Sales Forecasting Kaggle Challenge

Ruhi Sania
Net ID: LT7551

Business Problem:

This is a B2B sales prediction problem. The dataset is from a steel manufacturer that has other businesses in the Auto, Metal Fabrication and Infrastructure as customers. The goal is to predict the quarterly sales to each of the 75 customers. In addition to company specific data, general economic indicators are also included that can be used for the purposes of prediction.

Data:

- **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

Approach/Submission:

Submission No	Description	Train MAE	Validation MAE	Test MAE (Public)	Rank at Submission	Lessons
1	Basic Notebook with reading data,replaced missing data with median and run a regression model	1284.6	1866.7	1634.41	4	
2	Created dummies for categorical data and run a regression model	730.1	974.5	872.94	6	The creation and inclusion of dummy variables for categorical data significantly improved the model's performance across all metrics. This indicates that the categorical variables hold important information relevant to the target variable.
3	Removed outliers by identifying them with boxplots	716.4	973.4	878.92	9	Removing outliers gave a similar result in model's performance
4-6	Used backward elimination and rerun the model using best variables	725.5	988.1	884.15	9	
7-8	Applied a Decision Tree Model	143.6	1806.6	1701.84	12	Model performance improved for training data but decreased for validation and test data
9	Used Random forest to build a model	105.8	1089.4	1352.73	15	Compared to Decision Tree the model performance is better for Random Forest.
10	Added economic indicators to	820.4	850.7	143209.33	16	Predicting inventoryratio values

	training data and replaced missing values using linear regression predictions.Run the linear regression model again for test data predictions					gave negative values which led to wrong predtions
11	Added economic indicators (averaged to quarters) to training data and replaced missing values with median.Run the linear regression model again for test data predictions.	760.42	980.34	2281.65	18	Adding economic indicators was overfitting the model.
12	Applied Xgboost Regressor to train data with basic preprocessing	789.59	957.26	836.94	18	
13-14	Removed few features for the XGB regressor model	250	911	1100	18	Bond rating and Srock rating might have good correlation with sales as removing them did not help
15-16	Applied Ridge regression model and replaced missing values with mean	815.4	884.6	819.14	18	Prevented overfitting using Regularization.Model performance increased
17	Applied Gradient boost regressor but deleted the rows with missing values	693	1028	1720.19	18	Deleting rows with missing values did not improve model performance
18	Normalized numerical features and run Ridge regression model with regularization	803.3	873.5	811.1	16	Scaling or normalizing features is important to preserve relative differences in values while ensuring they are positive

19	Removed the rows where Sales were missing as they were the same rows used for test data and run linear regression model	708.1	982.6	758.35	9	Identifying missing values using domain knowledge is crucial
20	Run Ridge regression model after the changes in 19	708.4	927.4	751.37	7	
21	Performed k fold validation with ridge regressor(5 folds)	1073.5	970.2	878.38	11	Helps with the overfitting problem
22	Performed bootstrapping with regression model	895.4	974.66	837.26	11	When the data is small and has high variability, bootstapping works better than k-fold validation
23	Used MinMaxScaler() class for Standardization and run linear regression model	735.50	941.14	864.52	11	Linearly scaling each feature indepently between 0 and 1 range did not significantly improve the score for the linear regression model
24	Performed the Scaling with the merged data and run Ridge regression model with 80-20 train-test split	927.43	708.36	785.34	13	The model fit the new data better when the data was merged with economic indicators and scaled
25	Tried to use Lasso Regression with changing number of iterations and k fold cross validation	1581.5	704.35	794.33	13	Using Lasso regressor did not improve the model's performance.
26	Used Min max scaler and run 24 again	927.43	708.36	850.69	14	

27&28	Scaled the unmerged data and run Random Forest Regressor	335.15	1001.23	851.73	14	
29	Performed grid search to determine number of estimators(320) for RandomForest model	334.88	1016.54	1855.79	14	The model performance decreased by increasing the number of estimators for the Random forest Model
30&31	Used PowerTransformer class for scaling and run Ridge regression model	682.35	896.94	713.96	8	Power Transformer reduced the impact of outliers and stabilized the variance across features improving the model performance significantly
32	Run 30 again with 100 bootstrap samples	592.56	988.61	805.63	8	
33	Applied hyperparameter tuning and run GradientBoosting Regressor on 30 with {'learning_rate': 0.3, 'n_estimators': 140}	296.93	874.66	836.54	8	Even after making scaling better GradientBoost model was not a good fit
34	Replaced missing values of Inventory ratio with mean of respective company , merging with economic indicators and run the model again	672.60	947.47	781.29	8	
35&36	Applied Lasso regression to 34	681.87	909.69	870.64	8	
37	Removed few columns after creating dummies	717.20	906.52	782.04	9	Removing columns with low or zero coefficents

	by analyzing their coefficients(weightage) to make the model accurate					can improve efficiency of the model
38&39	To the train data and test data ,combined rare categories in categorical variables that have less than 50 values and run ridge model	682.34	896.79	822.48	9	Combining rare categories can lead to a simpler, more robust, and better-performing model by reducing overfitting, simplifying the data, and improving generalization.
40	Only did combining rare categories to the train data and not the test data for categorical variables that have less than 30 values in a category and run the ridge model	706.07	871.88	728.74	9	For the train data, the minimun count set as 30 works gives more accurate model
41	Only did combining rare categories to the train data and not the test data for categorical variables that have less than 50 values in a category and run the ridge model	789.43	939.68	823.54	10	

Individual Contributions Summary:

Data Preprocessing:

- Identified and addressed missing values in Inventory Ratio using different techniques (median imputation, linear regression prediction, mean imputation) and found that mean imputation improved model's performance.Removed the rows for missing values in Sales Columns after identifying them as Duplicate rows.
- Handled outliers through visual inspection (boxplots) and using regularization as removing the outliers was not improving the model performance.
- Experimented with different data scaling methods such as MinMaxScaler, StandardScaler, and PowerTransformer to normalize numerical features and stabilize model performance.
- Feature engineering: Created dummy variables for categorical features, analyzed feature coefficients to identify and remove potentially irrelevant features.Combined rare categories within categorical variables to improve model performance and reduce overfitting.

