BAN 693

PROJECT SUMMARY REPORT

Sales forcasting kaggle challenge

CONTRIBUTER: VENKATESH VUYYALA

PROFESSOR: DR. SURENDRA SARNNIKAR

NETID: KX8516

MAIL ID: vvuyyala@horizon.csueastbay.edu

Public Rank: 2. Public Score: 630.36

INTRODUCTION

Accurate sales forecasting is essential for effective decision-making and strategic planning in business operations. For B2B companies like steel manufacturers, predicting quarterly sales for different customers is crucial. This project endeavors to forecast quarterly sales for each of the 75 customers through innovative methodologies. The objective is to develop predictive models proficient in accurately estimating sales figures over quarterly periods. Utilizing the dataset provided by a Kaggle competition, a thorough analysis was conducted. Multiple machine learning techniques, including Linear Regression, RandomForest, Support Vector Machine (SVM), Bayesian Ridge Regression, Ridge Regression, XGBoostRegression, Gradientboost Regressor, MLPRegressor, CatBoost, and Neural Networks, were employed. Through extensive exploration of algorithms and features, the aim was to strike a balance between model stability and performance, crucial for real-world applicability

DATASET AND FEATURES

The dataset comprises a mix of synthetic and real data from a steel manufacturer, with customers in the Auto, Metal Fabrication, and Infrastructure sectors.

Train.csv: Contains company-specific data for the steel manufacturer's 75 customers. This dataset has 12 columns and 675 Rows.

Test.csv: Contains company-specific data for the steel manufacturer's 75 customers. This dataset has 11 columns and 150 Rows.

EconomicIndicators.csv (Economic information): Provides various economic indicators at a monthly level, which can be used in conjunction with company-specific data for sales prediction.

Features:

• **ID** - Row id column

• **Company** - Name of the company/customer

• **Quarter** - Quarter for which the sales are provided/to be predicted

• **QuickRatio** - Financial ratio indicating the customer's liquidity situation

• **InventoryRatio** - Ratio of sales over inventory

• **RevenueGrowth** - Revenue growth projections based on analyst and company projections

• **MarketshareChange** - 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)

• **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

OBJECTIVE

The objective of this project is to develop an accurate predictive model capable of forecasting quarterly sales for each of the 75 customers of a steel manufacturing company. In addition to leveraging company-specific data, the model will utilize general economic indicators.

PROJECT BACKGROUND

In this project, our primary objective is to forecast sales for the upcoming two quarters, namely Q8 and Q9, leveraging historical sales data spanning a timeframe of two years and three months. Our dataset encompasses sales records for a total of 75 customers, offering a comprehensive understanding of sales dynamics across diverse sectors.

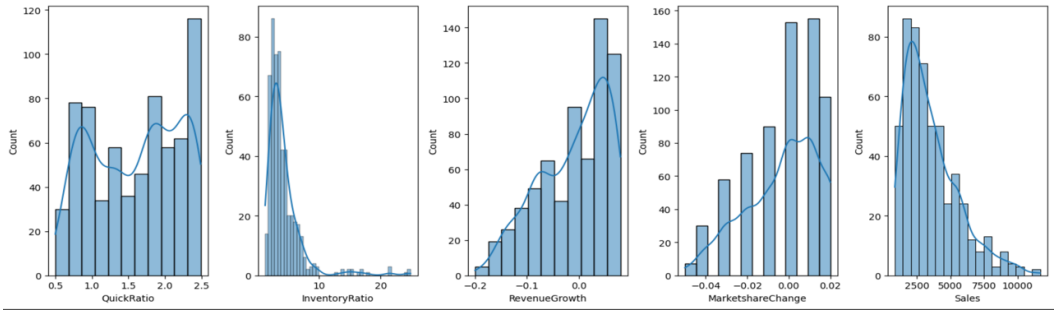
METHODOLOGY

1) Data Preprocessing:

