“BAN 693”

Project Summary Report

CONTRIBUTOR: ABHISHEK MALLA
amalla3@horizon.csueastbay.edu

PROFESSOR: SURENDRA SARNIKAR
surendra.sarnikar@csueastbay.edu

Project Title: Sales Forecasting Using Machine leaning and Neural Networks

1.Introduction: The goal of this project is to predict quarterly sales to various customers of a steel manufacturing company. The dataset includes a mix of synthetic and real data, with information about the company's customers in the Auto, Metal Fabrication, and Infrastructure sectors.

2.Dataset Description:
Main Dataset (train.csv): Contains company-specific data for the steel manufacturer's 75 customers.
Additional Dataset (EconomicIndicators.csv): Provides various economic indicators at a monthly level, which can be used in conjunction with company-specific data for sales prediction.

Columns

Train/test.csv

- ID - Row id column
- Company - Name of the company/customer
- Quarter - Quarter for which the sales are provided/to be predicted
- Quick Ratio - Financial ratio indicating the customer's liquidity situation
- Inventory Ratio - Ratio of sales over inventory
- Revenue Growth - Revenue growth projections based on analyst and company projections
- Market share Change - Market share growth projections based on analyst and company projections
- Bond rating - Bond rating of company
- Stock rating - Stock rating of company
- Region - Region in which the company is situated or operates primarily
- Industry - Industry are of company
- Sales - Sales for the given quarter (target variable)

EconomicIndicators.csv

- Month - Month for which the indicators are provided
- Consumer Sentiment - Consumer sentiment index value based on survey of consumers
- Interest Rate - Average yield of 5 year US Treasury
- PMI - Purchasing Managers Index
- Money Supply - M2 Money supply

- `NationalEAI' - National Economic Activity Index
- EastEAI, WestEAI, SouthEAI, NorthEAI - Regional Economic Activity Index

3.Objective:

The objective of this project is to develop an accurate predictive model that can forecast quarterly sales for each of the 75 customers. The model will utilize both company-specific data and general economic indicators to make predictions.

4.Project Background:

The goal of this project is to forecast sales for the next two quarters (Q8 and Q9) using historical sales data. The dataset contains sales data for 75 customers over a period of 2 years and 3 months. The project initially focused on using machine learning models such as Linear Regression, XGBoost, RandomForestRegressor, and MLPRegressor for sales prediction.

5.Machine Learning Models Explored:

5.1 Linear Regression:

Implemented linear regression models to predict sales based on numerical and categorical features.
Achieved moderate performance but further optimization was required to improve accuracy.

5.2 XGBoost (Extreme Gradient Boosting):

Utilized the XGBoost algorithm to predict sales, a powerful ensemble learning technique known for its performance and scalability.
Experimented with different hyperparameters and conducted hyperparameter tuning using GridSearch with CV(Cross Validation) to improve model performance.

5.3 RandomForestRegressor:

Implemented the RandomForestRegressor model to predict sales, which is an ensemble learning method based on decision tree regressors.
Conducted hyperparameter tuning using GridSearch with CV(Cross Validation) to optimize model performance.

5.4 Gradient Boosting:

Explored the Gradient Boosting algorithm for sales prediction, another ensemble learning technique that builds trees sequentially, with each tree correcting the errors of the previous one.
Fine-tuned hyperparameters to achieve better results.

5.5 Neural Networks (MLPRegressor):

Implemented Multi-layer Perceptron (MLP) models using the MLPRegressor class from scikit-learn.
Experimented with different architectures, activation functions, solvers, and regularization techniques to improve model performance.
Achieved significant improvement in accuracy compared to traditional machine learning models.

6.Project Evolution:

The project underwent several iterations and enhancements as follows:

7.Handling Missing Values:

Utilized SimpleImputer with the mean strategy to handle missing values in the "InventoryRatio" column.
Employed one-hot encoding for handling categorical columns.
Implemented pipelines with ColumnTransformer for preprocessing numerical and categorical features.

8.Model Selection and Optimization:

Explored various machine learning models including Linear Regression, XGBoost, RandomForestRegressor, and MLPRegressor.

Performed hyperparameter tuning using techniques such as GridSearchCV to optimize model performance.

Experimented with different configurations of hyperparameters for each model to achieve better results.

9.Incorporating Economic Indicators:

Imported an additional dataset containing economic indicators and merged it with the sales data based on the "Quarter" column.
Explored linear regression models to incorporate economic indicators into the forecasting process.

