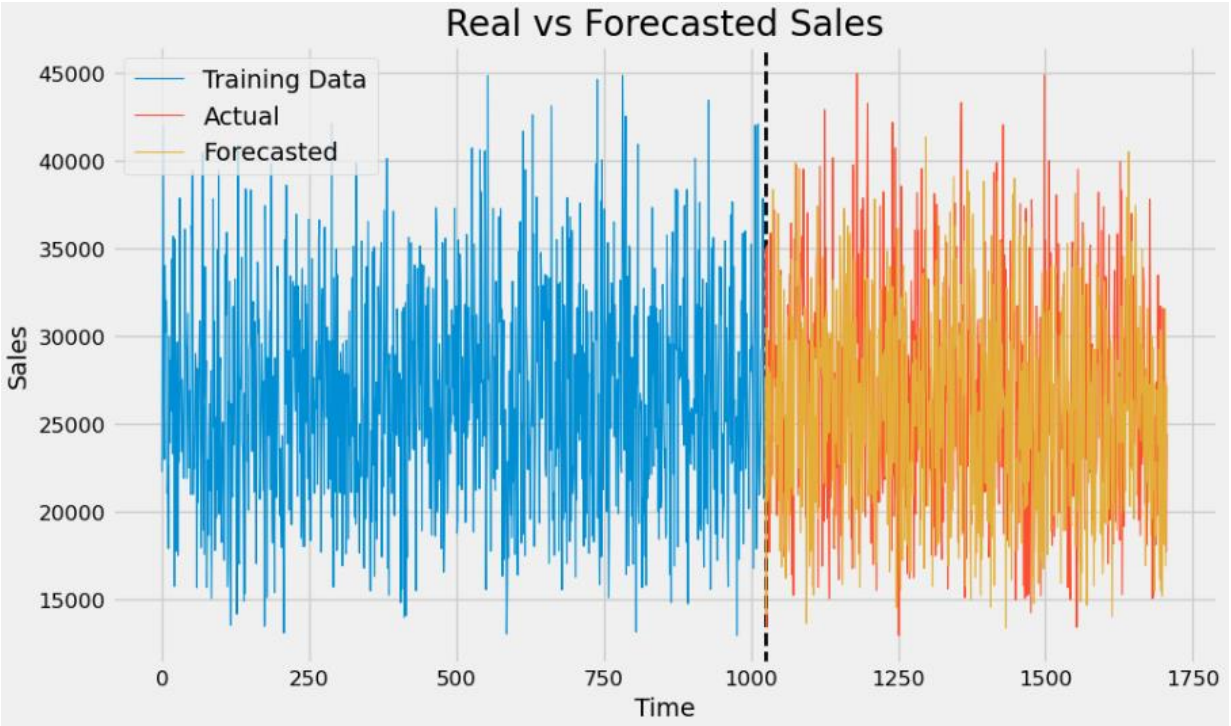


Fig 4.4.1: *Forecasting of sales using CNN-LSTM model (before hyperparameter tuning)*