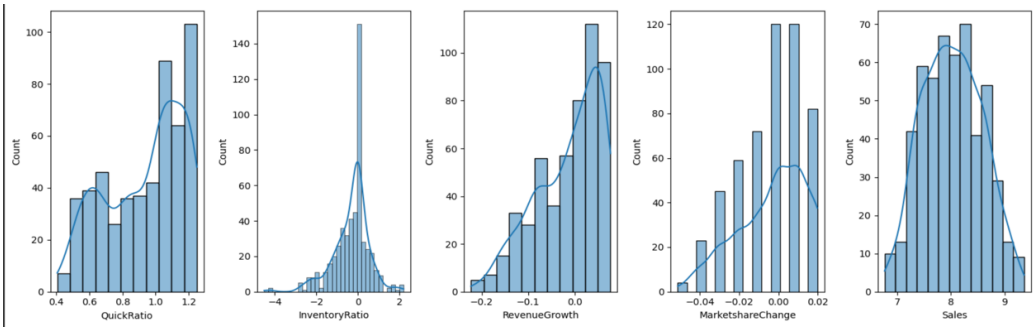
- Identified and addressed missing values in Inventory Ratio using different techniques (mean imputation, KNN Imputer = 2) and found that KNN Imputation has improved the model's performance. Considerd using the KNN imputer = 2 for the missing values in rows for the sales as there is less data available for Training and dropping the sales rows of around 150 would considerably missing a significant amount of data for training.
- Handled Outliers through visualizing the box plots and used L2 Regularization to remove the outliers. But It didn't performed well on machine Learning models like RandomForestRegressor, XGBRegressor. But it improved the model's performance on neural networks by implementing Kernel Regularization L2.
- Implemented data scaling methods such as MinMaxScaler, StandardScaler to normalize numerical features and stabilize model performance.
- Used Feature Engineering techniques like creating, dummies for columns containing categorical data using One Hot Encoding.
- Incorporated additional dataset containing economic indicators and merged it with the Train data and Test data based on the "Quarter" column by converting month column in economic indicators to the quarters.

2) Data Visualization:

- Created sub plots to display distribution plots for numerical columns and identified there was no standard normal distribution.



- Applied Log-transform to the numerical for the distribution plots to reduce the skewness of the data.



- Created Heat Map and Correlation matrix between Features and sales.

- After performing Feature importance through Decision Tree and through the heat map and correlation matrix, found that There was no strong correlation found between features and Sales.

3) Model Evaluation:

- Evaluated model performance using metrics such as Mean Absolute Error (MAE), Training R^2/Testing R^2 and Root Mean Squared Error (RMSE).

4) Model selection and optimization:

- **Linear Regression:** Used a fundamental statistical method that models the relationship between a dependent variable (sales) and independent variables (features).
- **Random Forest Regression with GridSearchCV:** Performed an ensemble learning method that leverages the power of multiple decision trees. an ensemble learning method that leverages the power of multiple decision trees. used GridSearchCV to find the best hyperparameters for the Random Forest model. The grid search tuned hyperparameters like the number of estimators (n_estimators), maximum depth of trees (max_depth), minimum samples required to split a node (min_samples_split), and minimum samples required at each leaf node (min_samples_leaf). Evaluated the model performance using r2_score on the training and testing sets.
- **Ridge Regression**: This model is similar to linear regression but uses L2 regularization to reduce the model's complexity and prevent overfitting. imported Ridge from sklearn.linear_model and performed GridSearchCV to find the optimal regularization parameter (alpha).Evaluated the model performance using r2_score on the training and testing sets.
- **CatBoost Regression Model:** This model uses gradient boosting, a powerful machine learning technique that combines weak learners (decision trees) into a sequentially stronger learner Defined the loss function as RMSE (root mean squared error) which emphasizes punishing large errors. Set the number of iterations (iterations=3000) and learning rate (learning_rate=0.07) through hyperparameter tuning. Evaluated the model performance using r2_score on the testing set.
- **XGBRegressor:** This is similar to catboost Regression model, Performed similar datapreprocessig techniques and An XGBoost regression model (XGBRegressor) is initialized and trained on the training data (X_train, Y_train).The model's performance is evaluated using the R-squared coefficient (score) on both the training and testing sets. Another XGBoost model is trained on the entire dataset (X, Y) without splitting.
- **MLPRegressor with GridSearchCV:** This model implements a Multi-Layer Perceptron (MLP), a type of artificial neural network with multiple hidden layers.
  imported MLPRegressor from sklearn.neural_network. Performed GridSearchCV
  using GridSearchCV to find the optimal hyperparameters for the MLP model. The grid search explored different combinations of hidden layer sizes [(100,199), (200,300), (50, 100)], activation functions (ReLU), solvers (Adam), and regularization parameters (alpha). The Best hyperparameters were {'activation': 'relu', 'alpha': 0.01, 'hidden_layer_sizes': (200, 300), 'solver': 'adam'}.
- **MLPRegressor:** Implemented a Multi-Layer Perceptron (MLP), a type of artificial neural network with only two hidden layers. hidden_layer_sizes=(115,55), activation function (ReLU), solver(Adam) and Maximum iterations were 1500.
- **Neural Networks using Sequential Model**: The model builds a sequential neural network architecture from layers. imported necessary libraries from tensorflow.keras.Defined a sequential model with Dense

