|  |
| --- |
| Photo displaying partial image of two pie charts on a canvas-textured page |
| Time Series Models |
| |  |  |  | | --- | --- | --- | | 22334905 & 22303142 |  | TECHNICAL PROGRAMMING | |

# Analysing Time Series Models: ARIMA, LSTM, and Facebook Prophet

## Making Data Stationary and Choosing Parameters(ARIMA MODEL)

For ARIMA models, the data must be stationary, meaning that its statistical properties like mean, variance, and autocorrelation are consistent over time. In the Model we used Augmented Dickey-Fuller test for stationarity

* result = adfuller(df['Sales'])   
  print('ADF Statistic:', result[0])   
  print('p-value:', result[1])

Choosing Model Parameters (p, d, q):

* **p (AR Term):** The number of lag observations included in the model (Autoregressive part). It can be determined using the **Partial Autocorrelation Function (PACF)** plot, which shows correlations between an observation and its lagged values.
* **d (Differencing):** The number of differences needed to make the series stationary, determined via tests like ADF.
* **q (MA Term):** The number of lagged forecast errors in the prediction (Moving Average part). This can be identified using the **Autocorrelation Function (ACF)** plot, which shows the correlation between the data and lagged versions of itself.

In the model we used the code underneath to plot the values because p value was less than 0.005.  
 plot\_acf(sales\_data)

plt.title('ACF Plot')

plt.show()

plot\_pacf(sales\_data)

plt.title('PACF Plot')

plt.show()

## LSTM Model: Training the Model and Evaluating Performance

**Training Process:** The LSTM (Long Short-Term Memory) model is a type of recurrent neural network (RNN) designed to handle sequences of data by retaining information over long periods. To train this model:

* **Data Normalization:** The data is scaled using a MinMaxScaler.  
  scaler = MinMaxScaler()  
   features\_scaled = scaler.fit\_transform(features)
* **Sequence Creation:** Time series data is split into sequences (input features and target values) for the LSTM model to learn the patterns.  
  def create\_sequences(data, target, seq\_length):  
   X, y = [], [] for i in range(len(data) - seq\_length): X.append(data[i:i + seq\_length]) y.append(target[i + seq\_length]) return np.array(X), np.array(y)
* **Model Architecture:** The model includes one or more LSTM layers followed by dense layers.   
  model = Sequential() model.add(LSTM(50, activation='relu', input\_shape=(X\_train.shape[1], X\_train.shape[2]))) model.add(Dense(1)) model.compile(optimizer='adam', loss='mse')
* **Training:** The model is trained using the backpropagation algorithm, minimizing the loss function (e.g., Mean Squared Error) over multiple epochs. Training history, including loss curves, is monitored to check for convergence.  
  history = model.fit(X\_train, y\_train, epochs=50, verbose=1, validation\_split=0.1)

**Performance Evaluation:** The model’s performance is evaluated on unseen test data using:  
**Mean Squared Error (MSE):** Measures the average squared difference between predicted and actual values. Lower MSE values indicate better performance.

**Loss Curves:** These plots show how the model's training and validation loss evolves over time. A steady decline in loss values indicates good learning behaviour, while overfitting is detected if the validation loss increases while the training loss decreases.

## Comparison of Models: ARIMA vs LSTM vs Facebook Prophet

ARIMA Model:

Strengths:

Well-suited for stationary data with clear autoregressive and moving average components.

Simple to interpret and requires fewer data points.

Weaknesses:

Does not handle non-linear relationships or complex time-dependent features.

Requires stationary data, which may involve pre-processing like differencing.

LSTM Model:

Strengths:

Capable of capturing non-linear dependencies in sequential data.

Can learn long-term dependencies in time series data without the need for stationarity.

Handles multivariate data (multiple features like sales, quantity, etc.).

Weaknesses:

More complex to train and requires more data compared to ARIMA.

Computationally intensive due to its deep learning architecture.

**Facebook Prophet Model:**

Strengths:

Designed to handle seasonality, holidays, and trends automatically.

Requires minimal data pre-processing, including working with missing data.

Provides interpretable forecast components, such as seasonality and trend effects.

Weaknesses:

Not as effective when data does not exhibit clear seasonal or trend-based patterns.

May overfit in cases of high variance data.

## Comprehensive Report: Data Preparation, Model Implementation, and Analysis

**ARIMA Model:**

Data was loaded and checked for stationarity using the ADF test.

ACF and PACF plots helped in determining the values for p, d, and q. Model Implementation:

An ARIMA model with specific parameters was built, trained, and used to forecast future values. Analysis:

Forecast results were compared to actual data, showing how well the model fits the observed data.

LSTM Model:

Data was normalized using MinMaxScaler and sequences were created for LSTM input.

A Sequential LSTM model was constructed with layers designed to capture long-term dependencies in the data.

The model was trained using training data and evaluated with unseen test data. Analysis:

The loss curve and Mean Squared Error were used to measure performance, showing how well the model captured the data patterns.

**Facebook Prophet Model:**

The date column was formatted as required, and sales data was cleaned to remove outliers.

Additional features such as month and day\_of\_week were added to enhance the seasonality component. Model Implementation:

The Prophet model was fitted, and forecasts were generated for future sales, displaying clear seasonal patterns. Analysis:

Prophet provided an intuitive view of seasonal trends and helped identify key components affecting sales.