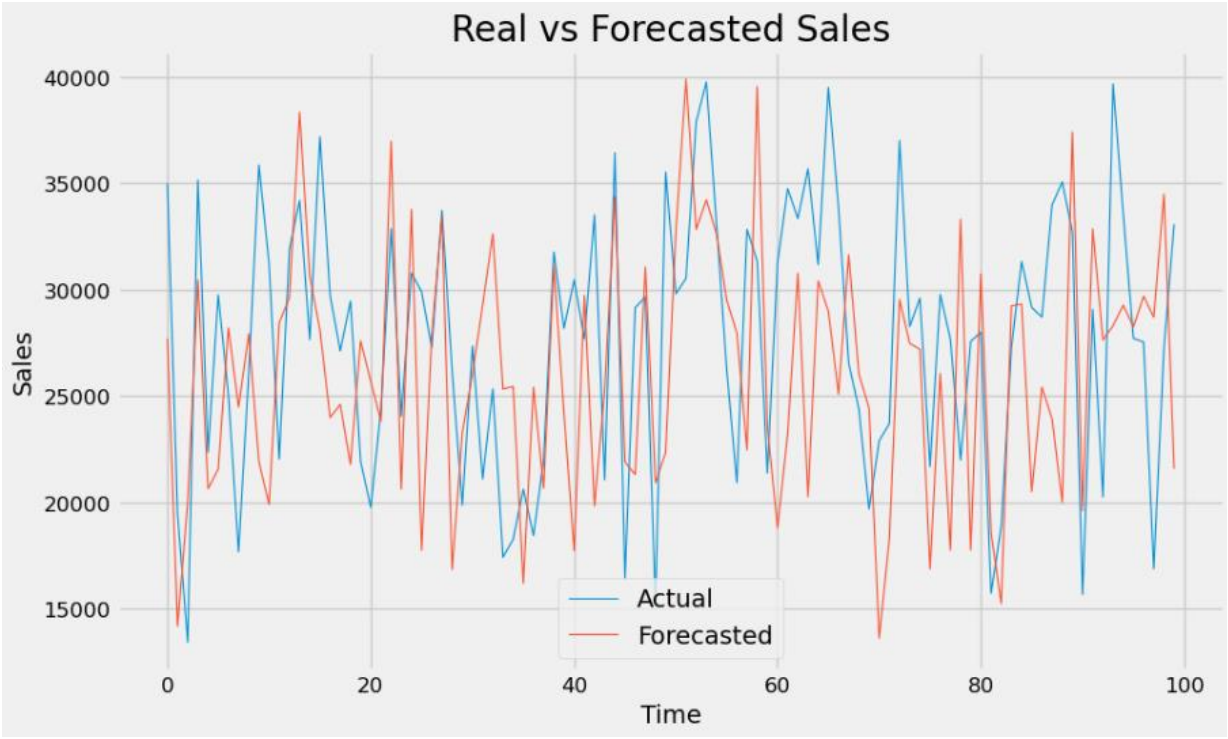


Fig 4.4.2: *Clear visualization of actual vs forecasted sales (1st 100 datapoints)*

- After hyperparameter tuning:

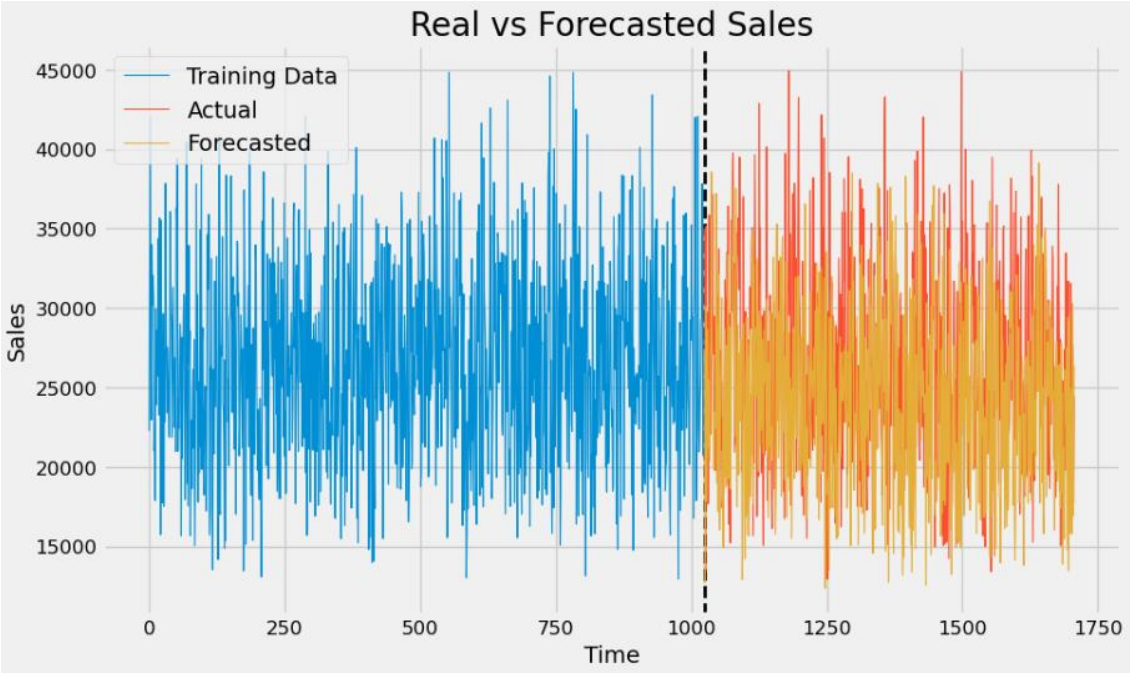


Fig 4.4.3: *Forecasting of sales using CNN-LSTM model (after hyperparameter tuning)*