Data Visualization:

Following were observed through bar charts, scatter plots and heatmap using the data:

- High frequency of sales distributions observed for Inventory Ratio values below 10.
- There was no significant change in Sales depending on the Quarter, Bond rating, Stock rating, Industry and Region categories.
- The total Sales for Companies CMP05,CMP18,CMP20,CMP27,CMP28,CMP29,CMP30,CMP33,CMP61,CMP68,CMP70 was higher than 40000
- There wasn no strong correlation found between other features and Sales.

Model Selection and Training:

- Evaluated various regression models (Linear Regression, Ridge Regression, Lasso Regression, Gradient Boosting Regressor, Random Forest Regressor) through experimentation and hyperparameter tuning.
- Performed k-fold cross validation and bootstrapping to assess model performance and address overfitting.
- Identified PowerTransformer scaling, Combining rare categories and Ridge Regression as the most effective combination for this dataset, achieving a Test MAE of 728.74

Analysis and Insights:

- Documented the exploration process, including model performance metrics (Train MAE, Validation MAE, Test MAE) for each approach.
- Identified that company-specific data along with economic indicators were crucial for accurate sales prediction.
- Highlighted the importance of feature scaling and addressing missing values for improved model performance.

AI Contributions Summary:

ChatGPT has assisted in the following aspects of the project:

Data Preprocessing: Provided coding guidance on handling missing values in the 'InventoryRatio' column using Simple Imputer.I have included the dataframe name and column name for the code and asked for code to replace missing values in InventoryRatio column with mean in the prompt.Following is an example prompt I've used for replacing missing values based on mean of the particular company.

"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 Inventory ratio by taking the mean of each company's Q1 to Q9 values of inventory ratio.If the inventory ratio is not available for any company then leave it blank".For scaling the data ChatGPT provided me with better techniques like Power transforms, such as the Yeo-Johnson or Box-Cox transforms when prompted for better scaling methods other than Standard Scaling.

Model Selection and Training: Offered guidance on evaluating various regression models (Linear Regression, Ridge Regression, Lasso Regression, Gradient Boosting Regressor, Random Forest Regressor) through experimentation and hyperparameter tuning. Prompts used were Perform "Model name" using sales _df and how can I perform hyperparameter tuning for random forest.

ChatGPT also recommended and gave modified codes to performing k-fold cross-validation and bootstrapping to assess model performance and address overfitting when prompted "Modify this code to use bootstrapping". Overall, ChatGPT contributed to the project by providing valuable insights and recommendations at various stages, facilitating the exploration and optimization of predictive models for Sales Forecasting.

Step 0: To start ...

In [1]:
```
# Turn on multi-threading on your computer for faster calculation
%env OMP_NUM_THREADS = 4
```

```
env: OMP_NUM_THREADS=4
```

Steps 1 and 2: Install and load the necessary packages and libraries

In [2]:
```
from pathlib import Path # to interact with file system.

import numpy as np # for working with arrays.
import pandas as pd # for working with data frames (tables).
from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split # for data partition.
from sklearn.metrics import r2_score # to identify r_squared for regression mo
from sklearn.linear_model import LinearRegression # for linear regression mode
from pandas.plotting import scatter_matrix, parallel_coordinates
from sklearn.impute import SimpleImputer
import pandas as pd
from sklearn.preprocessing import MinMaxScaler, StandardScaler
import dmba
from dmba import regressionSummary, exhaustive_search
from dmba import backward_elimination, forward_selection, stepwise_selection
from dmba import adjusted_r2_score, AIC_score, BIC_score
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score


%matplotlib inline
import matplotlib.pylab as plt #%matplotlib inline renders the figure in a note
```

```
no display found. Using non-interactive Agg backend
```

In [3]:
```
pip install xgboost
```

```
Requirement already satisfied: xgboost in /Users/ruhisania/anaconda3/lib/pytho
n3.10/site-packages (2.0.3)
Requirement already satisfied: scipy in /Users/ruhisania/anaconda3/lib/python
3.10/site-packages (from xgboost) (1.10.0)
Requirement already satisfied: numpy in /Users/ruhisania/anaconda3/lib/python
3.10/site-packages (from xgboost) (1.23.5)
Note: you may need to restart the kernel to use updated packages.
```

Step 3: Load the data

In [4]:
```
sales_df = pd.read_csv('train.csv')
```

In [5]:
```
sales_df.head()
```

Out[5]:

	ID	Company	Quarter	QuickRatio	InventoryRatio	RevenueGrowth	MarketshareChange	Bo rati
0	0	CMP01	Q1	2.02	7.71	0.05	-0.04	C(
1	1	CMP01	Q2	2.01	4.10	0.03	0.00	C(
2	2	CMP01	Q3	2.02	6.79	0.06	-0.02	C(
3	3	CMP01	Q4	1.98	3.97	0.01	0.02	C(
4	4	CMP01	Q5	1.96	7.41	-0.07	0.02	C(

In [6]:
```
sales_df.describe()
```

Out[6]:

	ID	QuickRatio	InventoryRatio	RevenueGrowth	MarketshareChange	Sa
count	675.000000	675.000000	523.000000	675.000000	675.000000	525.000(
mean	394.555556	1.603867	4.265124	-0.009733	-0.002904	3556.708
std	204.960069	0.595615	3.108644	0.067390	0.017622	2028.059:
min	0.000000	0.500000	1.260000	-0.200000	-0.050000	864.000(
25%	216.500000	0.990000	2.630000	-0.070000	-0.015000	1992.000(
50%	433.000000	1.730000	3.420000	0.000000	0.000000	3007.000(
75%	579.000000	2.155000	4.725000	0.050000	0.010000	4523.000(
max	674.000000	2.490000	24.840000	0.080000	0.020000	11686.000(

