Pol Pujol Santaella

Neuronal & Evolutionary Computation

Prediction with Supervised Learning

UNIVERSITAT ROVIRA I VIRGILI

2025

INDEX

1. General Information

This activity is part of the Neural and Evolutionary Computation (NEC). The goal of the assignment is to implement and evaluate different supervised learning models for a regression task. All experiments, code, and documentation associated with this work are available in the following Github repository:

GitHub Repository: PolPujolSantaella/Activity1-NEC-ML

The assignment focuses on three predictive modelling approaches:

1. **Multiple Linear Regression (MLR-F)** using the scikit-learn library.
2. A **Neural Network with Backpropagation (BP)** implemented from scratch, following the mathematical formulation provided in the course materials.
3. A **Neural Network with Backpropagation from a library (BP-F)** using a modern deep learning framework such as PyTorch or TensorFlow.

Throughout the activity, the selected dataset is preprocessed, normalized, and used to train and evaluate the three models. The performance of each method is compared using standard regression metrics (MSE, MAE, MAPE) as well as visual analyses such as predicted vs. true scatter plots and loss evolution curves. Hyperparameter exploration is carried out to identify the best configuration for the manually implemented neural network.

2. Selecting & Analyzing the Dataset

I selected the Realistic Sales Revenue Dataset for this supervised learning activity. The dataset is publicly available on Kaggles and can be accessed here: Realistic Sales Revenue Dataset. It contains synthetic but realistic data about sales revenue, encompassing multiple features such as product categories, customer attributes, sales channels, and geographical information. These rich and varied features make it well suited for regression tasks where the goal is to predict continuous revenue outcomes.

The requirements to select the dataset were:

- At least **10 input features and one output feature**
- At least **1000 patterns**
- The input features should include **numerical AND categorical values**
- The prediction must take real **(float or double)**
- Select randomly **80% of patterns for training and validation**, and the remaining **20% for tests.** (Important to shuffle the original data)

2.1 Dataset Justification

The chosen Realistic Sales Revenue Dataset satisfies all the requirements specified for this activity.

1. **Input Features and Output Feature:** The dataset provides 11 input features and 1 output feature (SalesRevenue).
2. **Number of Patterns:** The dataset contains 2000 entries (patterns).
3. **Feature Types (Numerical & Categorical):** The features are diverse, including both numerical (6 features) and categorical types (4 features).
4. **Prediction Variable Type:** The target variable, SalesRevenue, is a float64 (double), representing continuous, real-valued outcome.
5. **Data Split:** The dataset size of 2000 patterns allows for the required 80/20 split: 1600 patterns are randomly selected, and shuffled, for the training and validation set, leaving 400 patterns for the independent test set. This ensures an adequate amount of data for the training and unbiased final evaluation.

2.2 Dataset Information

The following table resume the 12 columns of the dataset, specifying the relevant information and how can I treat every feature:

Feature	DType	Variable Type	Comments	Treatment
ProductCategory	object	Categorical	**4 unique values**: ['Furniture', 'Toys', 'Electronics', 'Clothing']	One-Hot Encoding
Region	object	Categorical	**4 unique values**: ['East', 'West', 'South', 'North']	One-Hot Encoding
CustomerSegment	object	Categorical	**3 unique values**: ['High Income', 'Middle Income', 'Low Income']	One-Hot Encoding
IsPromotionApplied	object	Categorical	**2 unique values:** ['Yes'. 'No']	Binary Encoding
ProductionCost	float64	Numerical	-	StandardScaler
MarketingSpend	float64	Numerical	-	StandardScaler
SeasonalDemandIndex	float64	Numerical	-	StandardScaler
CompetitorPrice	float64	Numerical	-	StandardScaler
CustomerRating	float64	Numerical	-	StandardScaler
EconomicIndex	float64	Numerical	-	StandardScaler
StoreCount	int64	Numerical	-	StandardScaler
SalesRevenue	float64	Output (Target)	-	StandardScaler

2.3 Preprocessing Techniques Justification

Missing Values & Duplicates

No missing values were found in any variable, and no duplicate records were detected. This ensures that the preprocessing pipeline and subsequent modeling phases (MLR-F, BP, BP-F) operate on a clean dataset without requiring imputation or deduplication procedures.

Categorical Variables

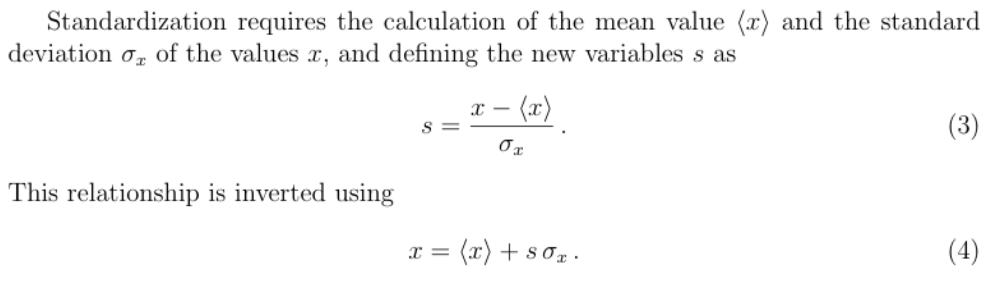
One-Hot Encoding will be used to convert the nominal categorical variables (ProductCategory, Region, CustomerSegment) into a binary format that regression models can interpret. For IsPromotionApplied, Binary Encoding will be used, resulting in columns of 1s and 0s.

Data Normalization / Transformation

Given the disparity in the scales of the numerical variables (for example, CustomerRating has a range of approx. 3, while ProductionCost has a range of approx. 730), **Standardization (StandardScaler)** is necessary. This process scales the data to have a mean of 0 and a standard deviation of 1, which is essential for the proper and efficient training of the Neural Network with Backpropagation.

The target variable (SalesRevenue) will also be standardized to facilitate model convergence, and the final prediction will be rescaled back to their original values for evaluation.

The input features and the target variable (SalesRevenue) must be scaled with different scalers. Each scaler learns the mean and standard deviation of the specific data it receives, and mixing inputs with the target in a single scaler would distort both.