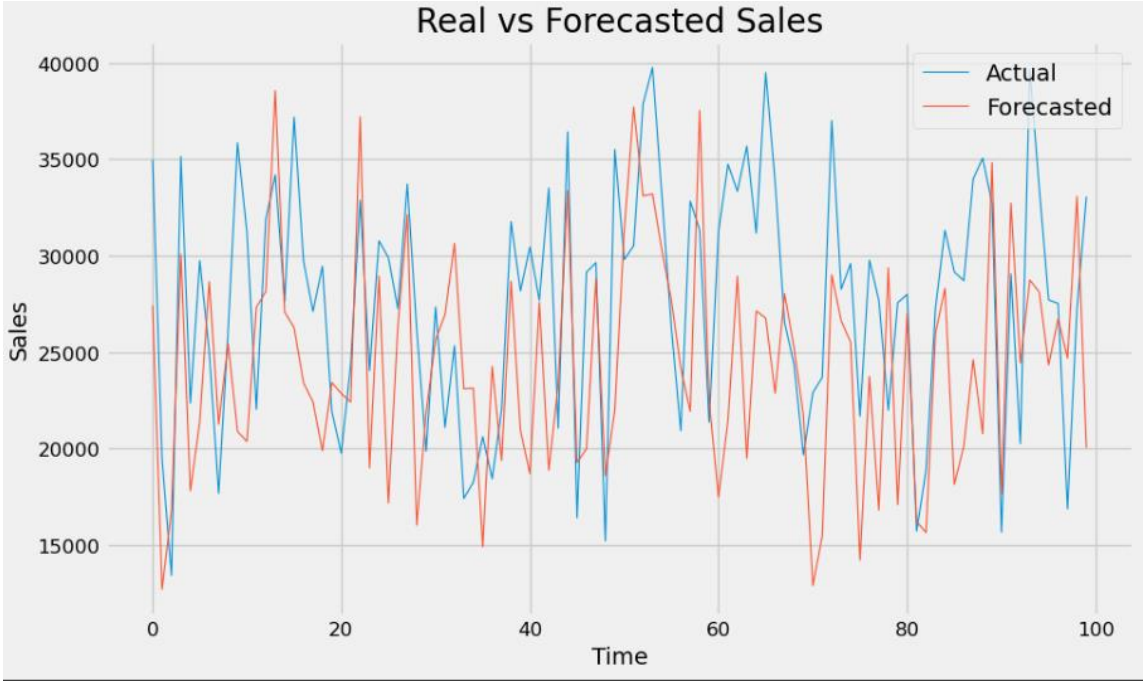


Fig 4.4.4: *Clear visualization of actual vs forecasted sales (1st 100 datapoints)*

- 4.5 Forecasting using ARIMA model
  - ➢ Forecast plot (observed vs forecasted sales)