layers, each with L2 regularization (kernel_regularizer='l2'). L2 regularization penalizes the model for having large weights, preventing overfitting. Defined a custom root mean squared error loss function (root_mean_squared_error). Trained the model using the Adam optimizer (opt=tf.keras.optimizers.Adam(lr=0.01)) and early stopping (early_stopping_monitor) to prevent overfitting.

RESULTS AND DISCUSSION

- Utilizing Linear Regression yielded its best score of 784.26, followed by RandomForestRegressor with a score of 755.69. However, XGBRegressor resulted in a lower metric score compared to the other algorithms. The use of diverse algorithms allowed for capturing various facets of the data, presenting opportunities for enhancing overall score.
- Upon discovering overfitting in the model, Ridge Regression was employed with hyperparameter tuning, resulting in a score of 774. However, this adjustment did not yield significant improvements and resulted in a lower score compared to RandomForest.
- Employing the MLP Regressor model gave better results with score 650.11. Trying different parameters for tuning contributed to the optimized model performance.
- Finally Developing Neural Network Sequential model and including L2 regularization to reduce the overfitting and adding early stopping monitor and lastly trying different parameters for tuning produced the highest score of 630.36 surpassing my previous models.

LESSONS LEARNED

Key takeaways and Insights:

- Iterative process: The project involved an iterative process of data exploration, modeling, and evaluation to refine the predictive models continuously.
- Importance of experimentation: Trying out different models and techniques helped in understanding which approaches worked best for the given dataset.
- Interpretability vs. complexity: Balancing model complexity with interpretability was crucial, considering the trade-off between model performance and explainability.

Project Documentation and Reporting:

- Clear documentation: Maintaining clear and concise documentation of data preprocessing steps, model building, and evaluation helped in understanding the workflow and reproducibility.
- Report writing: Summarizing key findings, insights, and lessons learned in a report format helps in communicating the outcomes of the project.