In [7]:
```
sales_df.columns
```

Out[7]:
```
Index(['ID', 'Company', 'Quarter', 'QuickRatio', 'InventoryRatio',
       'RevenueGrowth', 'MarketshareChange', 'Bond rating', 'Stock rating',
       'Region', 'Industry', 'Sales'],
      dtype='object')
```

In [8]:
```
sales_df.Company = sales_df.Company.astype('category')
sales_df.Quarter = sales_df.Quarter.astype('category')
```

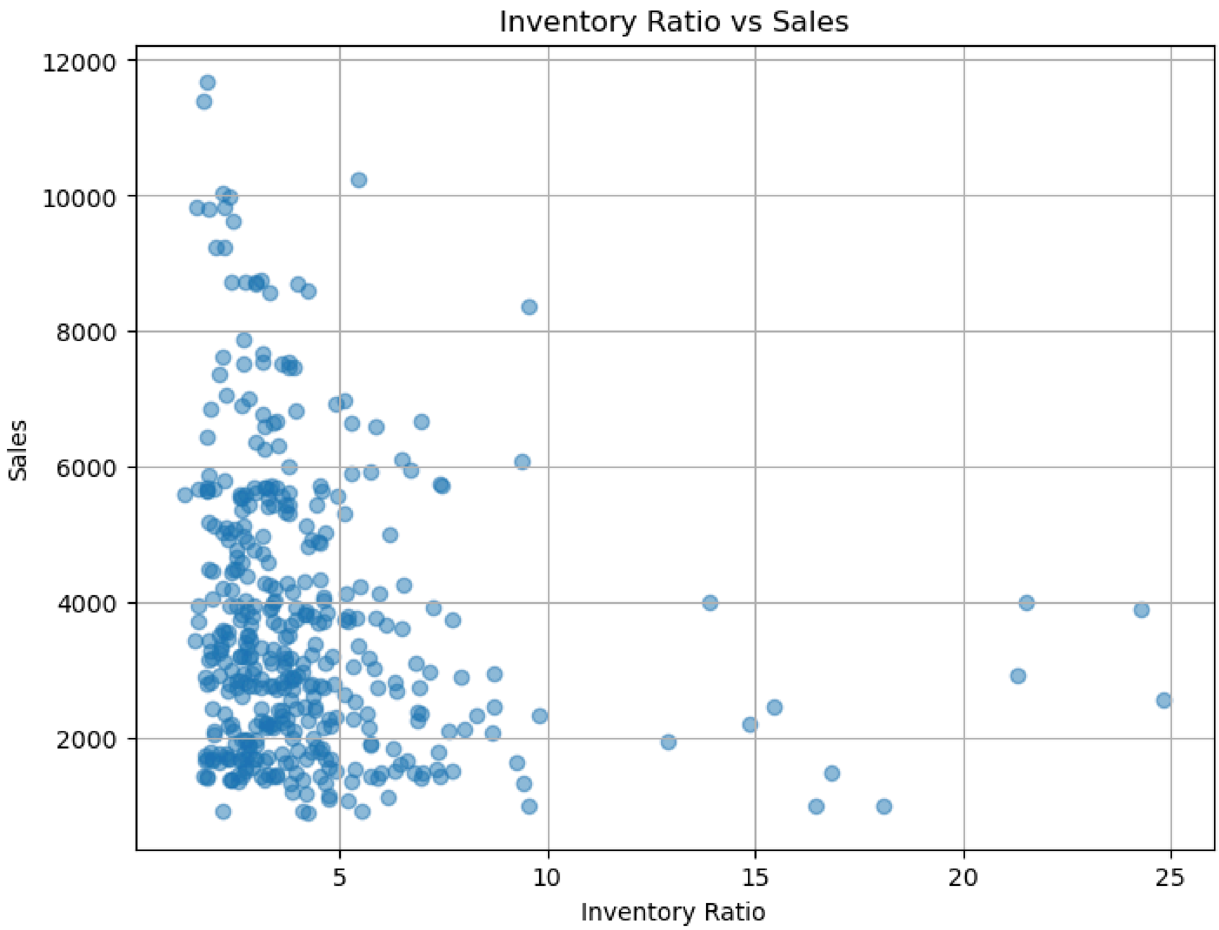
In [9]:
```
sales_df = sales_df.rename(columns={'Bond rating': 'Bond_rating'})
```

In [10]:
```
sales_df = sales_df.rename(columns={'Stock rating': 'Stock_rating'})
```

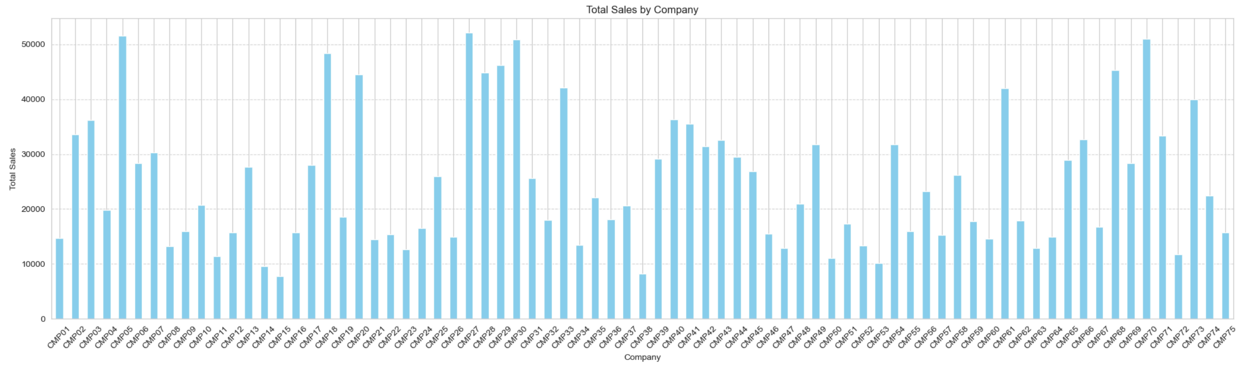
In [11]:
```
sales_df.Bond_rating = sales_df.Bond_rating.astype('category')
sales_df.Stock_rating = sales_df.Stock_rating.astype('category')
sales_df.Industry = sales_df.Industry.astype('category')
sales_df.Region = sales_df.Region.astype('category')
```