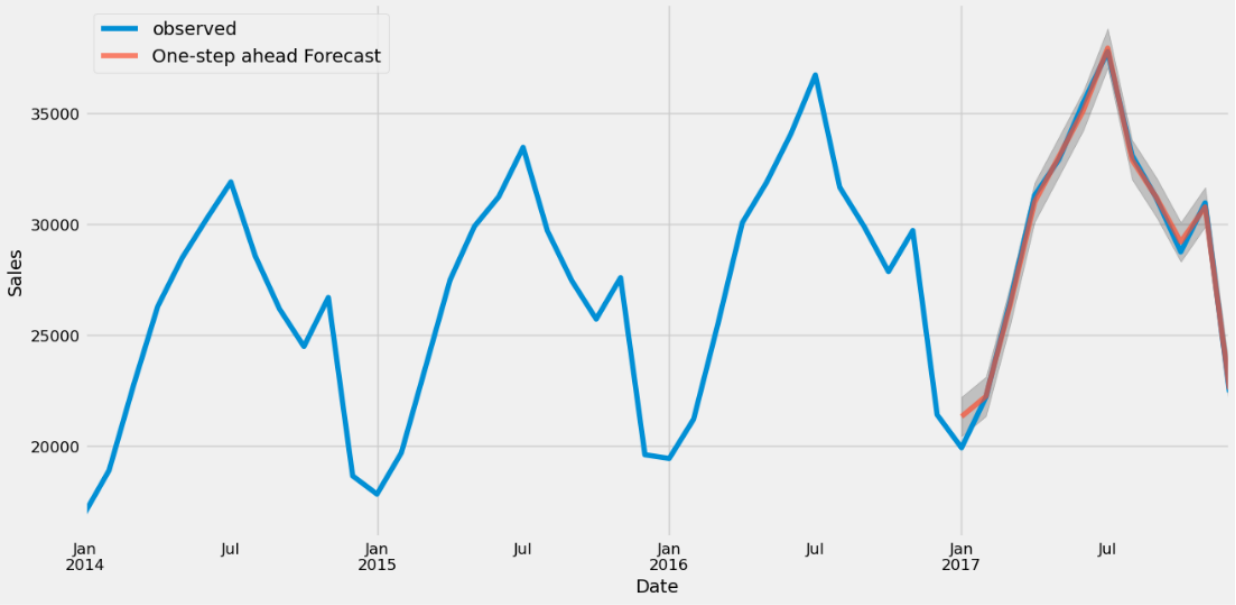


Fig 4.5.1: *Forecasting of sales using ARIMA model*

➢ Residual: A residual plot is a graphical representation of the differences between the observed values and the corresponding predicted values in a regression or time series analysis. It helps us assess the goodness of fit of a model and detect any patterns or systematic deviations in the residuals.

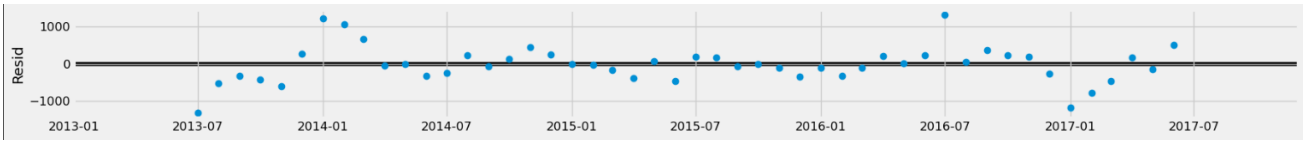


Fig 4.5.2 Residual plot for ARIMA model

By Residual plot there is some errors in predicting, but as we don't observe any patterns in the plot, we can assume that the model is capturing the underlying patterns in the data

➢ Standardized Residual, Histogram Plus Estimated Density, Normal Q-Q, Correlogram:

Standardized Residual: Standardized Residual Plot: This plot shows the standardized residuals, which are the residuals divided by their estimated standard deviation. The plot helps us assess if the residuals have constant variance and are randomly distributed around zero. If the standardized residuals fluctuate within a narrow range around zero, it indicates a good fit. However, patterns or systematic deviations in the plot suggest that the model may not adequately capture the underlying patterns in the data.

Histogram Plus Estimated Density: This plot shows the histogram of the residuals along with the estimated density curve. It provides insights into the distribution of the residuals. A symmetric and bell-shaped histogram with the estimated density curve closely matching the histogram suggests that the residuals follow a normal distribution, which is desirable for a well-fitted model.

Normal Q-Q (Quantile-Quantile) Plot: The Normal Q-Q plot compares the quantiles of the residuals to the quantiles of a theoretical normal distribution. If the residuals follow a normal distribution, the points in the plot should approximately fall along a straight line. Deviations from the straight line suggest departures from normality.

Correlogram (Autocorrelation Function Plot): This plot shows the autocorrelation function (ACF) of the residuals. It helps us assess if there is any remaining correlation or patterns in the residuals after accounting for the model's predictions. If the autocorrelation values fall within the blue shaded region (confidence interval), it indicates no significant autocorrelation in the residuals.