Standardization requires the calculation of the mean value $\langle x \rangle$ and the standard deviation σ_x of the values x, and defining the new variables s as

$$s = \frac{x - \langle x \rangle}{\sigma_x} \,. \tag{3}$$

This relationship is inverted using

$$x = \langle x \rangle + s \, \sigma_x \,. \tag{4}$$

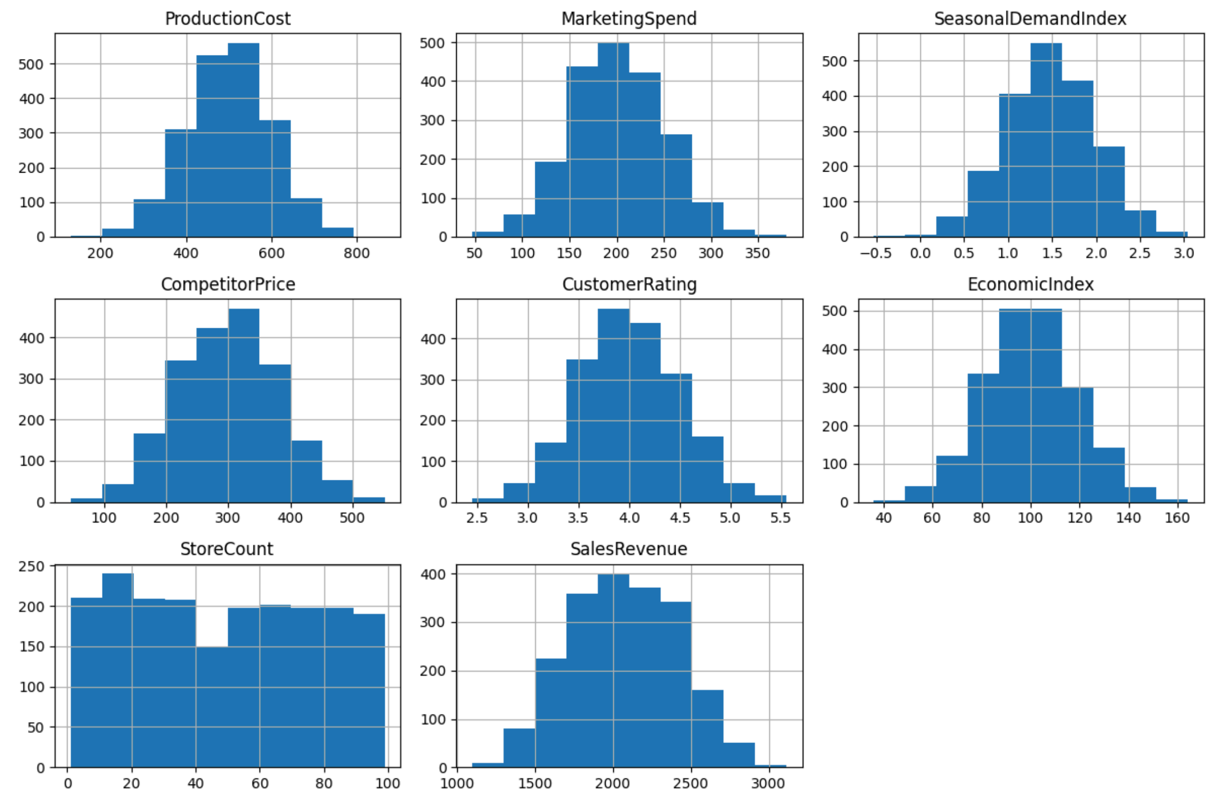
Standardization (Photo from PDF)

Analysis of Numerical Data Distribution

The histograms generated for the numerical features provide crucial insight into the data distribution and reinforce the necessity of preprocessing. Most variables, including ProductionCost, MarketingSpend, CompetitorPrice,CustomerRating, and the target variable SalesRevenue, exhibit distributions that are approximately symmetric and concentrated around their mean, resembling a normal or Gaussian distribution.

However, the variables display significant differences in their scale and range (for example, CustomerRating ranges from 2.5 to 5.5, while SalesRevenue ranges from 1000 to 3100).

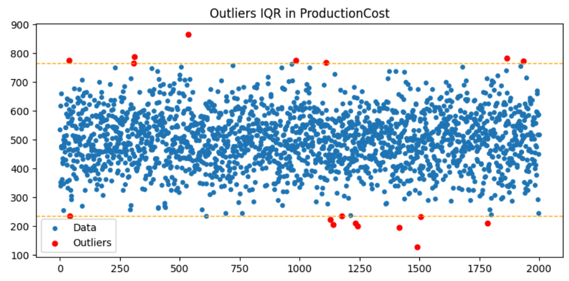
This analysis confirms the need for Standardization (Z-score normalization) across all numerical input features and the output variable. Scaling the data to a standard mean of 0 and standard deviation of 1 is essential to prevent features with larger magnitudes.

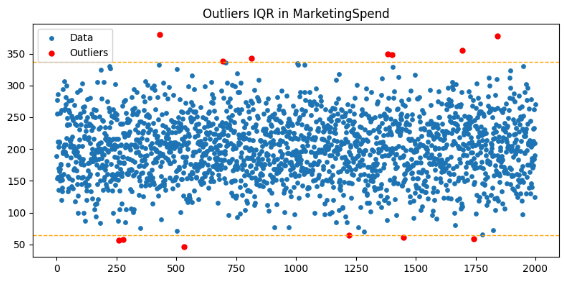


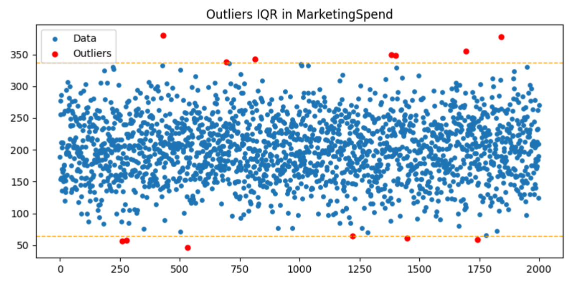
Outliers Detection

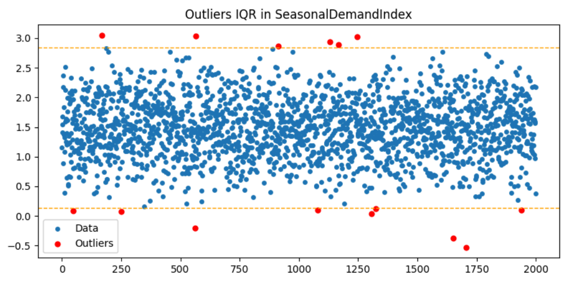
The presence of atypical data points (outliers) was investigated as part of the data cleaning process. The Interquartile Range (IQR) method was employed, defining an observation as an outlier if it falls below Q1 - 1.5 x IQR or above Q3 + 1.5 x IQR.