In [12]:
```
inventory_ratio = sales_df['InventoryRatio']
sales = sales_df['Sales']

# Plotting the graph
```

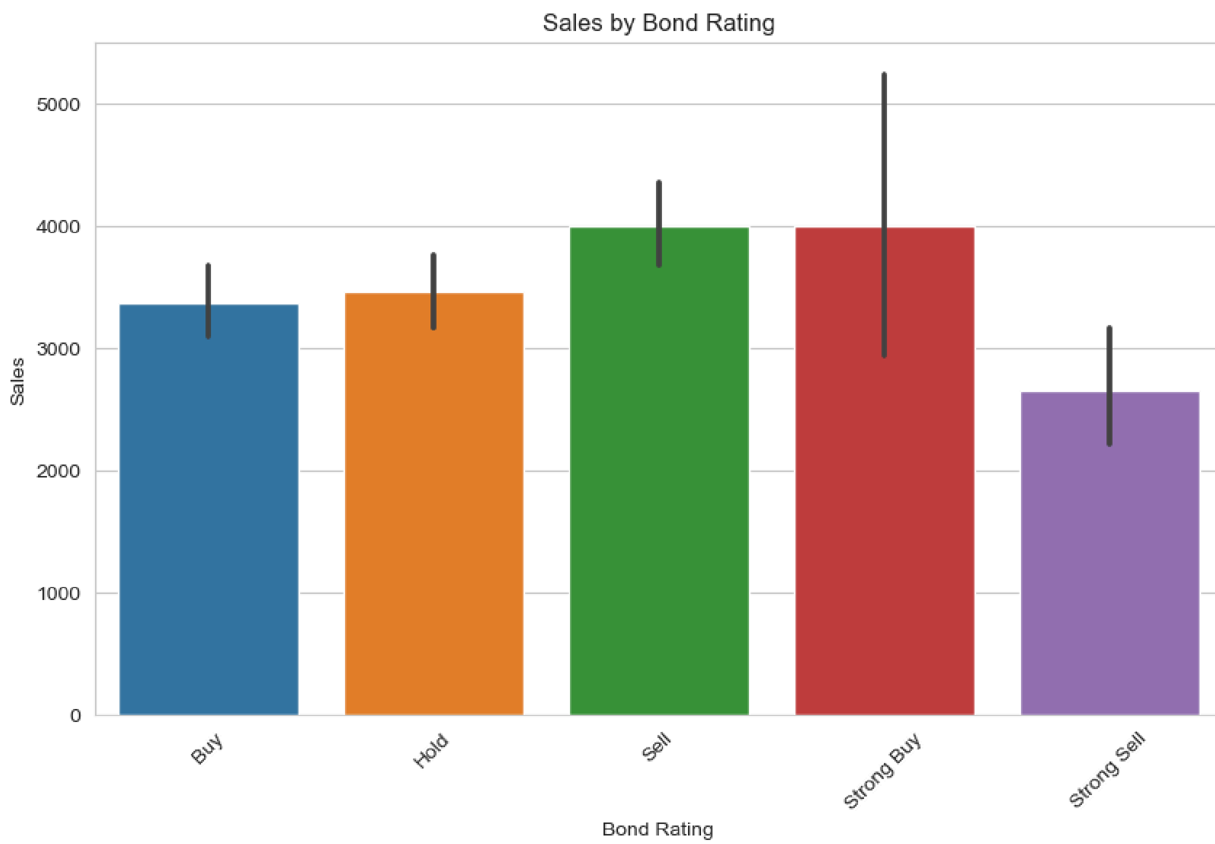
```
plt.figure(figsize=(8, 6))
plt.scatter(inventory_ratio, sales, alpha=0.5)
plt.title('Inventory Ratio vs Sales')
plt.xlabel('Inventory Ratio')
plt.ylabel('Sales')
plt.grid(True)
plt.show()
```



