Varsha Venkatapathy and

Harini Gurusankar

Forecasting Revenue with LSTM: A GUI-Deployable ML tool for Sales and Applications

Background

- In the Texas Instruments Sales Org, Annual Plan and Long-Term Revenue for each account has to be calculated every quarter by the Technical Sales Engineer and the Field Applications Engineer
- One of the sales tools to predict this revenue currently is using the Step 6 Dashboard. This outputs an excel file with the past transaction history of an account for the past 5 years, split by quarter
- Forecasting future revenue can pose many difficulties to the current sales/apps org due to:
  - Rapidly Evolving Design wins
  - Variable Customer Volume forecasts for the coming Years
  - Diverse Product Mix Across Market Sectors
- Current TSRs/FAE often have to do extensive market research in each account to predict the future revenue for metrics like volume, supply, and demand
- Data analysis of long term revenue is done with Microsoft Excel. Excel often struggles with:
  - Time-Based dependencies
  - Non-Linear Patterns and Trends
  - Manual Adjustments and Assumptions

Objective

Develop and deploy a machine learning based forecasting tool using Long Short-Term Memory (LSTM) network that improves the accuracy of revenue predictions for Texas Instruments' Annual Plan (AP) and Long Range Planning (LRP). The tool aims to:

- Handle non-linear trends and time based dependencies in revenue data
- Adapt to rapidly changing design wins, customer volume variability, and sector-specific product mixes
- Replace static Excel-based forecasts with a dynamic, scalable and user-friendly solution integrated via a GUI

Preprocessing

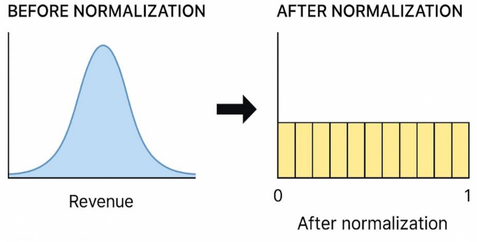
Data Cleaning: Acquired Dataset of an Account. Discarded empty/void rows in Microsoft Excel and converted to CSV. Extracted the Product family most frequently bought by the customer and the corresponding quarterly revenue from 2020 to current

SBE-1
AUDIO
BMS
BMS
MD
LP

→ **Classification (x)-**SBE-1 (Product Family) → **Corresponding Revenue (y)** by Quarter ($) listed as 2020 Q1-2025 Q3 going up by quarterly increments

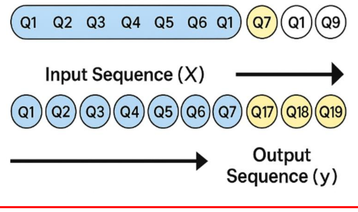
Normalize Revenue Values

MinMaxScaler applied to time series data



Before Normalization: Raw revenue distributions vary widely across quarters, leading to skewed scales.
After Normalization: All revenue values are scaled to [0, 1], ensuring consistent feature magnitudes and stable LSTM training.

Create Sequences with Sliding Window



A window of 16 past quarters (Input Sequence, X) is used to predict the next 3 quarters (Output Sequence, y).

By sliding this window across the time series for each product family, model generate multiple (X, y) pairs for model training.

Concatenate Features

Append encoded class label to each time step

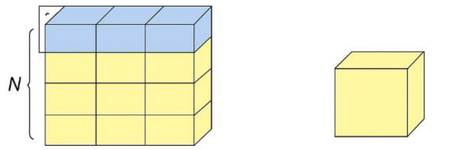
Time Step	Revenue	Encoded Label
Q1	0.81	3
Q2	0.76	3
Q3	0.58	3
Q16	0.55	3

Append encoded class label to each time step

- For each of the past 16 Quarters, normalized revenue is merged with the corresponding SBE-1 product family, however each unique SBE-1 product family is labeled by an integer
- This combined features allows model to learn both temporal categorical patterns simultaneously

Convert to Tensors

Using PyTorch in a Jupyter Environment



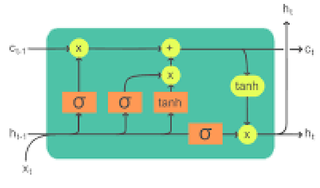
Vectors: [rev, label]