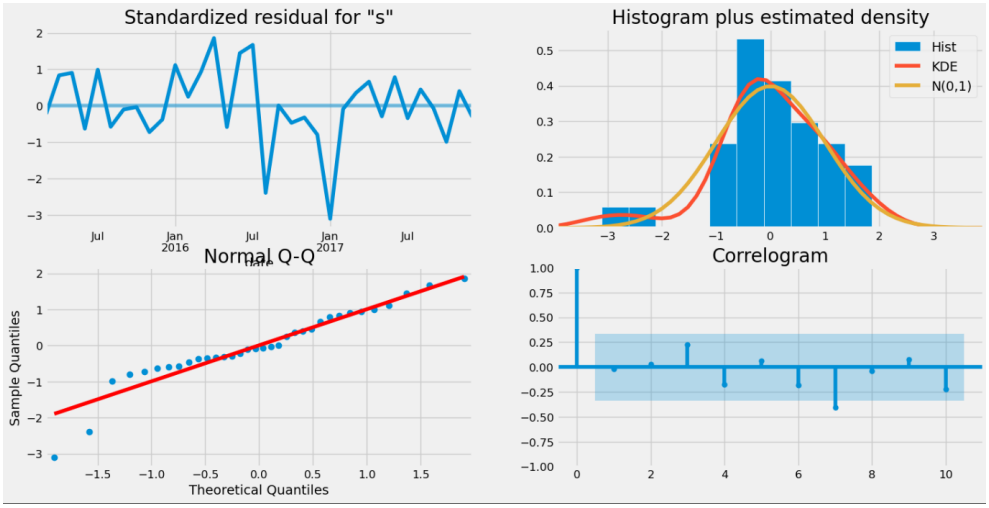


Fig 4.5.3: Standardized Residual, Histogram Plus Estimated Density, Normal Q-Q, Correlogram:

If the standardized residuals show no clear pattern, fluctuate around zero, and have a constant variance, it suggests a good fit.

If the histogram plus estimated density resembles a normal distribution, it indicates that the residuals follow a normal distribution.

If the points in the Normal Q-Q plot fall approximately along a straight line, it suggests that the residuals are normally distributed.

If the autocorrelation values in the correlogram are within the confidence interval, it indicates that there is no significant autocorrelation remaining in the residuals.

- 4.6 LOSS PLOT:

  Before hyperparameter tuning:

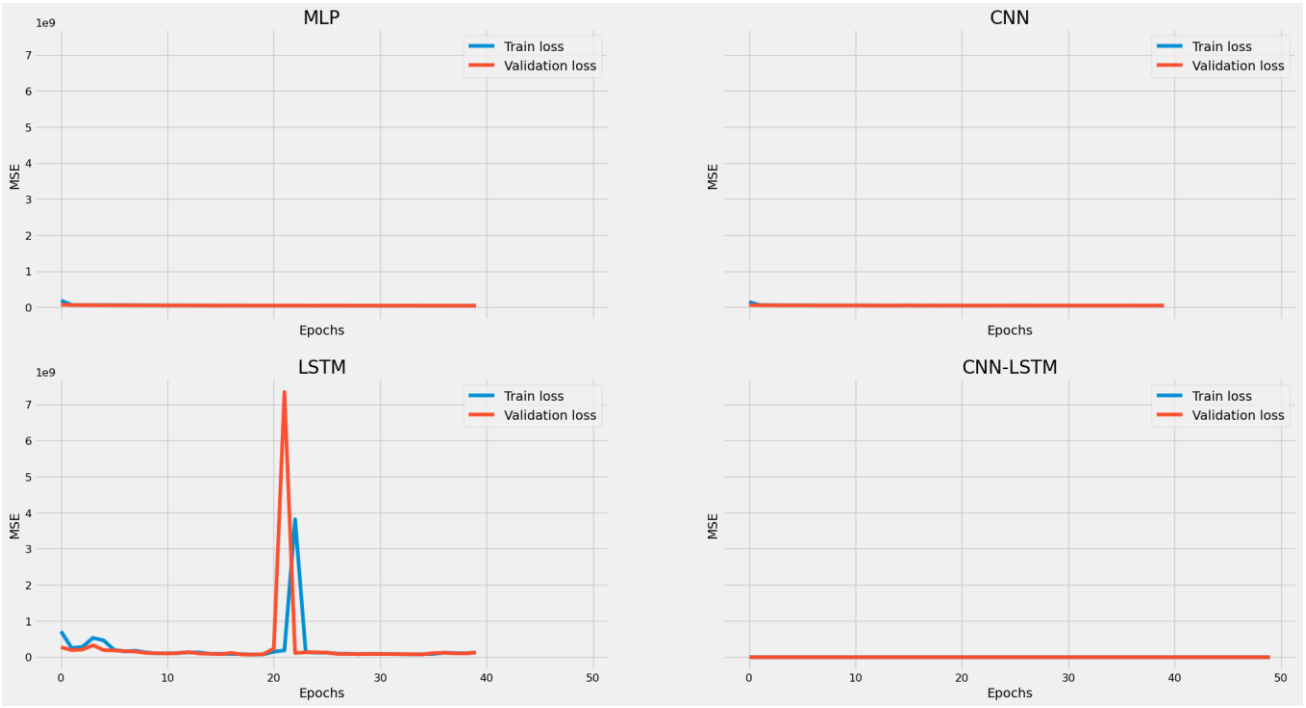


Fig 4.6.1: Loss plot for the neural network models (before hyperparameter tuning) – MLP, CNN, LSTM, CNN-LSTM

  After hyperparameter tuning:

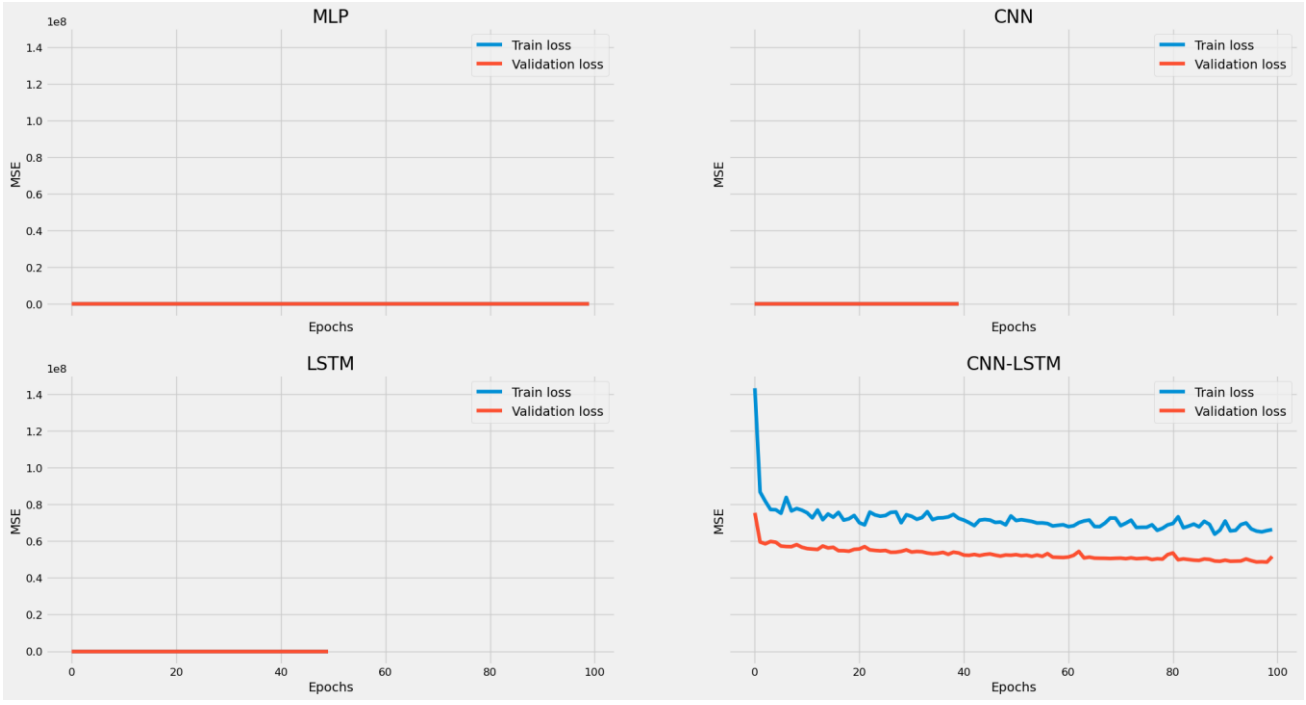


Fig 4.6.2: Loss plot for the neural network models (after hyperparameter tuning) – MLP, CNN, LSTM, CNN-LSTM