ID	DESCRIPTION	TRAIN r^2/MAE	TEST r^2/MAE	RANK	SUBMISSION SCORE	LESSON LEARNED
1	Imported train dataset and test dataset. checked Unique value count per feature and plotted graphs for train. Handled missing values of inventory ratio by filling it with zero. train_test_split(X,Y,test_size=0.2 , random_state=42). Created a basic machine learning pipeline with the preprocessor and regressor(LinearRegression(). Dropped the categorical variables and used only numerical features for training and target variable as sales and dropped all the null values in sales. Then predicted on test dataset	0.68	0.62	11	1515.18	Importance of data preprocessing: Handling missing values, encoding categorical variables, and scaling numerical features are essential steps in preparing data for machine learning models.
2	Handled missing values of inventory ratio using simple imputer mean. train_test_split(X,Y,test_size=0.2 , random_state=42). Created a basic machine learning pipeline with the preprocessor and regressor(GradientBoostingRegr essor(). Dropped the categorical variables and used only numerical features for training and target variable as sales and dropped all the null values in sales. Then predicted on test dataset	0.67	0.61	12	1589.95	Focused on numerical features, dropping categorical variables, and handling null values in the target 'Sales'. These lessons guided a more efficient and effective approach, improving overall model performance and workflow.
3	Same submission(linear Regression())	0.68	0.62	11	1515.18	
4	Developed a preprocessor by importing columnTransformer. Where, Numerical transformer: Applies mean imputation and scaling and Categorical transformer: Applies one-hot encoding to categorical columns. Created a machine learning pipeline with the preprocessor and GBRegressor(n_estimators=100,Learning_rate =0.1, max_depth=3,	0.77	0.52	9	1059.54	Optimizing model performance through parameter tuning underscores the importance of parameter selection in achieving optimal results.

	random_state=42). Finally predicted on testdata.					
5	checked Unique value count per feature and plotted graphs for train. Applied mean imputation and scaling and Categorical transformer: one-hot encoding to categorical columns. Created a machine learning pipeline with the preprocessor and a regressor(LinearRegression(). Finally predicted on testdata.	0.78	0.74	6	817.65	By examining unique value counts and visualizing data distributions, I gained insights into the dataset's characteristics. Applying preprocessing techniques such as mean imputation and one-hot encoding prepared the data for model training, with the Column Transformer
6	Here,I have imported an extra data set which is Economic indicators, then to merge with train csv and test csv I have found a common column i.e., Quarter. But in economic indicators there is a month column, so to merge this column with quarter column of train and test csv I have used economic_data['Month'] = economic_data['Month'].apply(lambda x:'Q'+str((x+2)//3)).[test_data = pd.merge(test_data, economic_data, how='left', on='Month'] and implemented gradient boosting regressor	0.65	0.60	6	1631.97	Merging the datasets with similar column and finding out the parameter to merge by converting month to Quarter and exploring the data to define the result
7	With same pipeline and preprocessing steps without econimc indicators and included standard scalar() for the numerical features and used support vector Regressor(SVR)model. With hyperparameter tuningand predicted the model on test data.	0.66	0.61	6	1629.89	I understood the significance of standard scaling for numerical features, tailoring preprocessing steps to specific datasets and models. Hyperparameter tuning demonstrated the impact of parameter optimization on model performance, emphasizing the iterative nature of model refinement.
8	Using the same pipeline,but there Generated synthetic data using make_regression function from scikit-learn.X, y =	0.67	0.58	6	1644.93	Gained insights by utilizing the make_regression function to create

	make_regression(n_samples=472, n_features=9, noise=0.1, random_state=42). Performed Bayesian Ridge Regression					synthetic data with specific characteristics.Integrating Bayesian Ridge regression into the pipeline introduced me to probabilistic models and their advantages in handling uncertainty..
9	Used The economic indicators with same pipeline but included standard scalar, preprocessor and with parameters of GradientBoostingRegressor(n_estimators=100, learning_rate=0.3, max_depth=6, random_state=42)	0.76	0.70	6	967.87	I gained insights into enriching datasets with external factors. Utilizing the same pipeline but including standard scaling emphasized the significance of data normalization for improving model convergence and performance.
10	started by analyzing the dataset, calculating a correlation matrix, and visualizing it with a heatmap to understand feature importance. Following this, inspected the distribution of numerical predictors by plotting scatter plots against the target variable 'sales' to check for standard normal distribution. Then, split the data into 'missing_df', containing rows with NaN values in 'InventoryRatio', and 'training_df', containing rows without missing values. After specifying predictor columns, trained a Linear Regression model on 'training_df' to predict missing 'InventoryRatio' values, inserting these predictions back into 'missing_df'. Next, applied the same preprocessing steps to the test dataset for predicting 'InventoryRatio'. Finally, utilized a pipeline and preprocessing techniques with Linear Regression for modeling	0.78	0.76	5	796.26	Calculating the correlation matrix and creating a heatmap allowed for understanding feature importance, aiding in feature selection. Plotting scatter plots for numerical predictors against the target variable helped assess the distribution of data and potential relationships. Creating separate data frames for missing and non-missing values of a specific feature demonstrated the importance of handling missing data effectively.
12	Same submission file of 9	0.76	0.70	5	967.87	