- Each row in X tensor is the sequence of the past 16 quarters, containing 2 values per step: normalized revenue & labeled-encoded SBE-1 product family..
- Y tensor holds the next 3 quarters revenue

Model Architecture

Input-(70% train-20% val., 10% test)

LSTM Layer



LSTM layer- 3 layers total 64 hidden neurons. LSTM is a special RNN designed to learn from sequential data, captures long-term dependencies,

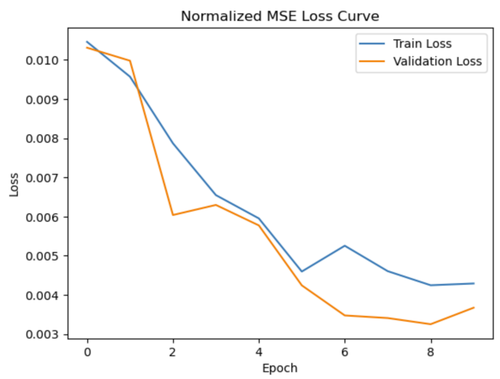
Core Components: Forget Gate, Cell State memory, Input Gate, Output Gate

Fully Connected Layer

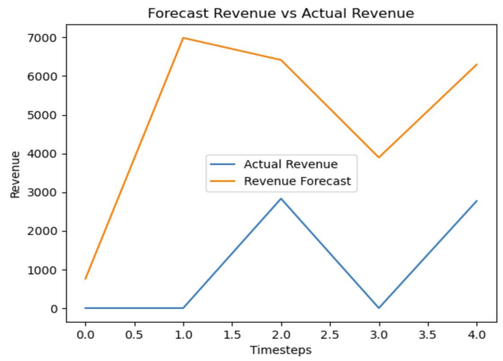
Training used:

- Learning Rate-(1 *10^-3)
- Batch Size=32
- 10 epochs
- Adam optimizer

Results



Graph 1 : Evolution of training and validation Mean Squared Error over 10 epochs. The downward trend indicates improved alignment between predictions and ground truth, with Mean Absolute Error and Mean Squared Error quantifying the average magnitude and squared magnitude of forecast errors, respectively.

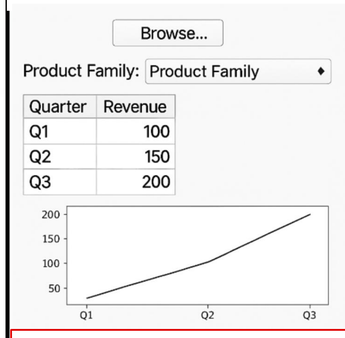


Graph 2: Comparison of predicted revenue (orange) against ground-truth revenue (blue) over successive time steps, demonstrating how closely the model's forecasts match actual values.

MSE = (1/n) * $\Sigma(y_i - \hat{y}_i)^2$
MAE = (1/n) * $\Sigma|y - \hat{y}|$
N-# of samples
Y hat-Predicted Value
Y-Actual Value

Normalized MSE Test Loss	0.0020
MAE Test Loss *Not normalized	5802.34
MSE Test Loss *Not normalized	107344042.069

Table 1: On unseen data, the model achieves a very low normalized MSE (0.0020), indicating strong pattern capture on scaled inputs. In original dollars, it produces an average absolute error of $5,802 (MAE) and an average squared error of 1.07×10^7 (MSE), reflecting typical revenue magnitudes and forecasting accuracy.



GUI Deployment
Built in a Jupyter environment using Streamlit for the interface, and Pandas for data, users can upload a historical revenue CSV, select SBE-1(Product family) in drop down, and future quarter, and instantly view the real time forecasting by the deployed model to see predicted market trends for informed decision making

Future Work

- Use model to predict total available market ($) or expected volumes of a certain material bought by a customer, so TSR can accurately predict how much total revenue expected based on new socket wins
- Train model on multiple inputs like Material, Sectors, etc,
- Experiment with other time forecasting models

Texas Instruments