- 4.7 MODEL PERFORMANCE AND EVALUATION:

Performance metrics before hyperparameter tuning for neural network models

MODELS/ PERFORMANCE METRICS	MLP		LSTM		CNN		CNN-LSTM	
	TRAIN	*VALIDATION*	*TRAIN*	*VALIDATION*	*TRAIN*	*VALIDATION*	*TRAIN*	*VALIDATION*
RMSE	6537.5	6567.6	10855.17	10936.2	6793.56	6791.4	7384.3	7295.4
MAE	5682.6	5701.1	8505.3	8761.05	5845.5	5701.1	6334.4	6340.6
MAPE	0.2221	0.2214	0.3158	0.3226	0.2253	0.2242	0.2479	0.2438
MSE	42739345	43133710	117834806	119602351.5	46152488.6	46124010.0	54528915.8	53223853
R2 SCORE	-0.112	-0.114	-2.066	-2.0906	-0.2011	-0.1918	-0.4191	-0.3753

Table 4.7.1: Performance metrics table (before hyperparameter tuning)

Performance metrics after hyperparameter tuning for neural network models and ARIMA

MODELS/ PERFORMANCE METRICS	MLP		LSTM		CNN		CNN-LSTM		ARIMA
	TRAIN	*VALIDATION*	*TRAIN*	*VALIDATION*	*TRAIN*	*VALIDATION*	*TRAIN*	*VALIDATION*	
RMSE	0.10486	0.1181	0.10323	0.1177	0.11443	0.1285	6539.4	6393.9	0.0139
MAE	0.0828	0.0961	0.0805	0.09447	0.0922	0.10421	5250.9	5306.9	0.0089
MAPE	0.25164	0.2780	0.2344	0.26207	0.3031	0.3219	0.2125	0.210	0.0240
MSE	0.0109	0.0139	0.0106	0.01387	0.0130	0.0165	41135784.3	40882150	0.0001
R2 SCORE	0.6840	0.6016	0.69376	0.6041	0.62374	0.52833	-0.0705	-0.0564	0.992

Table 4.7.2: Performance metrics table (after hyperparameter tuning)

Time Taken to run the improved model for forecasting (model after hyperparameter tuning):

MODELS	TIME TAKEN (seconds)
MLP	15.881219625473022
LSTM	7.352673768997192
CNN	9.190807580947876
CNN-LSTM	30.72464084625244
ARIMA	9.900813341140747

Table 4.7.3: Time taken to run the improved model (after hyperparameter tuning)

- 4.8 COMPARISON (Strengths and Weaknesses)

MLP Model:

Strengths:

The model has a relatively low RMSE and MAE, indicating good accuracy in predicting the target variable.

The R2 score of 0.6016 suggests that the model explains a significant portion of the variance in the data.

Weaknesses:

The MAPE value of 0.2780 indicates a moderate percentage of error in prediction.

The execution time of 15.8 seconds is relatively longer compared to some other models.

LSTM Model:

Strengths:

The model has low RMSE, MAE, and MAPE values, suggesting accurate predictions with low errors.

The R2 score of 0.6041 indicates a good fit to the data.

The execution time of 7.35 seconds is relatively fast.

Weaknesses:

No significant weaknesses based on the provided metrics.

CNN Model:

Strengths:

The model has a reasonable R2 score of 0.52833, indicating a fair fit to the data.

Weaknesses:

The model has higher RMSE, MAE, MAPE, and MSE values compared to the other models, suggesting higher errors in predictions.

The execution time of 9.19 seconds is relatively longer than the LSTM model.

CNN-LSTM Model:

Strengths:

No significant strengths based on the provided metrics.

Weaknesses:

The model has extremely high values for RMSE, MAE, and MSE, indicating significant errors in predictions.

The R2 score of -0.0564 suggests a poor fit to the data.

The execution time of 30.72 seconds is relatively longer than the other models.

The combination of CNN-LSTM wasn’t suitable for this model

ARIMA Model:

Strengths:

The model has the highest R2 score of 0.992, indicating an excellent fit to the data and explaining a large portion of the variance.

Weaknesses:

The model has very low RMSE, MAE, and MAPE values, suggesting extremely low errors in predictions.