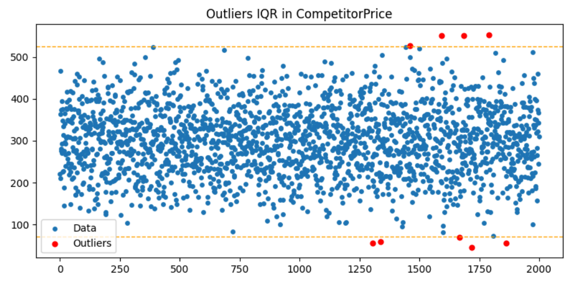
The analysis revealed a small number of outliers across several key features. Given the synthetic yet realistic nature of the Sales Revenue Dataset, these identified points are considered natural variations within the business process. Therefore, no outliers were removed or modified, and the data was retained in its entirety for the normalization and training.

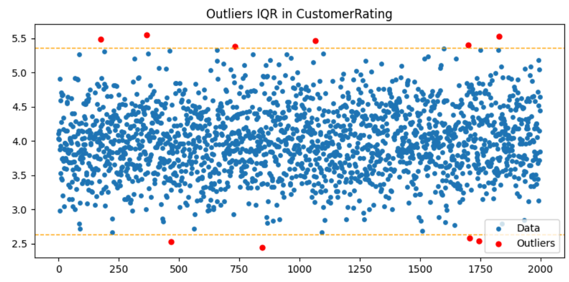


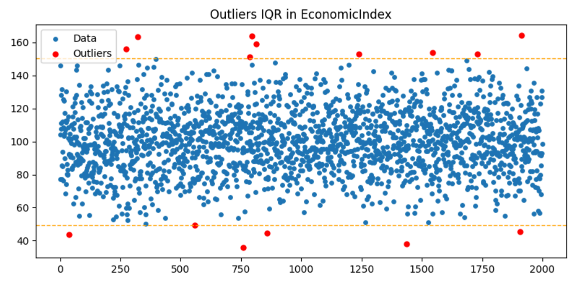












Data Split

The original dataset of 2000 patterns was divided into two main subsets after an initial shuffling process. Shuffling is essential to destroy any kind of sorting it could have.

The partition was carried out as follows:

- **Test Set (20%):** 400 patterns were reserved for the test set.
- **Training and Validation Set (80%)** The remaining 1600 patterns used for model training.

Additionally, within this set of 1600 patterns, a portion will be further separated for the validation phase during the training of the manually implemented Backpropagation Neural Network (BP).

3. Obtaining & Comparing Predictions (BP, BP-F, MLR-F)

3.1. Hyperparameter Comparison & Selection for BP

The performance of the implemented Backpropagation Neural Network (BP) is highly dependent on its architecture and training parameters. An extensive exploration was carried out on the training/validation set (80% of the data) to identify the configuration that yields the best generalization capability, focusing on minimizing the Mean Absolute Percentage Error (MAPE). The following table summarizes the 10 combinations tested, where the input layer has 16 units and the output layer has 1 unit.

Layers	Structure	Epochs	Learning rate	Momentum	Activation Function	MAPE	MAE	MSE
3	16:32:1	100	0.001	0.9	relu	0.048	96.88	14759.8
3	16:32:1	100	0.001	0.0	relu	0.058	109.18	19594.3
3	16:32:1	100	0.01	0.5	sigmoid	0.040	80.52	10399.6
4	16:64:32:1	100	0.01	0.9	relu	0.737	1493.2	3041363
3	16:128:1	100	0.001	0.9	tanh	0.0590	117.95	21675.3
3	16:64:1	100	0.1	0.9	relu	8.270	16403	318475566
3	16:64:1	100	0.01	0.0	sigmoid	0.041	82.89	11349.9
4	16:64:32:1	100	0.001	0.5	tanh	0.058	116.69	21589.8
3	16:8:1	100	0.0001	0.9	linear	0.037	74.89	8851.87
4	16:32:16:1	100	0.0001	0.9	relu	0.273	592.14	478481.9

We intentionally varied the network structure (ranging from simple models like 16:8:1 to more complex configurations such as 16:64:32:1), the type of activation function (including standard non-linear functions like Sigmoid, ReLU, and Tanh, as well as the linear case Linear), and the optimization parameters. Specifically, the influence of the learning rate was tested across a wide range (from 0.0001 to 0.1) to identify the optimal convergence speed, and momentum was experimented with to evaluate its effect on stability and its ability to overcome local minima.

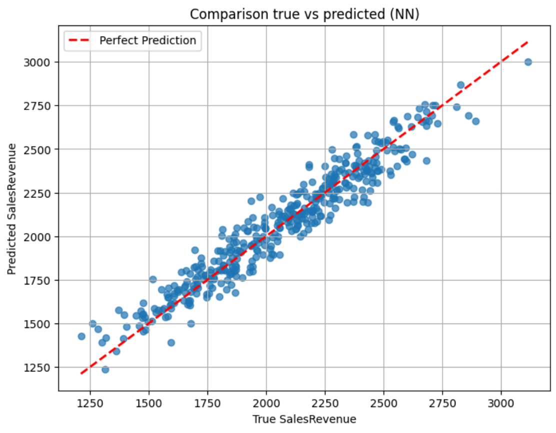
3 Best Configurations

To provide a comprehensive conclusion and show the robustness of different functional forms, we select the best performing configuration for three different activation functions based on the lowest MSE and MAE on the test set.