10.Transition to Time Series Analysis:

Proposed a transition from machine learning models to time series forecasting models, specifically the ARIMA model.
Distributed sales data into quarterly periods to align with natural business quarters (Q1: Jan-Mar, Q2: Apr-Jun, Q3: Jul-Sep, Q4: Oct-Dec).
Utilized the ARIMA model to capture seasonality and make accurate sales forecasts.

11.Model Evaluation and Optimization:

Evaluated model performance using metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), and Root Mean Squared Error (RMSE).

Continuously refreshed and re-run the code to fine-tune the models and achieve better scores.

12.Final Model Implementation:

Developed a MLPRegressor model using a Pipeline with preprocessed features and the MLPRegressor model.
Achieved a training R^2 score of 0.81 and a testing R^2 score of 0.72.

13.Results:

The project achieved a significant improvement in performance by transitioning from traditional machine learning models to neural networks.

The MLPRegressor model outperformed other machine learning models in terms of accuracy, achieving a training R^2 score of 0.81 and a testing R^2 score of 0.72.

Despite the relatively short duration of the dataset (2 years and 3 months), the MLPRegressor model effectively captured the underlying patterns and provided accurate sales forecasts.

The final model achieved a submission score of “670.73”, demonstrating the effectiveness of neural networks for sales forecasting.

14.Lessons Learnt:

The importance of feature engineering, preprocessing, and hyperparameter tuning in improving model performance.
Neural Networks analysis, particularly with the MLP Regressor model, proved to be more suitable for this forecasting task due to the dataset's structure.
Regularly refreshing and re-running the code, along with fine-tuning hyperparameters, significantly improved model performance.
Incorporating economic indicators into the forecasting process may provide additional insights and improve the accuracy of predictions.

15.Recommendations for Future Work:

Explore advanced time series models such as SARIMA (Seasonal AutoRegressive Integrated Moving Average) and Prophet for further improvement in forecasting accuracy.

15.1 Advanced Neural Network Architectures:

Investigate more complex neural network architectures such as Long Short-Term Memory (LSTM) networks and Gated Recurrent Unit (GRU) networks, which are well-suited for modeling sequential data and may capture long-term dependencies better than traditional feedforward networks.

15.2 Feature Engineering:

Explore additional features such as economic indicators, customer demographics, promotional activities, and external factors like weather conditions, holidays, and events that may influence sales.
Investigate feature engineering techniques such as lag features, rolling statistics, and exponential smoothing to capture the trends and seasonality in the data more effectively.
Ensemble Learning:
Explore ensemble learning techniques such as stacking, blending, and bagging to combine the predictions of multiple models and improve overall performance.
Implement model stacking with a combination of different types of models (e.g., linear models, tree-based models, neural networks) to leverage the strengths of each model.

15.3 Advanced Hyperparameter Optimization:

Explore advanced hyperparameter optimization techniques such as Bayesian optimization, genetic algorithms, and evolutionary strategies to efficiently search the hyperparameter space and find the optimal set of hyperparameters.

15.4 Cross-Validation Strategies:

Explore different cross-validation strategies such as time series cross-validation (e.g., rolling-window cross-validation, expanding-window cross-validation) to evaluate model performance more accurately on time-series data.
By exploring these avenues for future work, we can further enhance the accuracy, robustness, and scalability of the sales forecasting system, leading to better business insights and decision-making.

16.AI Contributions Summary:

16.1 Data Preprocessing:

Handling Missing Values: ChatGPT provided guidance on handling missing values in the 'InventoryRatio' column using Simple Imputer. When prompted with a specific scenario where missing values needed to be replaced with the mean of each company's values for different quarters, ChatGPT provided the necessary code.

Example Prompt: "If the feature 'Company' has categorical values CMP1 to CMP75 and feature 'Quarter' has categorical values Q1 to Q9 for each CMP, I need the code to fill missing values in the feature 'InventoryRatio' by taking the mean of each company's Q1 to Q9 values of the inventory ratio. If the inventory ratio is not available for any company then leave it blank."

Scaling Techniques: Additionally, ChatGPT recommendeded better scaling techniques such as Power transforms (Yeo-Johnson or Box-Cox transforms) when prompted for methods other than Standard Scaling but I proceeded to use minmax scaler and Standard scaler.

16.2 Model Selection and Training:

Regression Models Evaluation: ChatGPT provided guidance on evaluating various regression models including Linear Regression, Ridge Regression, Lasso Regression, Gradient Boosting Regressor, and Random Forest Regressor through experimentation and hyperparameter tuning.
Example Prompt: "Perform 'Model name' using 'sales_df' and how can I perform hyperparameter tuning for random forest."
Cross-Validation and Bootstrapping: ChatGPT recommended and provided modified codes for performing k-fold cross-validation and bootstrapping to assess model performance and address overfitting when prompted to modify existing code.