13	Performed same steps as submission 10 and used RandomeforestRegressor() to predict on the test data.	0.79	0.76	5	784.26	Tried a different model to know how the regressor worked on the data.
14	Performed same steps as submission 13 and used the same pipeline with preprocessor and implemented modeling with gradientBoostingRegressor() and finally predicted on test_df.	0.74	0.70	8	991.93	Learnt that trying different approaches is mandatory to know how well model works on the dataset.
15	The RandomForestRegressor() model is used as the model I have considered including economic indicators.Firstly I have trained on X_train,Y_train then based on training scores and testing scores fitted the model on X,Y without train test split.	0.99	0.93	8	823.01	Fitting the model on the entire dataset without a train-test split provided insights into the model's overall performance and potential overfitting. This approach emphasizes the importance of evaluating model performance on unseen data and considering the trade-offs between training and testing performance.
16	The RandomForestRegressor is used with the best parameters ('regressor', RandomForestRegressor(max_depth=3, min_samples_split=5, n_estimators=100))found from the grid search.StandardScaler is set for the numerical predictors.	0.98	0.91	8	833.53	Setting StandardScaler for numerical predictors ensures uniform scaling, importance of hyperparameter tuning and preprocessing techniques in enhancing model performance.
17	Introduced StandardScalar() to Standardize numerical features.created a pipeline with the preprocessor and LinearRegression() and fitted directly on X = Features, Y = Target variable and applied the same preprocessor to Test data	0.74	0.72	8	800.38	Fitting the pipeline directly on the features and target variable underscored the simplicity and efficiency of this approach. Applying the same preprocessor to the test data and making predictions further emphasized the importance of consistency in preprocessing between training and testing datasets.
18	Used Ridge regression() with alpha = 0.1 and performed	0.74	0.68	8	800.34	The use of Ridge regression in this

	training initially on X_train and Y_train without considering the economic indicators. And after finding out the R^2 fitted the model on X,Y . In then end predicted on Test dataset.					scenario likely stems from its ability to handle multicollinearity in the dataset and mitigate overfitting. Regularization helps to reduce the impact of multicollinearity, where predictor variables are highly correlated, by distributing the coefficients more evenly across correlated variables.
19	Considering the same RidgeRegression() performed training initially on X_train and Y_train with considering the economic indicators. And after finding out the R^2 fitted the model on X,Y . In then end predicted on Test dataset.	0.69	0.62	8	1016.36	Aggregating Economic Indicators was not useful so far by performing with different model.
20	It begins by loading and preprocessing the training and test datasets, including dropping unnecessary columns, handling missing values, and encoding categorical features. The data is then split into features and target variables, and a preprocessing pipeline is defined to handle numerical scaling and categorical encoding. The pipeline is applied to both the training and test data. A Random Forest regressor model is then trained on the preprocessed training data.	0.94	0.90	7	755.69	Learned that after performing feature distribution and implementing heatmaps, there is no standard normal distribution with all the feature to the target sales. And most the correlations were least positive which significantly says that their high importance of specific feature to the sales.
21	Random Forest regression model is trained using GridSearchCV to find the best hyperparameters {'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 100}. The hyperparameters tuned include .param_grid = { 'n_estimators': [50, 100, 200], 'max_depth': [None, 10, 20], 'min_samples_split': [2, 5, 10], 'min_samples_leaf': [1, 2, 4]	0.99	0.93	7	1014.39	Training and evaluating the model using R^2 scores on both training and testing sets ensure robust performance assessment. But found that there was overfitting with the model as the training was 0.99