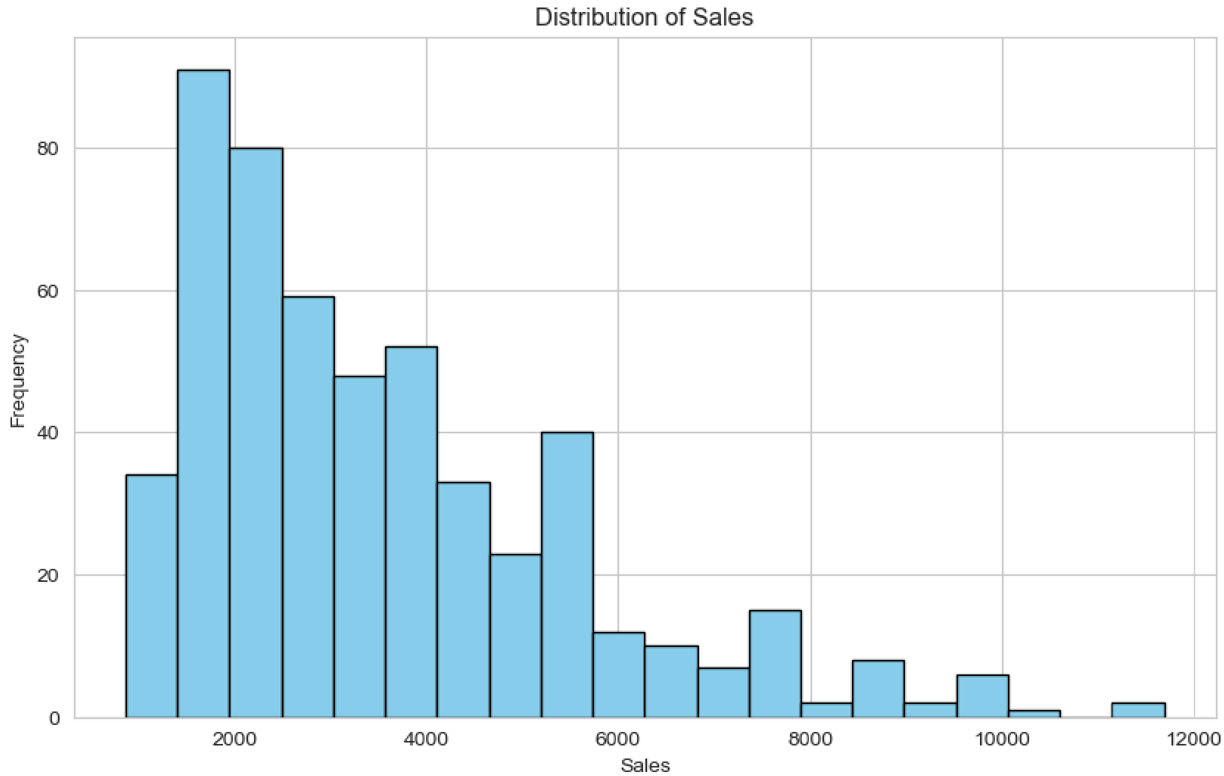
In [14]:
```
import matplotlib.pyplot as plt

# Assuming sales_df is your DataFrame containing the columns 'Company' and 'sa
# Plotting a bar chart for sales by company
plt.figure(figsize=(20, 6))
sales_df.groupby('Company')['Sales'].sum().plot(kind='bar', color='skyblue')
plt.title('Total Sales by Company')
plt.xlabel('Company')
plt.ylabel('Total Sales')
plt.xticks(rotation=45)  # Rotate x-labels for better readability
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```



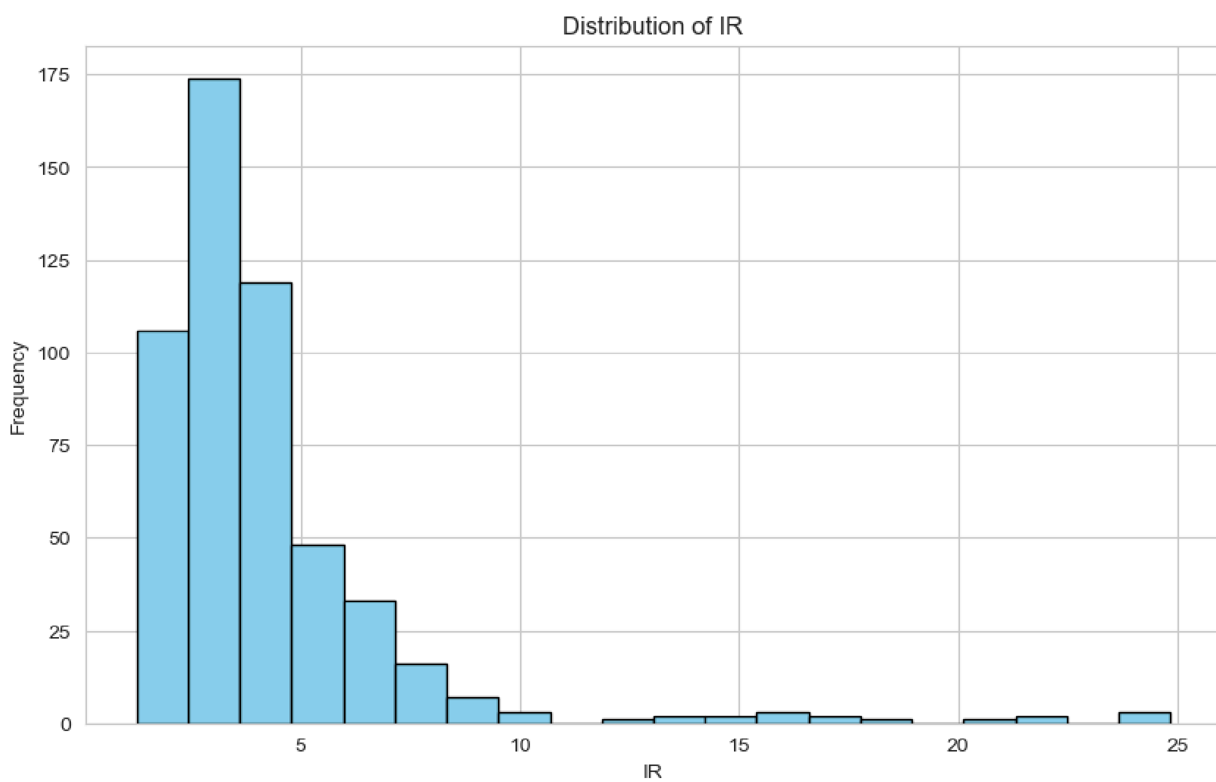
In [15]:
```
plt.figure(figsize=(10, 6))
sns.barplot(x='Stock_rating', y='Sales', data=sales_df)
plt.title('Sales by Bond Rating')
plt.xlabel('Bond Rating')
plt.ylabel('Sales')
plt.xticks(rotation=45)  # Rotate x-axis labels for better readability
plt.show()
```



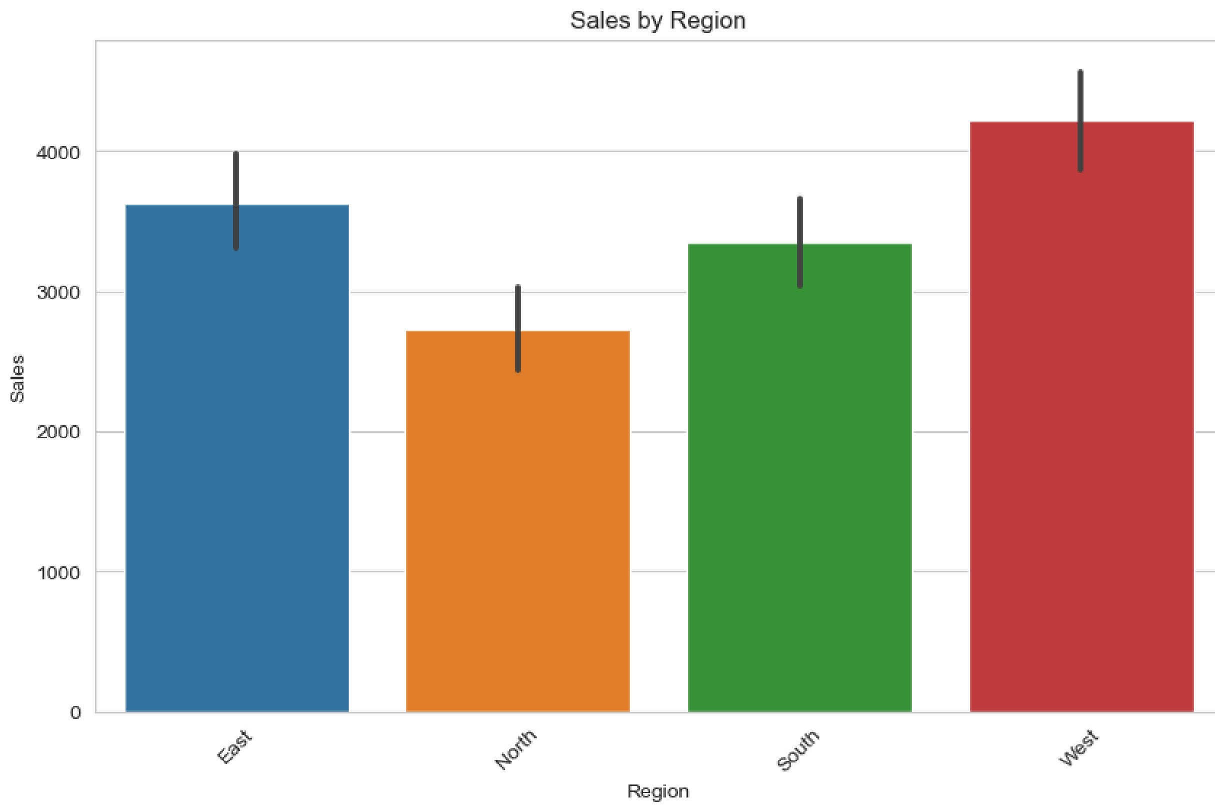
In [16]:
```
plt.figure(figsize=(10, 6))
plt.hist(sales_df['Sales'], bins=20, color='skyblue', edgecolor='black')
plt.title('Distribution of Sales')
plt.xlabel('Sales')
plt.ylabel('Frequency')
plt.grid(True)
plt.show()
```



In [17]:
```
plt.figure(figsize=(10, 6))
plt.hist(sales_df['InventoryRatio'], bins=20, color='skyblue', edgecolor='blac
plt.title('Distribution of IR')
plt.xlabel('IR')
plt.ylabel('Frequency')
plt.grid(True)
plt.show()
```



In [18]:
```
plt.figure(figsize=(10, 6))
sns.barplot(x='Region', y='Sales', data=sales_df)
plt.title('Sales by Region')
plt.xlabel('Region')
plt.ylabel('Sales')
plt.xticks(rotation=45)  # Rotate x-axis labels for better readability
plt.show()
```



In [19]:
```
sales_df.loc[:, 'Sales'] = sales_df.groupby(['ID', 'Quarter'])['Sales'].transf
```

In [20]:
```
sales_df = sales_df.drop_duplicates(subset=['ID', 'Company'])
```

In [21]:
```
sales_df.describe()
```

Out[21]:

	ID	QuickRatio	InventoryRatio	RevenueGrowth	MarketshareChange	Sa
count	559.000000	559.000000	443.000000	559.000000	559.000000	525.000
mean	352.148479	1.617531	4.198736	-0.009678	-0.003005	3556.708
std	199.682322	0.585503	2.921338	0.067685	0.017503	2028.059
min	0.000000	0.500000	1.260000	-0.200000	-0.050000	864.000
25%	178.500000	1.035000	2.630000	-0.065000	-0.020000	1992.000
50%	357.000000	1.750000	3.470000	0.000000	0.000000	3007.000
75%	534.500000	2.130000	4.670000	0.050000	0.010000	4523.000
max	674.000000	2.490000	24.840000	0.080000	0.020000	11686.000

In [22]:
```
test_df= pd.read_csv('test.csv')
```

In [23]:
```
test_df = test_df.rename(columns={'Bond rating': 'Bond_rating'})
```

In [24]:
```
test_df = test_df.rename(columns={'Stock rating': 'Stock_rating'})
```

In [25]:
```
test_df.Stock_rating = test_df.Stock_rating.astype('category')
test_df.Bond_rating = test_df.Bond_rating.astype('category')
test_df.Industry = test_df.Industry.astype('category')
test_df.Company = test_df.Company.astype('category')
test_df.Quarter = test_df.Quarter.astype('category')
test_df.Region = test_df.Region.astype('category')
```

In [26]:
```
sales_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 559 entries, 0 to 674
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   ID                 559 non-null    int64
 1   Company            559 non-null    category
 2   Quarter            559 non-null    category
 3   QuickRatio         559 non-null    float64
 4   InventoryRatio     443 non-null    float64
 5   RevenueGrowth      559 non-null    float64
 6   MarketshareChange  559 non-null    float64
 7   Bond_rating        559 non-null    category
 8   Stock_rating       559 non-null    category
 9   Region             559 non-null    category
 10  Industry           559 non-null    category
 11  Sales              525 non-null    float64
dtypes: category(6), float64(5), int64(1)
memory usage: 37.7 KB
```

In [27]:
```
pd.DataFrame(sales_df).isna().sum()
```

Out[27]:
```
ID                     0
Company                0
Quarter                0
QuickRatio             0
InventoryRatio       116
RevenueGrowth          0
MarketshareChange      0
Bond_rating            0
Stock_rating           0
Region                 0
Industry               0
Sales                 34
dtype: int64
```

In [28]:
```
sales_df.describe()
```

Out[28]:

	ID	QuickRatio	InventoryRatio	RevenueGrowth	MarketshareChange	Sa
count	559.000000	559.000000	443.000000	559.000000	559.000000	525.000
mean	352.148479	1.617531	4.198736	-0.009678	-0.003005	3556.708
std	199.682322	0.585503	2.921338	0.067685	0.017503	2028.059
min	0.000000	0.500000	1.260000	-0.200000	-0.050000	864.000
25%	178.500000	1.035000	2.630000	-0.065000	-0.020000	1992.000
50%	357.000000	1.750000	3.470000	0.000000	0.000000	3007.000
75%	534.500000	2.130000	4.670000	0.050000	0.010000	4523.000
max	674.000000	2.490000	24.840000	0.080000	0.020000	11686.000

In [29]:

```
sales_df.dropna(subset=['Sales'], inplace=True)
```

In [30]:

```
#Impute missing values in IR with mean
```

In [31]:

```
imputer = SimpleImputer(missing_values=np.nan, strategy='mean')

# Fit the imputer to columns with numerical data
imputer.fit(sales_df.select_dtypes(include=['int64', 'float64']))

# Transform and replace missing values in numerical columns wieith the mean
sales_df[sales_df.select_dtypes(include=['int64', 'float64']).columns] = imput
```

In [34]:

```
sales_df.describe()
```

Out[34]:

	ID	QuickRatio	InventoryRatio	RevenueGrowth	MarketshareChange	Sa
count	525.000000	525.000000	525.000000	525.000000	525.000000	525.000
mean	336.000000	1.622019	4.176124	-0.010400	-0.002914	3556.708
std	195.034495	0.581223	2.597026	0.068026	0.017389	2028.059
min	0.000000	0.500000	1.260000	-0.200000	-0.050000	864.000
25%	167.000000	1.050000	2.770000	-0.070000	-0.010000	1992.000
50%	336.000000	1.750000	3.940000	0.000000	0.000000	3007.000
75%	505.000000	2.120000	4.310000	0.050000	0.010000	4523.000
max	672.000000	2.490000	24.840000	0.080000	0.020000	11686.000

In [35]:

```
imputer = SimpleImputer(missing_values=np.nan, strategy='mean')

# Fit the imputer to columns with numerical data
imputer.fit(test_df.select_dtypes(include=['int64', 'float64']))

# Transform and replace missing values in numerical columns with the mean
test_df[test_df.select_dtypes(include=['int64', 'float64']).columns] = imputer
```

In [36]:

```
def CombineRareCategories(data, mincount, columns_to_process):
    for col in columns_to_process:
        if (type(data[col][0]) == str):
```

```
            for index, row in pd.DataFrame(data[col].value_counts()).iterrows(
                if (row[0] < mincount):
                    data[col].replace(index, 'Other_' + col, inplace=True)
                else:
                    None

# Specify the columns you want to process
columns_to_process = ['Bond_rating', 'Stock_rating', 'Region','Industry']  # Ad

# Apply the function to your data with the specified columns and mincount
CombineRareCategories(sales_df, 30, columns_to_process)

# Check the result for the first 10 rows
print(sales_df.head(10))
```

```
     ID Company Quarter  QuickRatio  InventoryRatio  RevenueGrowth  \
0   0.0   CMP01      Q1        2.02            7.71           0.05
1   1.0   CMP01      Q2        2.01            4.10           0.03
2   2.0   CMP01      Q3        2.02            6.79           0.06
3   3.0   CMP01      Q4        1.98            3.97           0.01
4   4.0   CMP01      Q5        1.96            7.41          -0.07
5   5.0   CMP01      Q6        1.96            2.12          -0.19
6   6.0   CMP01      Q7        1.97            3.47           0.05
7   9.0   CMP02      Q1        2.00            5.46          -0.07
8  10.0   CMP02      Q2        2.01            2.65          -0.01
9  11.0   CMP02      Q3        2.01            4.40           0.07

   MarketshareChange        Bond_rating Stock_rating Region  \
0              -0.04                CCC          Buy  South
1               0.00                CCC         Hold  South
2              -0.02                CCC          Buy  South
3               0.02                CCC          Buy  South
4               0.02                CCC          Buy  South
5               0.01                CCC          Buy  South
6              -0.01                CCC          Buy  South
7              -0.02  Other_Bond_rating  Strong Sell   West
8              -0.01                 AA  Strong Sell   West
9               0.01                 AA  Strong Sell   West

            Industry   Sales
0  Metal Fabrication  1517.0
1  Metal Fabrication  2968.0
2  Metal Fabrication  1497.0
3  Metal Fabrication  2929.0
4  Metal Fabrication  1452.0
5  Metal Fabrication  2918.0
6  Metal Fabrication  1460.0
7     Infrastructure  3376.0
8     Infrastructure  6899.0
9     Infrastructure  3393.0
```

In [37]:

```
features_to_normalize = ['MarketshareChange', 'RevenueGrowth','QuickRatio','Inv

from sklearn.preprocessing import PowerTransformer

scaler = PowerTransformer(method='yeo-johnson')
sales_df[features_to_normalize] = scaler.fit_transform(sales_df[features_to_no
scaler = PowerTransformer(method='yeo-johnson')
test_df[features_to_normalize] = scaler.fit_transform(test_df[features_to_norma
```

In [38]:
```
sales_df.columns
```

Out[38]:
```
Index(['ID', 'Company', 'Quarter', 'QuickRatio', 'InventoryRatio',
       'RevenueGrowth', 'MarketshareChange', 'Bond_rating', 'Stock_rating',
       'Region', 'Industry', 'Sales'],
      dtype='object')
```

In [39]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import xgboost as xgb
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
predictors = ['ID','Company','Quarter', 'QuickRatio','InventoryRatio','Revenue
       'Region','Industry']
outcome = 'Sales'

# partition data
X = pd.get_dummies(sales_df[predictors], drop_first=True)
y = sales_df[outcome]
# Create the training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, randon

from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import Ridge
from sklearn.metrics import mean_squared_error

# Instantiate Ridge regression model with regularization parameter alpha
ridge_reg = Ridge(alpha=0.1)

# Fit the model to the training data
ridge_reg.fit(X_train, y_train)

# Make predictions

y_train_pred = ridge_reg.predict(X_train)
mae_train = mean_absolute_error(y_train, y_train_pred)
print("Train MAE: %f" % mae_train)

y_pred = ridge_reg.predict(X_test)
mae = mean_absolute_error(y_test, y_pred)
print("Valid MAE: %f" % mae)
```

```
Train MAE: 706.070545
Valid MAE: 871.884966
```

In [42]:
```
test_df = pd.get_dummies(test_df, prefix_sep='_', drop_first=True)
sales_df = pd.get_dummies(sales_df, prefix_sep='_', drop_first=True)
all_column_names2 = test_df.columns.to_list()
all_column_names = sales_df.columns.to_list()

new_data = test_df[all_column_names2]
print(new_data)
drop_columns = ['ID', 'Quarter_Q2','Bond_rating_Other_Bond_rating', 'Quarter_Q
                'Bond_rating_CCC','Region_North','Stock_rating_Other_Stock_rat
                'Quarter_Q4', 'Quarter_Q5', 'Quarter_Q6', 'Quarter_Q7',
                'Quarter_Q8','Stock_rating_Strong Sell']

# If necessary, adjust fit_intercept
ridge_reg.fit(X_train.drop(drop_columns, axis=1), y_train)  # Remove Q5 if not
```

```
predicted_sales = ridge_reg.predict(new_data.drop(columns=['ID','Bond_rating_B
df = pd.DataFrame({
    'ID': new_data['ID'],
    'Sales': predicted_sales
})

# Display the DataFrame
print(df.head())
df['ID'] = df['ID'].astype(int)
#df.to_csv('submission40.csv', index=False)
```

```
        ID  QuickRatio  InventoryRatio  RevenueGrowth  MarketshareChange  \
0      7.0    0.540634       -0.689962      -0.670787          -0.575680
1      8.0    0.540634        0.738101      -0.177649           0.036252
2     16.0    0.612823       -1.195422       0.658823          -1.530771
3     17.0    0.540634        0.380752       0.658823           0.761041
4     25.0   -1.551101        0.266777      -0.948274          -0.575680
..     ...         ...             ...            ...                ...
145  656.0   -1.151426        0.266777       0.010628           0.761041
146  664.0   -0.552165        1.349148      -0.177649          -1.092894
147  665.0   -0.585413        0.833833       0.658823           0.761041
148  673.0    1.052497        1.050901       0.658823           0.761041
149  674.0    1.052497       -0.890988      -1.191481          -1.092894

     Company_CMP02  Company_CMP03  Company_CMP04  Company_CMP05  \
0                0              0              0              0
1                0              0              0              0
2                1              0              0              0
3                1              0              0              0
4                0              1              0              0
..             ...            ...            ...            ...
145              0              0              0              0
146              0              0              0              0
147              0              0              0              0
148              0              0              0              0
149              0              0              0              0

     Company_CMP06  ...  Bond_rating_BB  Bond_rating_BBB  Bond_rating_CCC  \
0                0  ...               0                0                1
1                0  ...               0                0                1
2                0  ...               0                0                0
3                0  ...               0                1                0
4                0  ...               1                0                0
..             ...  ...             ...              ...              ...
145              0  ...               0                1                0
146              0  ...               0                0                0
147              0  ...               0                0                0
148              0  ...               0                1                0
149              0  ...               1                0                0

     Stock_rating_Hold  Stock_rating_Sell  Region_North  Region_South  \
0                    0                  0             0             1
1                    0                  0             0             1
2                    0                  1             0             0
3                    1                  0             0             0
4                    0                  0             0             0
..                 ...                ...           ...           ...
145                  0                  0             0             0
146                  0                  0             0             1
147                  0                  0             0             1
148                  1                  0             0             0
149                  0                  1             0             0

     Region_West  Industry_Infrastructure  Industry_Metal Fabrication
0              0                        0                           1
1              0                        0                           1
2              1                        1                           0
3              1                        1                           0
4              0                        1                           0
..           ...                      ...                         ...
145            1                        0                           0
```

```
146           0                         0                           1
147           0                         0                           1
148           1                         0                           0
149           1                         0                           0

[150 rows x 92 columns]
     ID        Sales
0   7.0  2638.992091
1   8.0  2230.479509
2  16.0  5508.020258
3  17.0  5047.368477
4  25.0  4717.198180
```

In [ ]:

In [ ]: