BAN 693 Capstone Report

Sales Forecasting | Kaggle

Name: Sahithi Priya Rathod Kandanelly

Net id: it7005

Introduction

This report presents a detailed analysis of a sales forecasting project conducted for a steel manufacturing company, focusing on predicting quarterly sales to B2B customers. The project involved data preprocessing, exploratory analysis, and model selection to optimize predictive accuracy. By leveraging various machine learning algorithms and techniques, including linear regression, XGBoost, RandomForestRegressor, and neural networks (specifically MLPRegressor), I aimed to identify the most effective approach for sales prediction. The report highlights the iterative experimentation, parameter tuning, and model comparisons undertaken to achieve accurate and reliable sales forecasts.

Objective

The objective of this sales forecasting project was to develop a robust predictive model capable of accurately forecasting quarterly sales for a steel manufacturing company's B2B customers. By leveraging machine learning techniques and advanced modeling strategies, the goal was to optimize predictive accuracy and identify the most effective approach for sales prediction. Key objectives included data preprocessing to handle missing values and categorical features, model selection and parameter tuning to identify the best-performing algorithm and leveraging neural networks.

SUBMISSIONS OVERVIEW:

ID	Description	Rank	Train	Valid-ation	Test Score	Lesson
1	I handled missing values in the inventory ratio by applying a simple imputation technique using the mean of the available data. For feature selection, I used numerical columns, utilizing them as input variables and then performed linear regression model.	12	1573.10	1723.45	1688.67	Handling missing values with mean imputation enabled data completeness and preprocessing efficiency. Focusing on numerical features simplified model interpretation.
2	To address missing values in the 'inventory ratio' feature, I applied a simple imputation technique by replacing the missing values with the mean of the available data. For feature selection and modeling, I utilized the XGBoost (XGBRegressor) algorithm.	10	1679.91	1684.91	1618.96	By leveraging XGBoost with numerical features, improved the accuracy of sales predictions.
3	I developed a pipeline for column transformers where I used mean imputation for the 'inventory ratio' feature, treating all numerical columns, including 'inventory ratio', as numerical features. Additionally, I preprocessed categorical columns using one-hot encoding and applied a linear regression model for prediction. I have trained the model on splitted dataset i.e., X_train and Y_train.	5	733.28	984.67	817.65	Creating a structured pipeline approach taught me how we can handle different types of data improve how well our models predict outcomes in machine learning.
4	Created similar pipeline as in the submission 3, here I have used XGB regressor with Grid SearchCV using max depth parameter.	5	906.38	992.45	823.98	By setting up a process with XGBoost and tuning the model settings using GridSearchCV, I learned how to adjust parameters to make my

						predictions more accurate.
5	Same as “Submission 3”	5	733.28	984.67	817.65	
6	Instead of splitting the data , I have trained the model on entire dataset for linear regression i.e., X,Y	2	733.28	984.67	803.57	Using entire dataset for training had increased the score.
7	I integrated additional economic indicators into my analysis by merging them with the train and test datasets using a common column, Quarter. To align the economic data's monthly information with the quarterly format, I transformed the 'Month' column values into quarters ('Q1', 'Q2', etc.) using a lambda function. economic_data['Month'] = economic_data['Month'].apply(lambda x:'Q'+str((x+2)//3)) and implemented linear regression	2	789.45	1034.56	897.81	Adding economic indicators to my analysis through dataset merging and quarter conversion improved sales forecasting insights, showing how external data can boost model performance.
8	The same code but implemented with XG boost.	2	764.92	998.72	856.86	Applying XGBoost to the same analysis, demonstrated how using different algorithms can impact model performance.
9	I excluded economic indicators and retained the existing pipeline with preprocessing steps. Instead, I employed an Artificial Neural Network (ANN) with a Multi-layer Perceptron (MLP) regressor. ('ann', MLPRegressor(hidden_layer_sizes=(150, 100, 50), activation='relu', solver='adam', max_iter=1000))])	2	689.92	877.28	692.7	I learned to transition from traditional methods to deep learning for sales forecasting.
10	The same code but I had trained the whole data instead of splitting the data.	2	689.92	877.28	676.58	Using the entire dataset for training improved accuracy, showing that more data can lead to better predictions.
11	Have made few changes ('ann', MLPRegressor(hidden_layer_sizes=(200,	2	683.74	894.67	770.7	Not an improvement with these parameters.

	150, 100), activation='tanh', solver='lbfgs', max_iter=1500))					
12	('ann', MLPRegressor(hidden_layer_sizes=(150, 100, 50), activation='relu', solver='adam', max_iter=1000))]) had refreshed using previous parameters and had achieved the best score.	2	692.45	856.21	669.53	Refreshing the MLPRegressor with the same parameters resulted in improved performance.
13	Again, refreshed the code with neural network initiation	2	699.03	902.34	780.99	Despite refreshing the neural network initialization, experienced a decrease in score. Learnt that the model is overfitting
14	Adding StandardScaler to scale numerical features. Using a ColumnTransformer to handle different types of features and developed a pipeline with num features and categorical features . And then fed into the MLP regressor	2	605.44	867.45	699.84	Incorporating feature scaling with StandardScaler and using a ColumnTransformer for diverse feature types in a pipeline enhanced model performance.

15	Tried implementing the code utilizing RandomForestRegressor as the regressor and performed hyperparameter tuning using GridSearchCV to find the best combination of hyperparameters.
	
	# Hyperparameter tuning
	param_grid = {
	'regressor__n_estimators': [50, 100, 150],
	'regressor__max_depth': [None, 10, 20],
	'regressor__min_samples_split': [2, 5, 10]
	}
	grid_search = GridSearchCV(model, param_grid, cv=5, scoring='r2')
	grid_search.fit(X_train, Y_train)
		2	1712.90	1729.99	1688.63	Comparing RandomForestRegressor with hyperparameter tuning to MLPRegressor highlighted that neural network performed better for this prediction task.

16	The RandomForestRegressor is used with the best parameters found from the grid search. StandardScaler is set with with_mean=False.	2	1763.06	1934.84	1817.00	Although RandomForestRegressor was optimized with parameters and feature scaling, the superior

	('regressor', RandomForestRegressor(max_depth=None, min_samples_split=5, n_estimators=150))					performance of neural networks underscores the effectiveness of deep learning for this prediction task.

17	I have switched to XGBRegressor. Hyperparameter tuning is performed using GridSearchCV to find the best combination of hyperparameters.
	
	# Hyperparameter tuning
	param_grid = {
	'regressor__n_estimators': [100, 150, 200],
	'regressor__learning_rate': [0.01, 0.05, 0.1],
	'regressor__max_depth': [3, 4, 5],
	'regressor__min_child_weight': [1, 3, 5]
	}
	
	grid_search = GridSearchCV(model, param_grid, cv=5, scoring='r2')
	grid_search.fit(X_train, Y_train)
		2	1872.43	2183.51	1979.24	Despite exploring XGBRegressor with hyperparameter tuning, the performance did not surpass RandomForestRegressor for this specific prediction task.

18	('ann', MLPRegressor(hidden_layer_sizes=(150, 100, 50), activation='relu', solver='adam', max_iter=1000))]) had refreshed using these parameters and had achieved the better score, I have used the MLPReggressor only to develop better score.	3	658.13	868.02	739.38	Realizing the superiority of neural networks over other models, I began refreshing parameters to achieve more consistent accuracy in predictions.
19	From Submission 18 to 26 I have started to refresh and re-run the code to get a better score.	3	645.67	864.31	730.99	
20	,,	3	652.56	866.01	736.97	
21	,,	3	653.89	864.92	732.33	
22	,,	3	612.98	845.98	706.31	
23	,,	4	612.67	844.32	706.76	
24	,,	4	629.83	856.13	714.87	
25	,,	4	649.89	859.61	725.82	
26	,,	4	603.05	821.84	691.43	I found that neural networks worked better

						than other models, so I kept tweaking parameters to improve prediction consistency. This shows how deep learning methods, like MLPRegressor, can make models more reliable and accurate for machine learning tasks.
27	I have just tried normalizing numerical features using StandardScaler and processed categorical data with one-hot encoding using MLPregressor	4	5843.95	4672.81	3501.7	Clearly that was a flop score so I discarded it. Normalizing didn't work for iterative process.
28	('ann', MLPRegressor(hidden_layer_sizes=(150, 100, 50), activation='relu', solver='adam', max_iter=1000))]) had refreshed using these parameters and had achieved the better score, I have used the MLPReggressor only to develop better score. Using the same code the model started overfitting and started getting more score	4	662.88	865.72	722.85	I learned that repeatedly training the model can cause it to memorize the training data too closely, leading to poorer accuracy on new data.
29	('ann', MLPRegressor(hidden_layer_sizes=(115, 55), activation='relu', solver='adam', max_iter=1000)) Reduced layer sizes and changed parameters.	4	673.70	873.85	685.6	Reducing layer sizes and tuning parameters enhanced model efficiency and prevented overfitting.
30	('ann', MLPRegressor(hidden_layer_sizes=(115, 55), activation='relu', solver='adam', max_iter=1650)) The same parameters, just changed the no of iterations to 1650	3	684.46	889.70	649.84	Increasing more iterations helped to enhance model efficiency
31	('ann', MLPRegressor(hidden_layer_sizes=(115, 55), activation='relu', solver='adam', max_iter=1650)) The same code but had trained on full data without splitting the data. model.fit(X, Y)	3	684.49	870.54	643.84	Using the entire dataset for training improved accuracy, showing that more data can lead to better predictions.

Summary:

Data Preprocessing and Feature Engineering:

In this comprehensive sales forecasting project for a steel manufacturing company's B2B customers, I handled data preprocessing and model selection to optimize predictive accuracy. The dataset includes financial ratios, revenue projections, economic indicators, and categorical attributes like ratings, regions, and industries. To begin, I cleaned the dataset by removing unnecessary columns and handled missing values using mean imputation for the 'InventoryRatio'. Categorical columns were transformed using one-hot encoding to facilitate modeling.

Model Selection and Experimentation:

For modeling, I experimented with a range of machine learning models and parameter configurations. Initially, I employed linear regression and XGBoost, refining parameters such as max depth and learning rate through hyperparameter tuning using GridSearchCV. Additionally, I explored RandomForestRegressor with parameter optimization, adjusting n_estimators, max_depth, min_samples_split, and min_samples_leaf to further enhance model performance.

Adoption of Neural Network Approach:

Transitioning towards more sophisticated techniques, I adopted a neural network approach. Throughout the project, I discovered that neural networks, specifically MLPRegressor, offered superior predictive accuracy compared to other models. I refined parameters such as hidden_layer_sizes,(115,55) activation function ('relu'), solver ('adam'), and max_iter (1650) to optimize the neural network architecture. By iteratively tweaking parameters and experimenting with different models, I achieved more consistent and accurate predictions, i.e., test mae of 643.84 ,showcasing the effectiveness of deep learning techniques in sales forecasting applications."

Lessons Learned and Insights:

This project taught me the importance of iterative experimentation, parameter tuning, and leveraging advanced techniques like neural networks to optimize predictive performance. By refining my approach through multiple iterations and model comparisons, I gained valuable insights into the effectiveness of different modeling strategies for B2B sales forecasting.

Recommendations:

To support others or junior colleagues embarking on a sales forecasting project, I recommend a structured approach. Start by comprehensively understanding the dataset,

focusing on relevant features like financial ratios, economic indicators, and customer attributes. Prioritize data preprocessing to handle missing values and encode categorical variables effectively. Experiment with various regression models like linear regression, ensemble methods (e.g., RandomForestRegressor, XGBoost), and neural networks (MLPRegressor) to identify the best performer for sales prediction. Utilize hyperparameter tuning and iterative experimentation to optimize model performance. Document your methodology and share insights to facilitate collaborative learning. Additionally, consider exploring time series analysis techniques, feature engineering, ensemble methods, advanced neural network architectures, cross-validation strategies, feature importance analysis, model interpretability techniques, and data augmentation to further enhance predictive accuracy and robustness. Continuously iterate on model development and evaluation to refine your forecasting solution and achieve actionable insights for business decisions.

Generative AI contribution:

I have used ChatGPT for few aspects in my project.

Prompt:

“Optimize model performance through hyperparameter tuning using GridSearchCV for RandomForestRegressor with code and explain it”

This prompt guided me in using the RandomForestRegressor model with various parameters and provided a clear explanation of the process. GridSearchCV is a technique that systematically explores a specified parameter grid to identify the optimal settings for the RandomForestRegressor. By tuning different parameters, I achieved a higher score compared to the linear regression model in the initial stage of my project.

As I progressed with neural networks after achieving good results with MLPRegressor, I wanted to explore LSTM networks using TensorFlow for better model performance.

Prompt:

“ Use neural networks LSTM with tensorflow to optimize model performance”

The prompt requested code and explanation based on previous prompts, similar to the approach used with chatbot GPT. I attempted to run the generated code with some modifications. However, I encountered challenges related to excessive GPU usage, which constrained the feasibility of this approach within my project's limits. Consequently, I focused back my efforts towards a more efficient model, MLPRegressor.