Example Prompt: "Modify this code to use bootstrapping."
Overall, ChatGPT's contributions included offering valuable insights and recommendations at various stages, facilitating the exploration and optimization of predictive models for Sales Forecasting.

16.3 Hyperparameter Tuning with GridSearchCV:

GridSearchCV Implementation: ChatGPT assisted in implementing hyperparameter tuning using GridSearchCV for models like RandomForestRegressor and XGBoost. GridSearchCV was utilized to efficiently search the hyperparameter space and find the optimal set of hyperparameters, thereby improving the model's performance.

Example Prompt: "How can I perform hyperparameter tuning using GridSearchCV for RandomForestRegressor?"

MLPRegressor (Neural Networks):

Optimization Techniques: ChatGPT provided guidance on optimizing the MLPRegressor (Neural Networks) model. This included experimenting with different architectures, activation functions, solvers, and regularization techniques to enhance model performance.

Example Prompt: "How can I improve the performance of MLPRegressor?"

Overall, ChatGPT's contributions in implementing hyperparameter tuning and optimizing the MLPRegressor model significantly improved the accuracy and robustness of the predictive model for sales forecasting.

17.Conclusion:

The project successfully demonstrated the effectiveness of machine learning models for sales forecasting. The MLPRegressor model, in particular, outperformed other models and provided accurate sales forecasts. Further optimization and fine-tuning of hyperparameters significantly improved model performance, highlighting the importance of iterative model development and evaluation.

Project Contributor: ABHISHEK MALLA (VB7520)

Submission Score: “670.73”

SUBMISSIONS OVERVIEW AND LESSONS FROM EACH SUBMISSION:

ID:	Overview of the Code:	Public Rank:	Submission Score:	Training R^2 score:	Testing R^2 score:	Lesson learnt after submission:
1	Took care of missing values in the "InventoryRatio" column by filling them with the average value and scaling the data.	15	2286.3	0.67	0.54	Filling missing values with the average can help maintain data integrity and prevent bias in the model.
2	Encoded categorical columns like 'Bond rating', 'Stock rating', 'Region', 'Industry', and 'Company' using one-hot encoding.	15	2283.5	0.68	0.57	Using one-hot encoding effectively transforms categorical variables into a format suitable for machine learning algorithms.
3	This code snippet helped me split the data into training and testing sets and preprocesses the training data using the preprocessor defined earlier.	15	2243.4	0.67	0.63	Splitting data into training and testing sets, along with preprocessing, is essential for training and evaluating models.
4	Created a machine learning pipeline with a preprocessor and a Linear Regression model, allowing for seamless data preprocessing and model training in a single step.	13	1575.6	0.69	0.67	Building a pipeline that combines data preprocessing and model training simplifies the workflow and improves reproducibility.
5	This code fits the pipeline to the training data, allowing the model to learn patterns and relationships between the features and the target variable. After conducting a grid search, the RandomForestRegressor was employed with optimized parameters, including a max depth of None, a minimum samples split of 5, and 150 estimators. Additionally, the StandardScaler was configured with with mean set to False.	13	1538.7	0.68	0.6	Experimenting with various models, such as Linear Regression and RandomForestRegressor, helps identify the best model for the dataset. Evaluating models using appropriate metrics, such as Mean Absolute Error (MAE), provides insights into their performance.
6	This code segment splits the dataset into features (X) and the target variable (y). It then defines a preprocessing pipeline using ColumnTransformer, which applies mean imputation and scaling to numerical features and one-hot encoding to categorical features. Finally, it prepares the data for training and testing by splitting it into training and testing sets (X_train, X_test, y_train, y_test).	12	1259.4	0.65	0.61	Splitting data into features (X) and target variable (y) is crucial for supervised learning. ColumnTransformer enables different preprocessing for numerical and categorical features. Mean imputation, scaling, and one-hot encoding ensure data is ready for machine learning.
7	This code creates a pipeline that preprocesses the data using the	10	1015.5	0.68	0.65	Ensemble methods like Gradient Boosting can