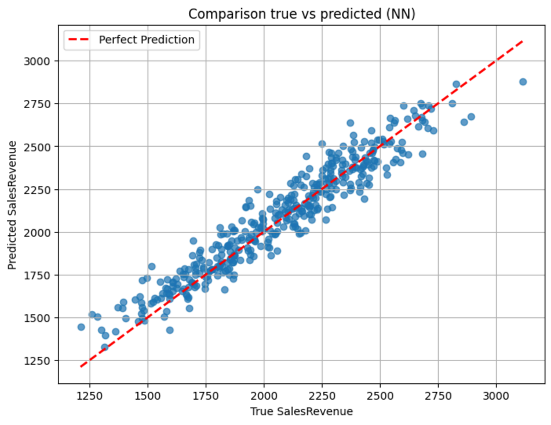
Rank	Config	Activation Function	Layer Structure	Learning Rate	Momentum	MAPE	MAE	MSE
1	9	Linear	16:8:1	0.0001	0.9	0.037	74.89	8851.87
2	3	Sigmoid	16:32:1	0.01	0.5	0.040	80.52	10399.6
3	1	Relu	16:32:1	0.001	0.9	0.048	96.88	14759.8

Scatter Plots Predicted vs. True

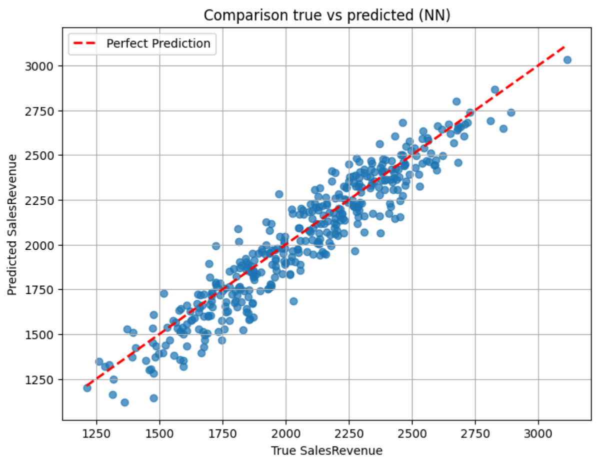
Rank 1



Rank 2

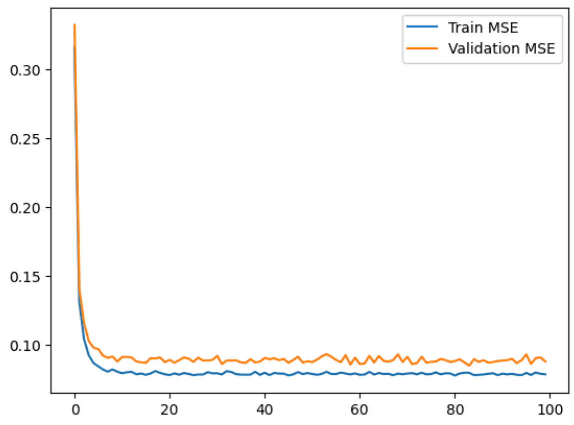


Rank 3

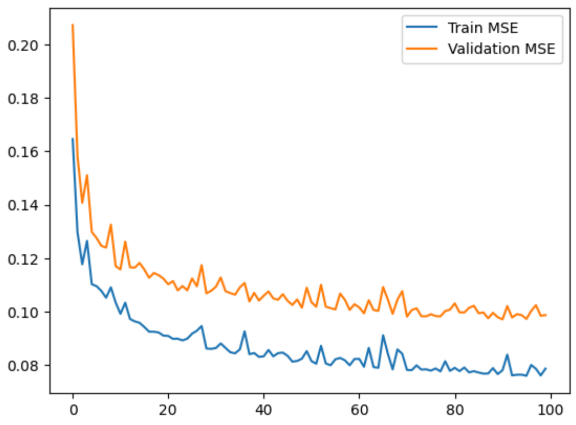


Evolution of the Error Training & Validation

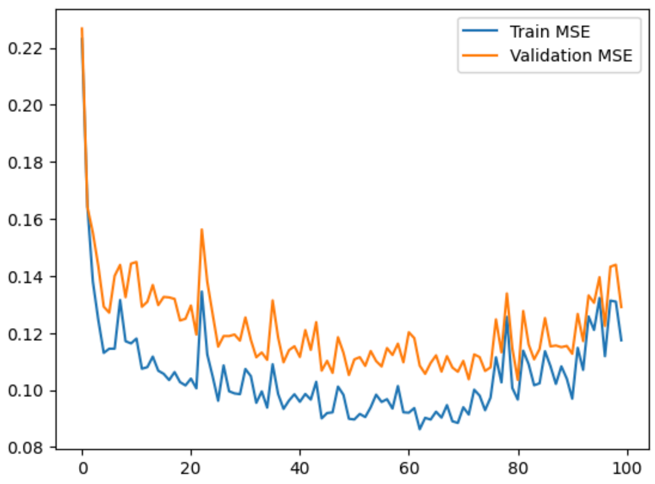
Rank 1



Rank 2



Rank 3



Discussion of the TOP 3 Configurations

The best-performing configuration (Configuration 9, linear activation) achieved the lowest MSE and MAE values. This strong performance suggests that the underlying relationship between the inputs and SalesRevenue is predominantly linear, or at least sufficiently linear for a simple architecture to outperform more complex ones. The requirement of an extremely small learning rate (0.0001) indicated high gradient sensitivity, where even minor updates could destabilize training.

The second-best model (Configuration 3, sigmoid activation) represents the most effective non-linear alternative. The sigmoid function inherently restricts gradient magnitude due to its saturating behavior, allowing the model to use a higher learning rate without suffering exploding gradients. The moderate momentum (0.5) improves training stability by smoothing the updates while still accelerating convergence. It demonstrates tha mild non-linearities exist in the data, even if they do not contribute enough predictive power to surpass the linear solution.

The third-ranked model (Configuration 1, ReLU activation) shows that ReLU-based networks can also be trained successfully on this database, but only under conservative learning conditions. The low learning rate (0.001) prevents gradient explosions typically associated with ReLU, while the high momentum (0.9) compensates by boosting convergence speed.

For the purpose of a robust comparison and the exploration of the non-linear capabilities of neural networks, Configuration 3 (Sigmoid Activation), the 2nd-best model, will be adopted as the definitive Backpropagation (BP) model. This will be a genuine assessment of whether the non-linear representation capacity provides a tangible predictive value that justifies the increased complexity.