The execution time of 9.9 seconds is relatively longer compared to some other models.

- 4.9 CONCLUSION:

The best models based on performance metrics are ARIMA and LSTM. Detailed explanation of why these models are selected as best is given below:

ARIMA Model:

The ARIMA model demonstrates several strengths that make it a suitable choice for time series forecasting:

High R2 Score: The ARIMA model achieves the highest R2 score of 0.992, indicating an exceptional fit to the data. This suggests that the model captures a significant portion of the variance in the target variable, enabling accurate predictions.

Low Errors: The ARIMA model exhibits remarkably low values for RMSE, MAE, and MAPE, signifying minimal errors in prediction. This implies that the model consistently estimates the target variable with high precision, making it reliable for forecasting purposes.

Statistical Approach: The ARIMA model is a statistical model that utilizes the autoregressive (AR), moving average (MA), and differencing (I) components. This approach leverages the historical patterns, autocorrelation, and stationarity properties of the time series data to generate forecasts. As a result, the ARIMA model can be effective in capturing seasonality, trends, and dependencies within the data.

Interpretable and Explainable: The ARIMA model provides interpretable coefficients and parameters, which can offer insights into the relationship between past observations and future predictions. This transparency enhances the model's interpretability, making it easier to understand and explain the forecasting results.

LSTM Model:
The LSTM model also demonstrates several strengths that position it as a compelling choice for time series forecasting:

Accurate and Precise: The LSTM model exhibits low values for RMSE, MAE, and MAPE, indicating accurate predictions and minimal errors. The model's architecture, which includes memory cells and gates, allows it to capture long-term dependencies and complex patterns within the time series data, leading to accurate forecasts.

Sequential Data Processing: The LSTM model is specifically designed to handle sequential data, making it well-suited for time series forecasting tasks. The model can effectively learn and leverage the temporal relationships and dependencies present in the data, enabling it to make informed predictions based on the historical sequence of observations.

Nonlinear Relationships: Unlike traditional linear models such as ARIMA, the LSTM model is capable of capturing nonlinear relationships between past and future observations. This flexibility is especially advantageous when dealing with complex and nonlinear time series patterns, as the model can capture intricate dependencies that linear models may struggle to represent.

Generalization Ability: The LSTM model has the ability to generalize well on unseen data, making it robust for forecasting tasks. By learning from the entire historical sequence, the LSTM model can effectively extract relevant features and patterns to generate accurate predictions even when faced with varying trends, seasonality, or irregularities in the data.

In summary, both the ARIMA and LSTM models have distinct strengths that make them suitable choices for time series forecasting. The ARIMA model, with its statistical foundation, high R2 score, and interpretability, can be an excellent option for capturing underlying patterns and dependencies in the data. On the other hand, the LSTM model, with its ability to handle sequential data, capture nonlinear relationships, and generalize well, offers accurate predictions even in the presence of complex time series patterns. The choice between the two models depends on factors such as the specific characteristics of the data, the nature of the forecasting problem, and the importance of interpretability versus flexibility in modeling.

5.REFERENCE

- Dataset: https://www.kaggle.com/competitions/demand-forecasting-kernels-only/overview

- **[1] Deep learning for time series forecasting:**
  By Jason Brownlee in Deep Learning for Time Series
  https://machinelearningmastery.com/how-to-get-started-with-deep-learning-for-time-series-forecasting-7-day-mini-course/
  Year: 2018

- **[2] Project on time series analysis and forecasting**
  By Susan Li, in Towards Data Science
  https://towardsdatascience.com/an-end-to-end-project-on-time-series-analysis-and-forecasting-with-python-4835e6bf050b
  Year: 2018

- **[3] Predicting sales: Time series analysis and forecasting in python**
  By Bisman Preet Singh, published in Analytics Vidhya
  https://medium.com/analytics-vidhya/predicting-sales-time-series-analysis-forecasting-with-python-b81d3e8ff03f
  Year: 2020

- **[4] Time series forecasting of seasonal item sales using ML**
  By Yasaman Ensafi [a], Saman Hassanzadeh Amin [a], Guoqing Zhang [b], Bharat Shah [c]
  https://doi.org/10.1016/j.jjimei.2022.100058
  Year: 2022