	preprocessor defined earlier and then fits the Gradient Boosting model (`gb_model`). After fitting the pipeline to the training data, it makes predictions on the test data. Finally, it evaluates the model using the Mean Absolute Error (MAE) metric and prints the result.					further enhance model performance compared to individual models.
8	Created similar pipeline as the above submissions , here I have used Random forest regressor implemented Grid SearchCV (n_estimators, max_features, max_depth, min_samples_split, min_samples_leaf) with cross-validation	8	960.1	0.72	0.68	Tuning hyperparameters using techniques like GridSearchCV helps find the best model configuration for improved performance.
9	I imported additional economic indicator data and merged it with the training and test datasets using the common column "Quarter". Since the economic data had a "Month" column, I transformed it to align with the quarters in the training and test datasets by converting months to quarters using the expression `economic_data['Month'] = economic_data['Month'].apply(lambda x:'Q'+str((x+2)//3))`. Finally, I implemented linear regression using this merged dataset.	8	947.92	0.73	0.69	Incorporating additional economic indicator data can enhance the predictive power and robustness of the model.
10	Identified categorical columns and defines a preprocessor using ColumnTransformer. It applies mean imputation and scaling to numerical features and one-hot encoding to categorical features, while passing through any unspecified columns without transformation.	8	927.98	0.75	0.68	Identifying and preprocessing categorical columns is essential for machine learning. ColumnTransformer allows different preprocessing for numerical and categorical features. Mean imputation, scaling for numerical data, and one-hot encoding for categorical data ensure comprehensive data preprocessing.
11	Defined an MLPRegressor model with the following specifications: Hidden Layer Sizes: Three hidden layers with 150, 100, and 50 neurons respectively. Activation Function: Rectified Linear Unit (ReLU) activation function. Solver: Adam optimizer, which is an extension to stochastic gradient descent. Maximum Iterations: The maximum number of iterations set to 1000, which	7	897.98	0.77	0.64	Using deep learning models like MLPRegressor allows capturing complex patterns in the data.

	determines the number of iterations the model will undergo during training.					
12	Created a machine learning pipeline by combining the preprocessor and the MLPRegressor model. The pipeline will first preprocess the data using the preprocessor and then apply the MLPRegressor model for prediction.	7	818.1	0.79	0.68	Optimizing model parameters, including the number of iterations, helps improve convergence and model performance.
13	I made enhancements to the MLPRegressor model by adjusting its architecture and parameters. I increased the size of the hidden layers to (150, 100, 50) to allow for more complex pattern capturing.I changed the activation function to ReLU for better handling of non-linear relationships. I switched the optimizer to Adam for its efficiency and fast convergence.I increased the maximum number of iterations to 1000 to allow for more training epochs, enhancing the model's learning capability.	7	754.7	0.76	0.68	Experimenting with different architectures and parameters, such as hidden layer sizes and activation functions, can significantly impact model performance.
14	I incorporated StandardScaler to scale the numerical features and utilized a ColumnTransformer to preprocess various feature types. Subsequently, I constructed a pipeline integrating numerical and categorical features, feeding them into the MLP regressor.	9	741.24	0.77	0.69	StandardScaler and ColumnTransformer can be used for effective feature preprocessing, enhancing model performance.
15	After integrating StandardScaler and ColumnTransformer to preprocess features, I employed GridSearchCV to optimize the MLPRegressor's parameters, ensuring the best model performance.	9	732.33	0.78	0.68	Using GridSearchCV to optimize model parameters ensures the best model performance.
16	I refreshed the kernel and reran all the blocks with consistent parameters.	10	725.12	0.82	0.69	-
17	“	11	722.62	0.84	0.68	-
18	After refreshing with some changes in parameters, I achieved a better score. I utilized only the MLPRegressor to improve the model's performance.	8	706.84	0.81	0.77	Iteration and continuous parameter tuning result in better model performance across multiple submissions.
19	“	6	695.98	0.85	0.69	-

20	Change in hidden layers and iterations:
	# Define the MLPRegressor model
	mlp_regressor = MLPRegressor(
	hidden_layer_sizes=(150, 100, 50), # Architecture of the neural network
	activation='relu', # Activation function
	solver='adam', # Optimizer
	max_iter=1000)
		7	689.81	0.87	0.73	-

21	Change in hidden layers and iterations:
	
	# Define the MLPRegressor model with modified parameters
	mlp_regressor = MLPRegressor(
	  hidden_layer_sizes=(175, 75, 100), # Adjusted architecture
	  activation='relu', # Changed activation function
	  solver='adam', # Optimizer
	  max_iter=875 # Increased maximum number of iterations
	)
		6	687.43	0.83	0.71	-

22	Change in hidden layers and iterations:
	
	# Define the MLPRegressor model with modified parameters
	mlp_regressor = MLPRegressor(
	  hidden_layer_sizes=(185, 75, 100), # Adjusted architecture
	  activation='relu', # Changed activation function
	  solver='adam', # Optimizer
	  max_iter=1100 # Increased maximum number of iterations
	)
		6	695.98	0.79	0.76	Overfitting Awareness: Monitoring and mitigating overfitting risks by evaluating the model's performance on both training and testing data to ensure generalizability.

23	Change in hidden layers and iterations:
	
	# Define the MLPRegressor model with modified parameters
	mlp_regressor = MLPRegressor(
	  hidden_layer_sizes=(190, 90, 100), # Adjusted architecture
	  activation='relu', # Changed activation function
	  solver='adam', # Optimizer
	  max_iter=850 # Increased maximum number of iterations
	)
		6	696.83	0.81	0.7	-

24	Change in hidden layers and iterations:
	
	# Define the MLPRegressor model with modified parameters
	mlp_regressor = MLPRegressor(
	  hidden_layer_sizes=(160, 75, 90), # Adjusted architecture
	  activation='relu', # Changed activation function
	  solver='adam', # Optimizer
	  max_iter=850 # Increased maximum number of iterations
		6	688.21	0.86	0.69	Iterative Parameter Optimization: The process of adjusting hidden layers and the number of iterations iteratively significantly impacted the model's performance.

25	Change in hidden layers and iterations:
	
	# Define the MLPRegressor model with modified parameters
	mlp_regressor = MLPRegressor(
	  hidden_layer_sizes=(160, 75, 100), # Adjusted architecture
	  activation='relu', # Changed activation function
	  solver='adam', # Optimizer
	  max_iter=830 # Increased maximum number of iterations
	)
		6	683.64	0.89	0.7	-

26	Change in hidden layers and iterations:
	
	# Define the MLPRegressor model with modified parameters
	mlp_regressor = MLPRegressor(
	  hidden_layer_sizes=(190, 79, 101), # Adjusted architecture
	  activation='relu', # Changed activation function
	  solver='adam', # Optimizer
	  max_iter=870 # Increased maximum number of iterations
	)
		6	680.91	0.87	0.74	-

27	Change in hidden layers and iterations:
	# Define the MLPRegressor model with modified parameters
	mlp_regressor = MLPRegressor(
	  hidden_layer_sizes=(185, 80, 100), # Adjusted architecture
	  activation='relu', # Changed activation function
	  solver='adam', # Optimizer
	  max_iter=850 # Increased maximum number of iterations
	)
		6	675.33	0.85	0.73	Considering deployment constraints, such as model interpretability and computational resources, is crucial when choosing the final model architecture.

28	Change in hidden layers and iterations:
	# Define the MLPRegressor model with modified parameters
	mlp_regressor = MLPRegressor(
	  hidden_layer_sizes=(185, 80, 100), # Adjusted architecture
	  activation='relu', # Changed activation function
	  solver='adam', # Optimizer
	  max_iter=850 # Increased maximum number of iterations
	)
		6	676.26	0.82	0.72	-

29	Change in hidden layers and iterations:
	# Define the MLPRegressor model with modified parameters
	mlp_regressor = MLPRegressor(
	  hidden_layer_sizes=(185, 80, 100), # Adjusted architecture
	  activation='relu', # Changed activation function
	  solver='adam', # Optimizer
	  max_iter=850 # Increased maximum number of iterations
	)
		6	683.04	0.87	0.69	Balancing Complexity: Finding the right balance between model complexity (hidden layers) and computational efficiency (iterations) is crucial for achieving optimal results.

30	Change in hidden layers and iterations:
	# Define the MLPRegressor model with modified parameters
	mlp_regressor = MLPRegressor(
	  hidden_layer_sizes=(185, 80, 100), # Adjusted architecture
	  activation='relu', # Changed activation function
	  solver='adam', # Optimizer
	  max_iter=850 # Increased maximum number of iterations
	)
		6	674.46	0.85	0.7	-

31	Change in hidden layers and iterations:
	# Define the MLPRegressor model with modified parameters
	mlp_regressor = MLPRegressor(
	  hidden_layer_sizes=(185, 80, 100), # Adjusted architecture
	  activation='relu', # Changed activation function
	  solver='adam', # Optimizer
	  max_iter=850 # Increased maximum number of iterations
	)
		6	680.7	0.80	0.71	-

32	This is the highest score I've achieved using the MLPRegressor, after finding the optimal parameters for the model.
	
	# Define the MLPRegressor model with modified parameters
	mlp_regressor = MLPRegressor(
	  hidden_layer_sizes=(180, 75, 100), # Adjusted architecture
	  activation='relu', # Changed activation function
	  solver='adam', # Optimizer
	  max_iter=850 # Increased maximum number of iterations
		6	670.73	0.81	0.72	Regular Evaluation: Continuously evaluating the model's performance and fine-tuning the parameters is essential for improving predictive accuracy.