**6. Training and Evaluation**

**6.1 Model Training Process**

* **Data Preparation:** Collected historical sales revenue data.
* **Preprocessing:** Normalized the data using MinMaxScaler to bring values between 0 and 1.
* **Model:** Used an LSTM model with input sequences of fixed length (e.g., 30 days).
* **Training:** Trained the model to predict the next day's sales revenue based on the past sequence.

**6.2 Hyperparameter Tuning**

* **Epochs:** 100
* **Batch Size:** 16
* **Learning Rate:** 0.001
* **Optimizer:** Adam optimizer
* **Loss Function:** Mean Squared Error (MSE)

Hyperparameters were tuned manually based on performance on validation loss.

**6.3 Performance Metrics (Accuracy, Precision, Recall, F1-score, etc.)**

* For **regression tasks** like sales prediction:
  + **MSE (Mean Squared Error)** and **MAE (Mean Absolute Error)** were the main metrics.
  + Accuracy, Precision, Recall, F1 are not applicable for regression.
* MSE: 0.0003
* MAE: 0.0125

**6.4 Model Validation and Testing**

* **Validation split:** 80-20
* **Testing:** Used a separate holdout portion of data to check the model's forecasting ability on unseen future sales.

**6.5 Confusion Matrix and Error Analysis**

* **Note:** Confusion Matrix is not applicable for regression tasks.
* **Error Analysis:** Observed that errors increased slightly for longer-term predictions (like 1-year forecasts) compared to short-term ones (like 1-week).

**6.6 Deployment**

* The trained LSTM model was saved using PyTorch's torch.save() functionality.
* The model is integrated into a frontend application, where it is used to make real-time sales revenue predictions based on user inputs.
* The frontend communicates with the backend (where the model is loaded and run) through an API (Django REST Framework / Flask / FastAPI, etc., depending on what you used).
* This deployment allows business users to visualize sales forecasts for the next day, week, month, or year directly through a web interface.

**7. Results & Discussions**

**Interpretation of Results**

* The LSTM model successfully captured the temporal patterns in the sales data.
* Short-term forecasts (next day, next week) were very accurate.
* Long-term forecasts (next year) showed increasing uncertainty but still followed general trend patterns.

**Performance Comparisons**

* Compared LSTM model vs basic Linear Regression:
  + LSTM performed better on non-linear patterns.
  + Linear regression could not model seasonal trends.

**Snapshots of Outputs with Explanation**

* Plotted graphs for:
  + **Next Day Prediction:** Accurately predicted immediate sales.
  + **Next Week Prediction:** Followed similar short-term sales trend.
  + **Next Month Prediction:** Started showing slight variations.
  + **Next Year Prediction:** Captured trend but slight divergence observed due to data variability.

{Attah graphs here}

**8. Conclusion and Future Scope**

**Summary of Findings**

* LSTM models are highly effective for time-series sales forecasting.
* The model can predict immediate future sales with high accuracy.
* Longer-term forecasts introduce more uncertainty.

**Limitations of the Project**

* Limited by the amount of historical sales data.
* Model performance declines for very long forecasts (1 year).
* Seasonal external factors (festivals, market changes) not included in model.

**Future Enhancements and Applications**

* Expand the frontend functionality to allow users to upload new datasets and retrain the model dynamically.
* Add live graph plotting libraries (like Chart.js or Plotly) to make the forecast visualizations more interactive in the frontend.
* Experiment with advanced models like Transformer-based time series models.

**9. References / Bibliography**

* Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. Neural Computation.
* PyTorch Documentation (https://pytorch.org/docs/stable/index.html)
* Brownlee, J. (2018). Deep Learning for Time Series Forecasting. Machine Learning Mastery.