Predicting Bike Rentals Using Multiple Linear Regression

A Case Study of BoomBikes

Sathish Madhiyalagan

Objective

The primary objective of this project is to develop a predictive model using multiple linear regression to accurately forecast the daily bike rental counts for BoomBikes. By analyzing various influencing factors such as weather conditions, day of the week, and seasonal trends, the model aims to provide actionable insights that can help BoomBikes optimize their operations, enhance customer satisfaction, and increase revenue.

Software and Libraries

- **Python**: Programming language for implementing the machine learning model.

- Pandas For data manipulation and analysis.
- Numpy For numerical computations.
- Matplotlib and seaborn For data visualization.
- Scikit-learn For building and evaluating the regression model.

Step 1 : Data Import and Preliminary Exploration

Importing and understanding the dataset is the initial step in our analysis. We begin by importing the data into our Python environment and gaining an overview of its structure and contents. Let's explore the dataset's features and their descriptions in data dictionary.

Upon loading and examining the dataset, it becomes evident that our primary objective is to predict the total bike rentals, as represented by the **'cnt'** column. This understanding guides our initial exploration and subsequent analysis.

Step 2 : EDA

- Print information about the dataset, such as the data types of each column and the number of non-null values.
- Calculate and display summary statistics (e.g., mean, standard deviation, min, max) for numerical columns.
- Identify and print the number of missing values in each column to assess data completeness.

- Determine and print the number of duplicate rows in the dataset, if any.
- Plot a histogram to visualize the distribution of the target variable 'cnt', which represents rental counts.

This steps are EDA first steps,its helping to understand its characteristics and identify any potential issues or patterns.

After thoroughly reviewing my dataset, I'm pleased to confirm that everything appears to be in good order, with no missing values, duplicates.

Then we go with **Univariate analysis,Bivariate Analysis and Multivariate Analysis.**

- Univariate Analysis (categorical variables or numerical variables)

  Univariate analysis examines individual variables to understand their distribution and properties, using techniques like histograms and summary statistics.

- Bivariate Analysis (Categorical vs. Categorical ,Categorical vs. Numerical and Numerical vs. Numerical)

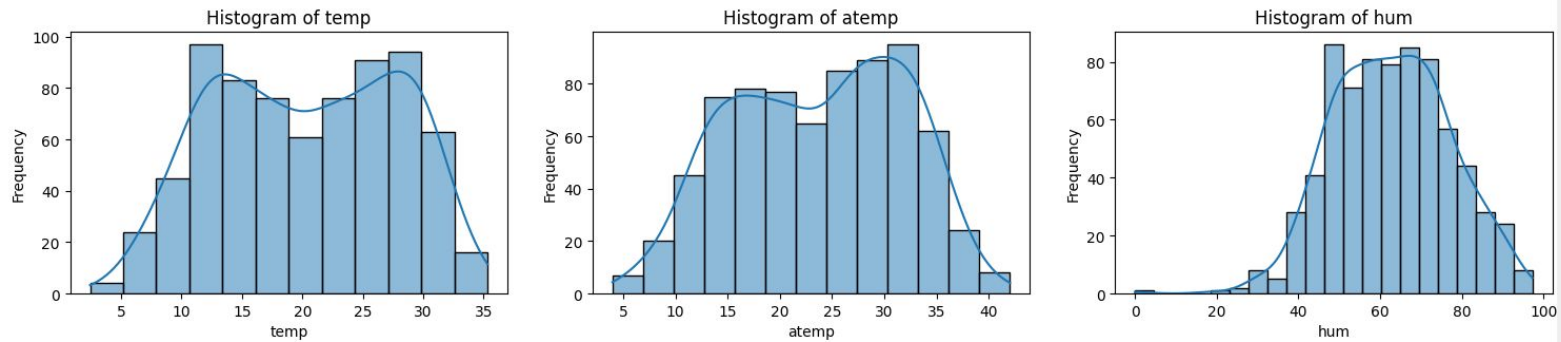
  Bivariate analysis explores relationships between pairs of variables, revealing patterns and correlations through scatter plots and cross-tabulations.