	}					
22	Used the before submission pipeline with RandomForestRegressor model and GridsearchCv with parameters- param_grid = { 'n_estimators': [100, 200, 300], 'max_depth': [None, 20,30], 'min_samples_split': [3, 7, 12], 'min_samples_leaf': [1, 3, 5]	0.96	0.85	7	831.89	Learned that after performing preprocessing techniques. It all lies in hyperparameter tuning to select the perfect parameters to the model to perfectly fit.
23	Implemented Ridge regression for sales prediction on a dataset, tuning the regularization parameter (alpha) via GridSearchCV.GridSearchCV parameter (alpha) over a range of predefined values ([0.001, 0.01, 0.1, 1, 10, 100]). The best performing Ridge regression model is determined based on cross-validation.	0.74	0.68	7	774.86	By systematically exploring different alpha values, the best-performing model was identified through cross-validation. By its ability to handle multicollinearity and overfitting.
24	Implemented XGBRegressor with specified Parameters followed by same preprocessing techniques. XGBRegressor(n_estimators=113, learning_rate=0.1, max_depth=5,alpha=0.01,random_state=42).	0.71	0.63	7	984.98	Maintaining consistency in preprocessing techniques ensures that both the training and test datasets are prepared in the same way, reducing the risk of introducing bias or errors during modeling.
25	Same as above submission but here I haven't use the economic indicators for the training. And with parameters XGBRegressor(n_estimators=98, learning_rate=0.001, max_depth=5,alpha=0.1,random_state=42)	0.73	0.65	7	954.96	Removing the Economic Indicators has achieved me a high metric score which seems to be not working well compared to other models.
26	Implemented RandomforestRegressor Model with the same preprocessing techniques and used the economic indicators. but I fitted the model on full training dataset without train test split.	0.99	0.93	7	866.21	Previously I used to fit the model on splitted training data. But I learned that fitting the model on the whole training data is feasible as we were given separate datasets for train and test.

27	Implemented RandomForestRegressor with GridsearchCV using parameters param_grid = { 'n_estimators': [50, 100, 200], 'max_depth': [None, 10, 20],'min_samples_split': [2, 5, 10],'min_samples_leaf': [1, 2, 4] }.GridSearchCV(RandomForest Regressor(random_state=42), param_grid, cv=5, verbose=2, n_jobs=-1).found the Best hyperparameters: {'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 100}. The Fitted the model on whole training dataset.	0.99	0.93	7	863.17	I gained insights into systematically exploring different configurations for the model. fitting the model on the entire training dataset ensures that it learns from the maximum amount of available data, potentially improving its predictive capability..
28	Trained the MLPRegressor neural network model with two hidden layers containing MLPRegressor(hidden_layer_sizes=(150,100,50),max_iter = 800,activation = 'relu',solver = 'adam')neurons, respectively. The model is trained with a maximum 800 of iterations using the Adam solver and ReLU activation function.	0.81	0.72	6	722.62	Training the MLPRegressor with hidden layers containing 150, 100, and 50 neurons, respectively, provides flexibility in capturing complex relationships in the data. Utilizing the Adam solver and ReLU activation function enhances the model's ability to converge efficiently and capture nonlinear patterns.
29	Performed with different parameters MLPRegressor(hidden_layer_sizes=(165,135,75),max_iter =500,activation = 'relu',solver = 'adam')	0.75	0.71	6	817.62	By changing hidden layer sizes and reducing the iterations the Mae score increase which tells us
30	Same as before but with parameters MLPRegressor(hidden_layer_sizes=(150,100,50),max_iter =1000,activation = 'relu',solver = 'adam')	0.82	0.75	5	698.62	Leardned that the hidden layer sizes of 150,100,50 and changing the iterations to 1000 helped to improve the performance and achieved a good score relatively.
31	Intitialized MLPRegressor with early stopping. model =	0.22	0.14	5	1673.95	Learned about Enabling early stopping=True which allows the

