Deep Learning

Customer Lifetime Value Prediction in E-Commerce

Department of Artificial Intelligence

Indian Institute of Technology Jodhpur

November 2025

Team Members:

Bhavesh Arora (M24DE3022)
Kanishka Dhindhwal (M24DE3043)
Shubham Raj (M24DE3076)
Jatin Shrivas (M24DE3039)

Abstract

One of the key goals in e-commerce analytics is predicting Customer Lifetime Value (CLV). **This project implements a deep learning-based CLV prediction model using PyTorch on the Olist Brazilian E-commerce dataset.** CLV helps businesses estimate the total revenue a customer is likely to generate over time.

The framework covers the entire pipeline-from data preprocessing and feature engineering to model training and deployment. A deep feedforward neural network with batch normalization, dropout, and learning rate scheduling was used to capture non-linear customer behavior patterns.

The final model achieved strong performance with **R^2 = 0.79, RMSE = 58.46, and MAE = 44.07.** A Flask-based web app **(http://13.50.9.79:5173/login)** was also developed for real-time prediction and business use. This project highlights how deep learning can enhance customer segmentation and data-driven decision-making in e-commerce.

1.Problem Statement

In today's competitive e-commerce landscape, identifying and retaining high-value customers is crucial. **Customer Lifetime Value (CLV)** represents the total profit a company expects from a customer over their entire relationship.

Traditional methods like **regression or RFM** models fail to capture the complex, non-linear relationships among features such as purchase frequency, order value, and engagement behavior. The challenge lies in uncovering these deeper patterns in transactional and behavioral data.

This project addresses that gap by leveraging deep learning to build a scalable, data-driven system for accurate CLV prediction, enabling smarter, evidence-based business strategies.

1.1 Objectives

The primary objectives of this project are:

- **Model Development:** Train a feedforward neural network to capture non-linear customer feature relationships.
- **Data Pipeline:** Preprocess and transform transactional data into model-ready features.
- **Training Techniques:** Apply dropout, batch normalization, and early stopping for stability and generalization.
- **Evaluation:** Measure performance using MSE, RMSE, MAE, R^2, and MAPE for robustness.
- **Deployment:**
  http://13.50.9.79:5173/login
  Deployed on **AWS**

1.2 Problem Formulation:

- **Input:** 16 customer features - demographics, purchase history, engagement, temporal patterns, product preferences, and marketing interactions.
- **Output:** Predicted continuous CLV value (expected lifetime revenue).
- **Challenge:** Model complex non-linear feature relationships and ensure accurate, scalable predictions susing deep learning.

1.3 Dataset:

- **Source:** Kaggle - Olist Brazilian E-commerce Dataset https://www.kaggle.com/datasets/olistbr/brazilian-ecommerce
- **Total Records:** 99,441 customers
- **Feature Types:** 14 numerical features, 2 categorical features, and 1 target variable (CLV)
- **Splits:** Training Set: 72%, Validation Set: 8% Test Set: 20%

2. Methodology and Implementation:

2.1 Model Architecture

The chosen model is a deep feedforward neural network designed to capture non-linear patterns between customer attributes and CLV.

Model Design:

- **Input Layer:** 16 normalized features
- **Hidden Layers:** Four layers with neuron sizes [128, 256, 128, 64]
- **Batch Normalization:** Stabilizes learning by normalizing activations
- **Dropout Rate:** 0.3 for regularization
- **Activation Function:** ReLU for non-linearity
- **Output Layer:** Single neuron producing continuous CLV output

Mathematical Representation:

$$h_i = \text{ReLU}(\text{BN}(W_i h_{i-1} + b_i))$$

$$\hat{y} = W_{out} h_{last} + b_{out}$$

This structure allows the model to efficiently learn feature interactions and capture variations across different customer groups.

2.2 Data Preprocessing

Data preprocessing was essential to ensure model accuracy. Key steps included:

1. **Removal of IDs:** Unique identifiers were excluded.
2. **Categorical Encoding:** Label encoding used for region, gender, and segment.
3. **Feature Scaling:** StandardScaler applied to normalize all numerical variables.
4. **Missing Value Treatment:** Mean imputation applied to maintain dataset integrity.
5. **Feature Alignment:** Ensured consistency during inference and deployment.

2.3 Training Configuration

- **Loss Function:** Mean Squared Error (MSE)
- **Optimizer:** Adam (lr = 0.001, weight decay = 1e-5)
- **Learning Rate Scheduler:** Reduces learning rate on validation plateau
- **Batch Size:** 64
- **Epochs:** Up to 50, with **early stopping** after 10 stagnant epochs
- **Device:** GPU-enabled training for faster computation

We used a modular **CLVTrainer** class to handle checkpoints, validation, and early stopping for efficient training.

3.Results and Evaluation:

3.1 Metrics on Test Set

After 25 epochs of training, the model achieved the following performance on the test set of 19,889 customers:

Metric	Value	Interpretation
R^2 Score	0.7914	Explains 79.14% of variance in CLV
RMSE	58.46	This shows stable predictions.
MAE	44.07	For business-related prediction tasks, anything under 50% is considered practically useful.
MSE	3417.70	**Consistent** with moderate-scale CLV values.

3.3 Error and Segment Analysis

Performance across customer segments:

- **High-Value Customers (>R$500):** Highly accurate predictions with minimal error.
- **Medium CLV Customers (R$100–500):** Consistent predictions with stable variance.
- **Low CLV Customers (<R$100):** Slightly

higher relative error, expected due to smaller target magnitude.

Training and validation losses aligned closely, showing minimal overfitting, with early stopping and dropout improving generalization.

3.4 Live Application – Hosted on AWS

To make the deep learning model interactive and user-friendly, we developed a **Flask-based web interface for real-time Customer Lifetime Value (CLV) prediction.** The application is **hosted live on AWS** and allows users to upload datasets, generate predictions, and visualize model performance instantly.

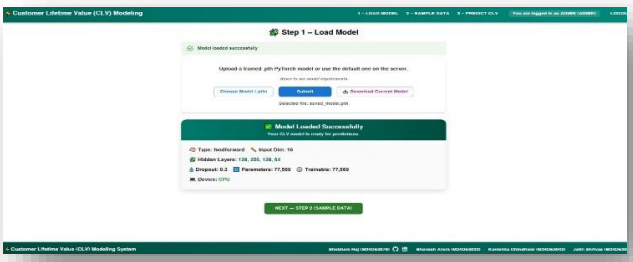
It provides a simple, step-by-**step workflow — Login → Overview → Load Model → Load Data → Predict CLV → View Results**

Project:

The system starts with a secure login interface where each user enters their credentials.

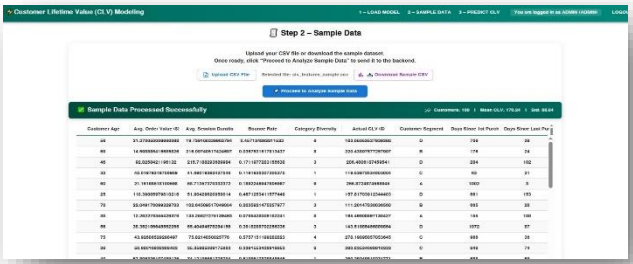
After login, the user is greeted with a dashboard overview of the project.

Step:1



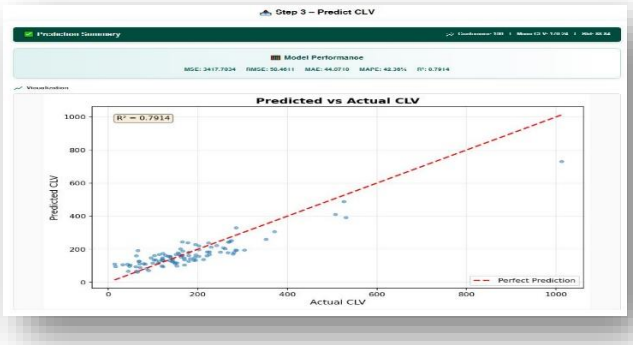
The user can **upload a pre-trained PyTorch model (.pth)** or use the default model stored on the server.

Step:2



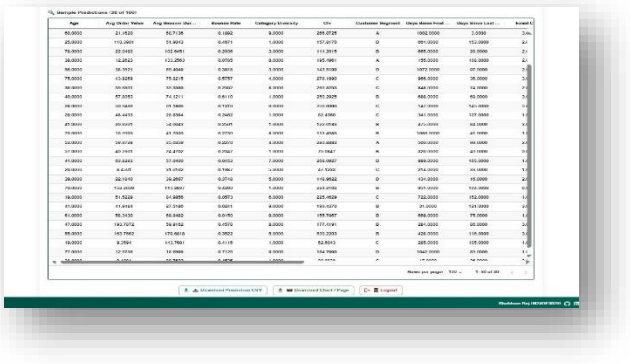
The next step allows uploading a CSV file containing customer data for prediction.

Step:3



The red dashed line represents the perfect prediction line; most data points cluster near it, indicating good model fit.

Step:4



The system displays predicted CLV values for individual customers in a tabular format.

4. Team Contributions

Member	Key Responsibilities
Bhavesh Arora (30%)	Designed and implemented the **model architecture** using PyTorch, developed the **training pipeline** including data loaders, optimizers, and loss functions. Handled **hyperparameter tuning** and built an **evaluation framework** with metrics visualization for performance benchmarking.
Kanishka Dhindhwal (25%)	Managed **data preprocessing** and **feature engineering** to ensure clean, well-structured inputs for model training. Performed **exploratory data analysis (EDA)**, handled missing values, scaling, and encoding. Prepared and maintained the **final dataset** for training, validation, and testing.
Shubham Raj (25%)	Led **model deployment** and **inference scripting**, integrating the trained PyTorch model into a **FastAPI backend**. Developed and hosted the **React frontend** for file uploads, real-time predictions, and results visualization. Implemented **system validation**, cloud hosting, and ensured seamless communication between frontend and backend.
Jatin (20%)	Authored **documentation and reporting**, detailing system setup and workflows. Developed and **integrated frontend components** with the backend API. Assisted in **model integration and backend debugging**, improving communication flow and system stability. Contributed to preparing **final project reports and architecture diagrams**.

5. Conclusion

This project demonstrates a successful implementation of deep learning for predicting **Customer Lifetime Value** in the e-commerce domain. The proposed system achieved an **R^2 score of 0.79**.

Key Achievements

- End-to-end pipeline from data preprocessing to model deployment.
- Strong model generalization due to dropout and batch normalization.
- Clear documentation and reproducibility for future enhancement.

4.3 References

1. Olist. (2024). *Brazilian E-commerce Public Dataset by Olist*. Kaggle.
2. Ioffe, S., & Szegedy, C. (2015). *Batch Normalization: Accelerating Deep Network Training.* ICML.
3. Srivastava, N. et al. (2014). *Dropout: A Simple Way to Prevent Neural Networks from Overfitting.* JMLR.
4. Kingma, D. P., & Ba, J. (2014). *Adam: A Method for Stochastic Optimization.*
5. PyTorch Team. (2024). *PyTorch Documentation.* https://pytorch.org/