3.2. Model Result Comparison (BP vs. MLR-F vs BP-F)

MLR-F

The Multiple Linear Regression (MLR-F) model was implemented using the LinearRegression class from the scikit-learn library. This model is used as a baseline to assess the performance complexity gained by the non-linear neural network models (BP and BP-F).

The output variable (SalesRevenue) was scaled back using the inverse transformation to present the final metrics in the original units, ensuring a meaningful comparison of the metrics.

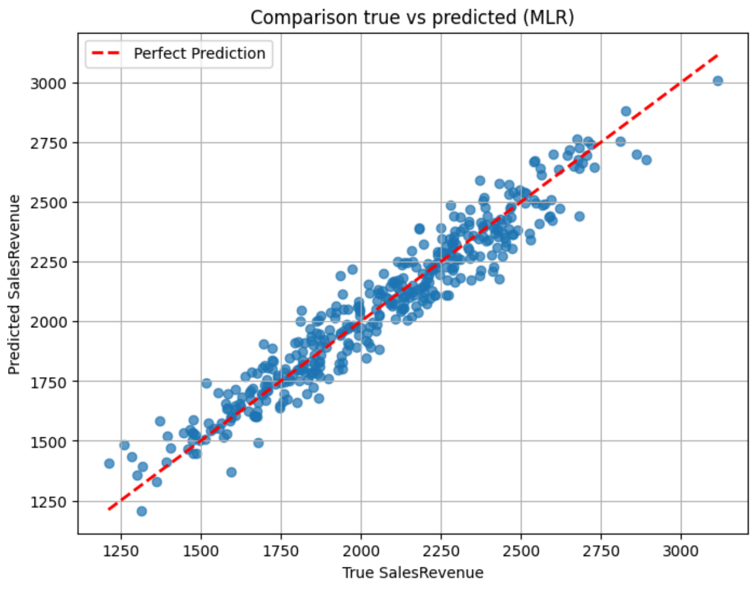
Performance

METRIC	TRAINING RESULT	TEST RESULT
MAE	77.7947	73.4672
MSE	9486.6399	8560.7863
MAPE	0.0387	0.0365

The MLR-F model achieves a low MAPE of approximately 3.65% on the test set, indicating that a simple linear relationship captures a significant portion of the variance in the Sales Revenue.

Scatter Plot Predicted vs. True

The tight clustering of the points around the red diagonal line (Perfect Prediction) visually confirms the model's strong predictive capability, even with a basic linear assumption.



BP-F (PyTorch)

The second model, a simple, fully-connected Neural Network (NN), was implemented using the PyTorch deep learning framework. This model serves as the first non-linear attempt to capture potentially more complex relationships in the data compared to the linear baseline.

The parameters used in this model are the same 2nd best configuration of the BP implemented.