	MLPRegressor(activation='relu', solver='adam',alpha=0.001, max_iter=1000, random_state=42, validation_fraction=0.15, # Percentage of training data to use for validation early_stopping=True # Enable early stopping)					training process to halt when performance on the validation set stops improving, preventing overfitting and improving efficiency. Additionally, setting alpha=0.001 controls the L2 regularization term, helping prevent overfitting. But it didn't perform well.
32	Implemented MLPRegressor with parameters MLPRegressor(hidden_layer_sizes=(250,150,100),max_iter = 1000,activation = 'relu',solver = 'adam')	0.78	0.71	5	754.34	Changed the hidden layer sizes to achieve better performance.
33	Same as before submission but changed Max_iteration = 1500	0.81	0.76	5	731.24	Learned that trying different parameters and tuning to get the better performance.
34	Trained CatBoostRegressor model with the following parameters: loss function set to RMSE, 3000 iterations, and a learning rate of 0.07. The model is trained and validated using train-test split. Finally, it evaluates the model's performance using RMSE and R2 score metrics	0.74	0.66	5	950.76	Got to know that CatBoostRegressor has its ability to handle categorical features seamlessly without the need for one-hot encoding, thus simplifying the preprocessing step. Its efficient handling of categorical data, along with built-in support for handling missing values, made it a suitable choice for datasets with diverse feature types.
35	Initialized MLPRegressor after performing Feature importance using decision Tree and finding out feature scores for all the features against the target variable. Considering the features and based on their scores used only features that has good feature importance against sales and Performed the MLPRegressor with parameters MLPRegressor(hidden_layer_sizes=(150,100,50),max_iter =	713.89	822.57	5	777.28	By first performing feature importance analysis using decision trees, I gained insights into the relative importance of each feature with respect to the target variable, sales. Selecting only the features with high importance scores ensures that the model focuses on the most

	500,activation = 'relu',solver = 'adam')					relevant information for prediction.
36	Refreshed The similar Code of previous submission with Max_iterations =800	698.12	800.37	5	741.26	Increasing the iterations reduced the Error.
37	Refreshed The similar Code of previous submission with Max_iterations =1000	678.82	798.16	5	699.51	Changing the iterations to 1000 which improved the model performance than with 800
38	Refreshed The similar Code of previous submission with Max_iterations =1500	710.35	820.54	5	776.58	By increasing the iterations to 1500 the model overfitted.
39	This code utilizes to perform regression analysis using a Multi-layer Perceptron (MLP) regressor, instantiated with specific hyperparameters: consists of two hidden layers with 115 and 55 neurons respectively, runs for a maximum of 1500 iterations, employs the ReLU activation function, and utilizes the Adam optimizer. The dataset is split into training and testing subsets, with a 15% allocation for testing.	637.76	789.96	4	650.11	Learned that reducing the architecture to two hidden layers with 115 and 55 neurons and running for a maximum of 1500 iterations, utilizing the ReLU activation function, and employing the Adam optimizer and allocating only 15% for testing increased the model performance.
40	Irrelevant columns are removed from both datasets. Categorical features are selected for one-hot encoding. Missing values in the training and test datasets are imputed using KNNImputer = 2 , The imputed datasets are converted into DataFrames. A Sequential model with multiple dense layers is created using TensorFlow's Keras API. The model comprises an input layer and five hidden layers with 103, 64, 32, 16, and 8 neurons, respectively, activated by ReLU functions. The output layer is a single neuron for regression. The model is compiled with a custom root mean squared error (RMSE) loss function and the Adam optimizer with a learning rate of 0.01. Training is performed on	671.47	575.33	4	694.59	I learned that Reducing the Learning rate and increasing the patience led to increase the validation loss and observed model overfitting.

	the training dataset using 10% of the data for validation					
41	Initialized The same DL model with an input layer and five hidden layers with 103, 64, 32, 16, and 8 neurons and ReLU function. The model is compiled with a custom root mean squared error (RMSE) loss function and the Adam optimizer with a learning rate of 0.0001 and patience = 2	639.32	564.18	4	647.50	It becomes evident that adjusting parameters is essential for optimizing neural network models. Beyond preprocessing, parameter tuning significantly influences how well the model learns and generalizes from the data. Parameters such as the network architecture, neuron counts, activation functions, learning rate, batch size, and epochs play crucial roles.
42	Executed the same code with learning rate = 0.05 and patience = 10 with similar other parameters.	688.95	613.19	4	702.17	I learned that Increasing the learning rate to 0.05 from 0.0001 with patience to 10 has significantly increased the validation loss, which refers to model overfitting.
43	Initialized The same DL model and preprocessing Techniques. But here I have changed the hidden layers and used sequential model with hidden layers of 98,60,20 neurons respectively. Additionally applied regularization – l2. With learning rate = 0.0001 and patience = 2	641.27	586.31	4	666.60	Despite using the same preprocessing techniques, modifying the hidden layers in the DL model allows for exploration of different network configurations. Introducing kernel regularization with l2 regularization further enhances the model's ability to generalize by mitigating overfitting. Additionally, setting a specific learning rate and implementing early stopping with patience of 2 ensures efficient training and prevented overfitting.

44	Executed the same DL model with same hidden layers as of submission 43 but with learning rate = 0.001 and patience = 5	630.58	579.29	4	659.55	Increasing the learning rate to 0.001 and patience to 5 resulted in a slight improvement in the metric score, indicating better performance compared to the previous configuration.
45	The neural network model architecture consists of an input layer with 98 features, followed by three hidden layers with 98, 60, and 20 neurons, respectively. The model is compiled with an Adam optimizer and trained for a maximum of 100 epochs with a batch size of 2. Including learning rate = 0.01 and patience = 10. The root mean squared error loss function is used for model evaluation during training. Achieving the best metric score in the Kaggle challenge.	616.35	553.74	2	630.36	Changing Dense layers and reducing the learning rate to 0.01 and patience to 10 and increasing the epochs to 100 the model performed pretty well and observed the model was not overfitting and validation loss was stable, when compared to others. Achieving the best metric score in the Kaggle challenge underscores the effectiveness of this model architecture and parameter configuration for the given task.

RECOMMENDATIONS

1. Feature Engineering: Consider exploring additional features or engineering existing ones to better capture the underlying patterns and dynamics in the data. This could involve creating new variables, transforming existing ones, or incorporating external data sources to enrich the dataset.
2. Ensemble Methods: Explore ensemble methods such as Stacking or Boosting, which combine multiple models to improve predictive accuracy. By leveraging the strengths of different algorithms, ensemble methods can often outperform individual models and provide more robust predictions.
3. Given the sequential nature of sales data, Time Series Analysis techniques to account for temporal dependencies and seasonality. Models such as ARIMA, SARIMA, or Prophet could be valuable in capturing time-related patterns and improving forecast accuracy.
4. Explore the potential of deep learning models, capturing complex nonlinear relationships and temporal dependencies, making them suitable for time-series prediction tasks.

Enhancing Sales Forecasting with Generative AI:

1) Data Preprocessing Guidance: To integrate the economic indicators into both the train and test datasets, the month column was utilized to convert the data into quarters, aligning with the corresponding column format in the train and test datasets. This process facilitated the merging of economic indicators seamlessly, ensuring compatibility with the existing datasets.

"Generate Python code to merge economic indicators with both the training and testing datasets based on the quarter information. Ensure alignment between the month column in the economic data and the quarter column in the datasets. Utilize pandas library functions for merging and data manipulation."

2) Hyperparameter Tuning Strategies:To optimize model performance, the AI emphasized the importance of hyperparameter tuning. Specific parameters such as max depth, learning rate, and hidden_layer_sizes were highlighted as critical for enhancing predictive accuracy. The prompts guided the refinement of model configurations through systematic parameter adjustments.

"Optimize model performance by applying hyperparameter tuning using GridSearchCV. Experiment with parameters like max depth, n_estimators, and min_samples_split for RandomForestRegressor."

By incorporating these prompts and suggestions from generative AI, It helped me in coding part wherever I found it difficult to enhance my knowledge.