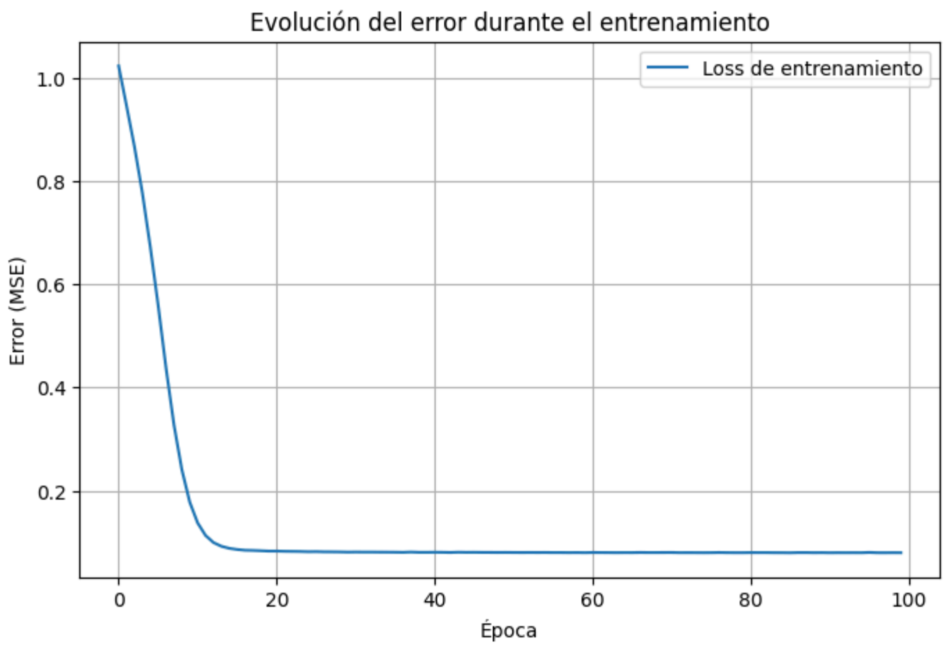
Performance

METRIC	TRAINING RESULT	TEST RESULT
MAE	78.1648	74.1323
MSE	9521.4823	8667.7908
MAPE	0.0399	0.0369

Scatter Plot Evolution of Error

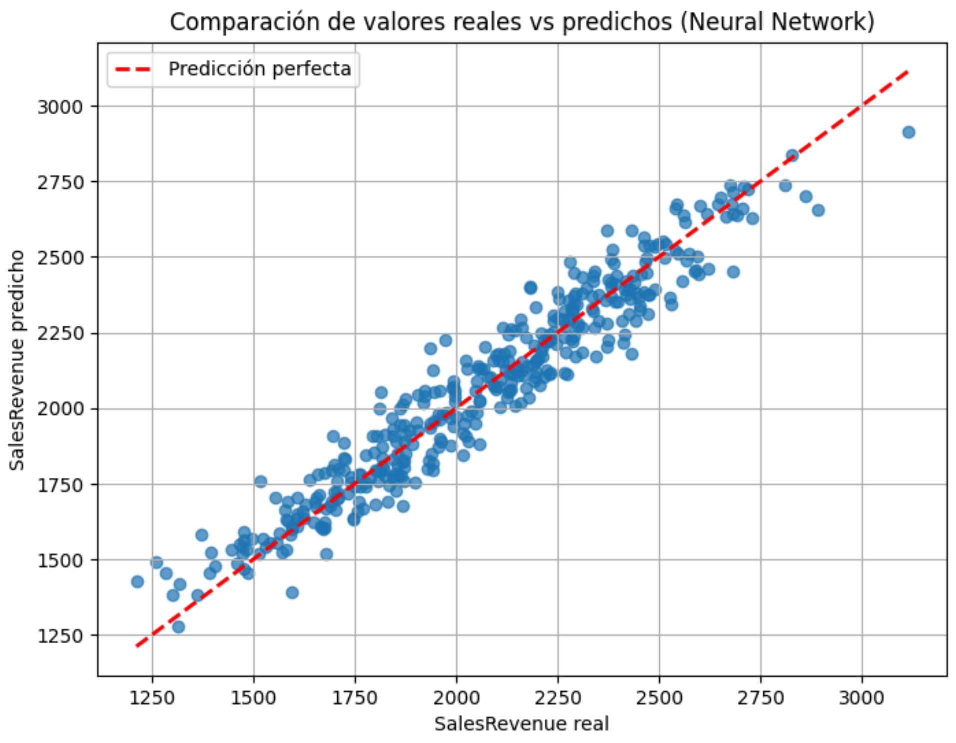
The following figure displays the evolution of the MSE loss during the 100 epochs of training for the simple Neural Network model.

The plot shows the speed and stability of the model's convergence, illustrating how quickly the network optimized its internal parameters.



Scatter Plot Predicted vs. True

The tight clustering of the points around the red diagonal line (Perfect Prediction) visually confirms the model's strong predictive capability, even with a basic linear assumption.



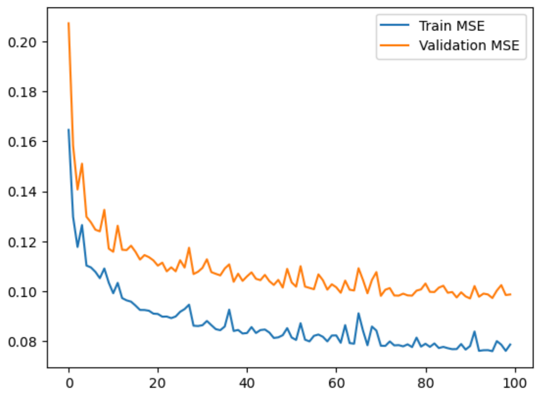
Comparison Table of 3 Models

To conclude the model evaluation, the performance of the three implemented models, the MLR (Multiple Linear Regression) baseline, the simple Neural Network (BP), and the advanced Neural Network (BP-F), is consolidated and presented in the table below.

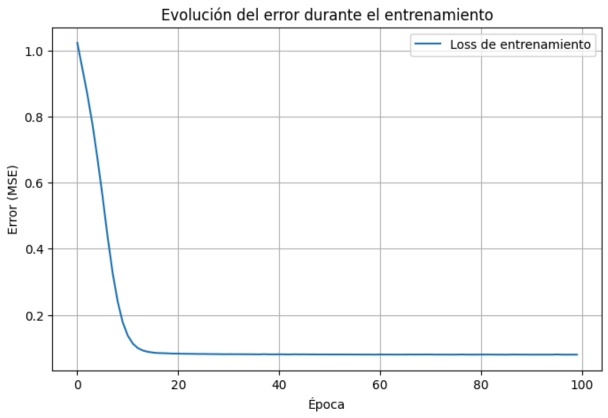
MODEL	MAE	MSE	MAPE
MLR-F	73.4672	8560.7863	0.0368
BP-F	74.1323	8667.7908	0.0369
BP	80.52	10399.6	0.040

Comparison Scatter Plots

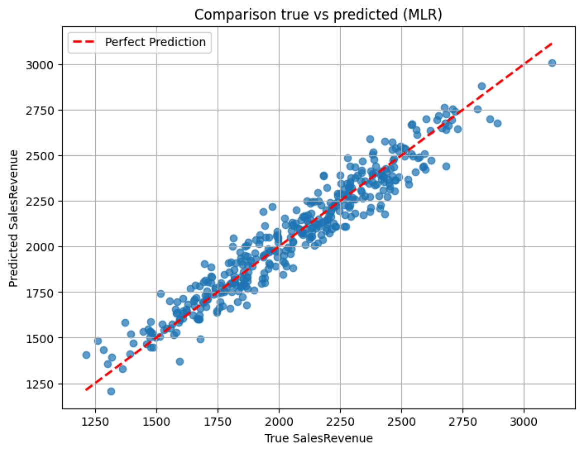
BP Implemented



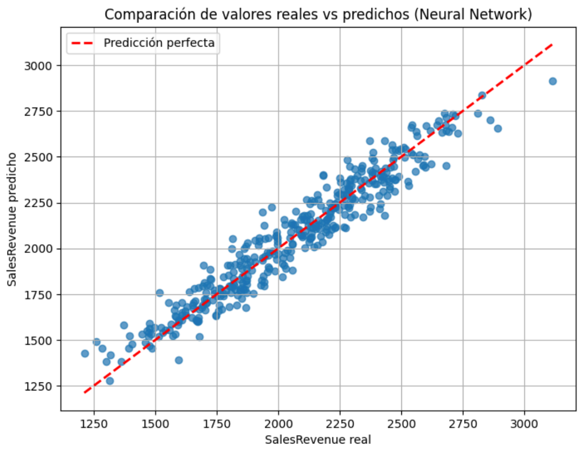
BP Pytorch



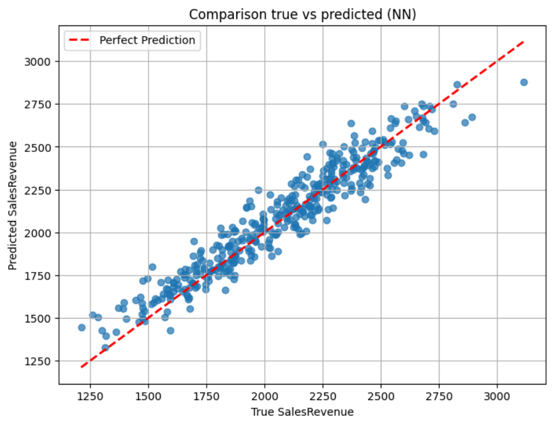
MLR



BP Pytorch



BP Implemented



Discussion

3 Models Conclusion

The comparative results clearly demonstrate the nature of the relationship within the Realistic Revenue Dataset. The Multiple Linear Regression (MLR-F) model achieves the best overall performance across all test metrics (MAE: 73.47, MSE: 8560.79, MAPE: 0.0368). This superior performance, confirmed by the tight clustering in its Predicted vs True Scatter plot, serves as a strong indication that the variance in Sales Revenue is predominantly explained by a linear combination of the input features.

The slightly more complex neural network models (BP-F and BP), despite their capacity for non-linear representation, were unable to surpass the linear baseline. The PyTorch implementation (BP-F) performed marginally worse than the baseline (MAE: 74.13, MAPE: 0.0369), suggesting the mild non-linearities captured by the Sigmoid activation did not provide enough predictive value to offset the added complexity and optimization challenge. The simplest adequate model (MLR) is the best choice for this specific problem and dataset.

BP Vs. BP-F

The 2 Neural Network approaches, the manually implemented Backpropagation (BP) and the framework-based BP-F (Pytorch), offered distinct insights into functionality. The BP-F model benefited from highly optimized, stable gradient computation, automatic differentiation, and robust parameter initialization/update mechanisms, leading to its efficient convergence and the second-best performance.

In contrast, the custom BP implementation was a significant pedagogical challenge. While the custom BP was successfully trained and achieved a respectable result, its performance was slightly lower and its sensitivity to hyperparameters (especially the learning rate) was far greater, as evidenced by the extensive hyperparameter search and the evolution error scatter plot.

4. Optional Parts

Part 1: Study the effect of the different regularization techniques in the Neural Network used in the (BP-F)

This section documents the investigation of methods to improve the generalization of the PyTorch Neural Network (NN) model by incorporating two crucial regularization techniques: L2 Regularization (Weight Decay) and Dropout.

4.1 L2 Regularization (Weight Decay)

L2 Regularization adds a penalty term to the loss function that is proportional to the square of the magnitude of the network's weights (w).

$$\text{Total Loss} = \text{Original Loss} + \lambda \sum w^2$$

This penalty encourages the optimization algorithm to prefer smaller weights. Models with smaller weights are considered "smoother" and simpler, which helps reduce their complexity and sensitivity to minor fluctuations in the data.

Tested Parameters

Value	Rationale
0.0001	Very small penalty to test even minimal constraint provides benefit
0.001	Moderate regularization strength
0.01	Stronger value to test the boundary of beneficial regularization before the model starts to underfit.

Comparison Table

MODEL	MAE	MSE	MAPE
BP-F	74.1323	8667.7908	0.0369
BP-F (L2: 0.0001)	74.0028	8724.2083	0.0369
BP-F (L2: 0.001)	74.09886	8692.8105	0.0368
BP-F (L2: 0.01)	73.7523	8686.2591	0.0367

The comparison reveals:

- **MAE improvement:** All L2-regularized models achieve a lower MAE, confirming a marginal improvement in generalization and prediction accuracy. The highest L2 value tested (0.01) yielded the best MAE of 73.75, suggesting that a stronger penalty was necessary to push the model towards a simpler representation and better test set performance.
- MSE and MAPA Stability: The MSE remained highly stable across all L2 values. The best MSE was achieved with L2 = 0.01 (8686.26), showing a very slight increase from the base model.
  The MAPE also remained almost identical across all configurations, indicating that the relative error in sales revenue prediction was minimally affected by L2 regularization.

The application of L2 regularization to the BP-F model provides a minor, yet consistent, improvement in performance on the test set. The optimal regularization strength is L2 = 0.01, which results in the best balance between simplicity and predictive power, achieving a test MAE of 73.75.

4.2. Dropout

Dropout is a regularization method unique to Neural Networks that is applied during the training phase. It involves randomly deactivating (setting to zero) a fixed proportion (p) of the neurons in a layer for each forward and backward pass.

$$\text{Active Neurons} = (1 - p) \times \text{Total Neurons}$$

By randomly dropping neurons, the network cannot rely too heavily on any single feature or a small group of interconnected neurons. This forces the network to learn more robust and redundant representations, where features are processed by multiple independent paths. Dropout effectively trains an ensemble of many smaller, shared-weight networks.

Tested Parameters

Value	Rationale
0.1	Small dropout rate keeping most neurons active. Should offer minimal improvement.
0.3	Balanced rate. Expected to be the optimal choice.
0.5	Maximum rate. Forces the network to generalize heavily, but many lead to high training error or underfitting.

Comparison Table

MODEL	MAE	MSE	MAPE
BP-F	74.1323	8667.7908	0.0369
BP-F(Dropout: 0.1)	75.99770	9205.91	0.0382
BP-F(Dropout: 0.3)	81.5519	10609.24	0.0410
BP-F(Dropout: 0.5)	88.933	12664.45	0.0448

The comparison reveals:

- **Degraded Performance:** Unlike the beneficial effect of L2 Regularization, the application of Dropout degraded the performance of the Neural Network across all tested rates when compared to the BP-F original model. The lowest MAE achieved with Dropout was 75.98, which is worse than the BP-F MAE of 74.13.

- **Inverse Relationship to p:** The model performance directly decreased as the dropout rate increased. P = 0.1 gave the best results (MAE: 75.98) and P = 0.5 resulted in the worst model (MAE: 88.93), which is even worse than the initial, non-regularized, custom-implemented Neural Network (BP-F: 80.52)

In conclusion, for this specific Sales Revenue forecasting task and model architecture, Dropout is not an effective regularization technique. The method introduces too much randomness and bias, hindering the learning process and making the final model less accurate than the non-regularized benchmark. The L2 Regularization method was confirmed to be the superior technique for optimizing this model.

Part 3: Ensemble Learning Models

4.3 XGBoost

Gradient Boosting is a sequential ensemble method where new models are trained to specifically correct the errors (residuals) made by previous models in the sequence. XGBoost is an optimized version that uses a more regularized model formalization (L1 and L2 regularization) to prevent overfitting, which leads to better performance and faster computation compared to traditional Gradient Boosting implementations.

The process can be summarized as:

1. An initial prediction is made
2. A weak learner (a shallow decision tree) is trained to predict the residuals (the difference between true target and the current prediction).
3. The weak learner’s prediction is added to the current prediction
4. Steps 2 and 3 are repeated for a specified number of estimators, with each new tree correcting the errors of the preceding ensemble.

4.4. Random Forest

The second ensemble technique implemented is the Random Forest Regressor, a type of Bagging ensemble method specifically tailored for decision trees.

Random Forest is built on two key concepts:

1. Bagging (Bootstrap Aggregating): It involves training multiple estimators (decision trees) on different subsets of the training data, sampled with replacement (bootstrap samples).
2. Feature Randomness: During the tree construction, instead of considering all features for the best split at each node, only a random subset of features is considered.

By introducing randomness in both the data and the feature set, the trees are made independent, resulting in an ensemble of diverse, high-variance models. For regression, the final prediction is obtained by averaging the individual predictions from all trees in the forest.

Parameter Selection and Justification

Hyperparameter tuning was performed using a Grid Search with 3-fold Cross-Validation and scoring of negative mean squared error. This rigorous method ensures that the final parameters are selected based on the best performance across multiple subsets of the training data, optimizing for the lowest MSE.

XGBoost

Parameter	Value	Justification
n_estimators	[200, 300, 500]	Moderate number of trees to balance performance
learning_rate	[0.01, 0.05, 0.1]	Lower values require more estimators but improve generalization
max_depth	[3, 4, 5]	Keeping the trees shallow to prevent overfitting
subsample	[0.8. 1.0]	Introduces randomness to make the model more robust
colsample_bytree	[0.8, 1.0]	Prevents individual trees from relying too heavily on a small set of features.

Random Forest

Parameter	Value	Justification
n_estimators	[300, 500, 700]	Large number of trees to ensure sufficient aggregation
max_depth	[5, 7, 9]	Prevents individual trees from becoming too complex and overfitting the bootstrap samples.
min_samples_split	[6, 10, 14]	More regularization, forcing nodes to have more samples before splitting.
min_samples_leaf	[3, 5, 8]	Prevents trees from creating pure leaves on only one or two samples, stabilizing the model.

Optimal Parameters

XGBoost

The GridSearch identified the following combination of parameters as the best with a cross-validation score of -0.09428 (Negative MSE).

Parameter	Value	Justification
n_estimators	300	Sufficient number of boosting rounds.
learning_rate	0.05	Low learning rate, slower-to-overfit model.
max_depth	3	Remain “weak learners”.
subsample	0.8	Add regularization and stability
colsample_bytree	0.8	Reduce variance and overfitting.

Random Forest

The GridSearch identified the following combination of parameters as the best with a cross-validation score of -0.12478 (Negative MSE).

Parameter	Value	Justification
n_estimators	700	Highest number of trees explored
max_depth	9	Highest maximum depth, deeper trees were beneficial to capture the data’s complexity.
min_samples_split	6	6 samples to initiate an internal split.
min_samples_leaf	3	Minimum numbers of samples required at a leaf node, promoting generalization

Comparison of Results

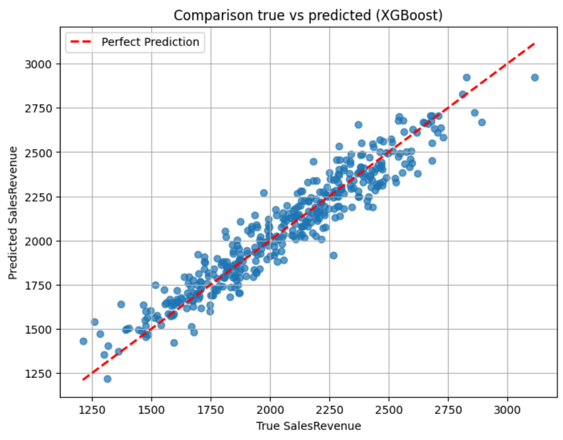
The table below compares the performance of the implemented ensemble models (XGBoost and Random Forest) against the previously established benchmark models (MLR-F, BP-F, and BP).

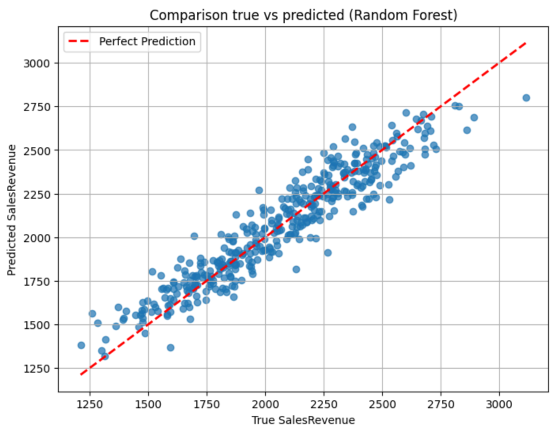
MODEL	MAE	MSE	MAPE
MLR-F	73.4672	8560.78	0.0368
BP-F	74.1323	8667.79	0.0369
BP	80.52	10399.6	0.040
XGBoost	80.11	10388.68	0.039
Random Forest	89.85	12928.05	0.0448

The Multiple Linear Regression (MLR-F) model remains the best overall performer, achieving the lowest MAE and MSE.

Among the ensemble techniques, the XGBoost Regressor performed significantly better than the Random Forest, with a Test MAE of 80.11 vs. 89.85. The Random Forest model had the worst performance across all test set metrics compared to all other models (MLR-F, BP-F, BP, and XGBoost.)

In conclusion, for this specific dataset, the simple linear model (MLR-F) proved the most effective. While the tree-based ensembles are powerful, they did not outperform the linear model. XGBoost demonstrated better generalization and performance stability than Random Forest, establishing itself as the superior ensemble technique for this task, though it still falls short of the MLR-F baseline.


Comparison true vs predicted (XGBoost)
Perfect Prediction
Predicted SalesRevenue
True SalesRevenue


Comparison true vs predicted (Random Forest)
Perfect Prediction
Predicted SalesRevenue
True SalesRevenue