- Multivariate Analysis (Its combined all)

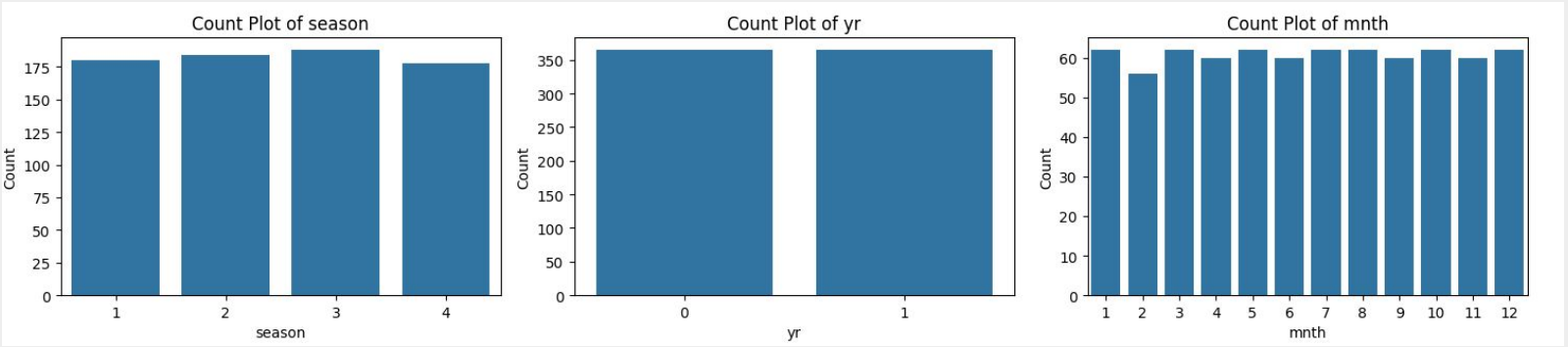
  Multivariate analysis delves into interactions among multiple variables, uncovering complex patterns and relationships using techniques like PCA, factor analysis, and multiple regression.

Univariate analysis

Histograms for numerical variables

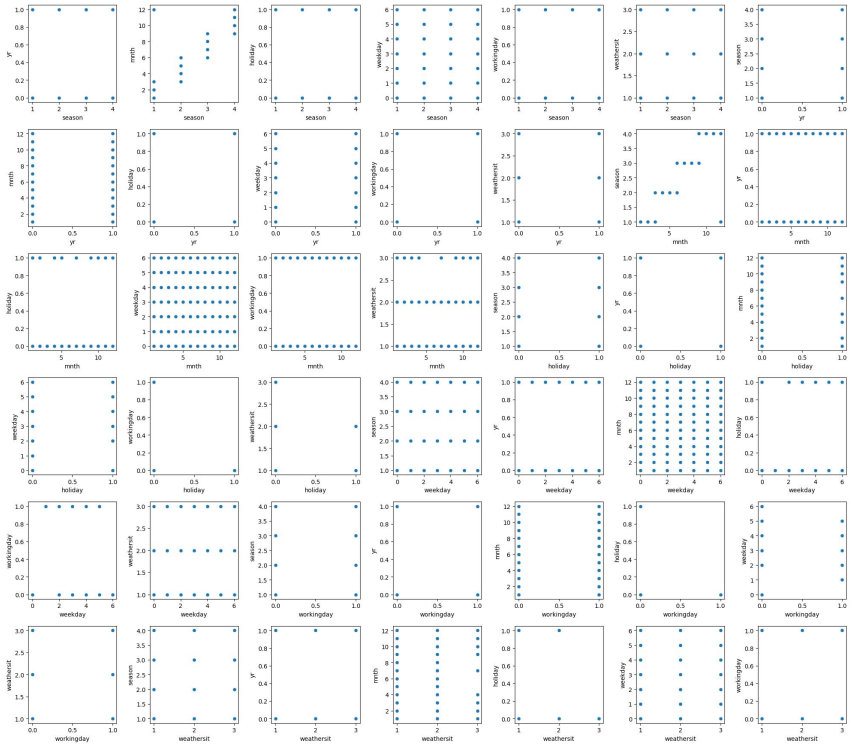


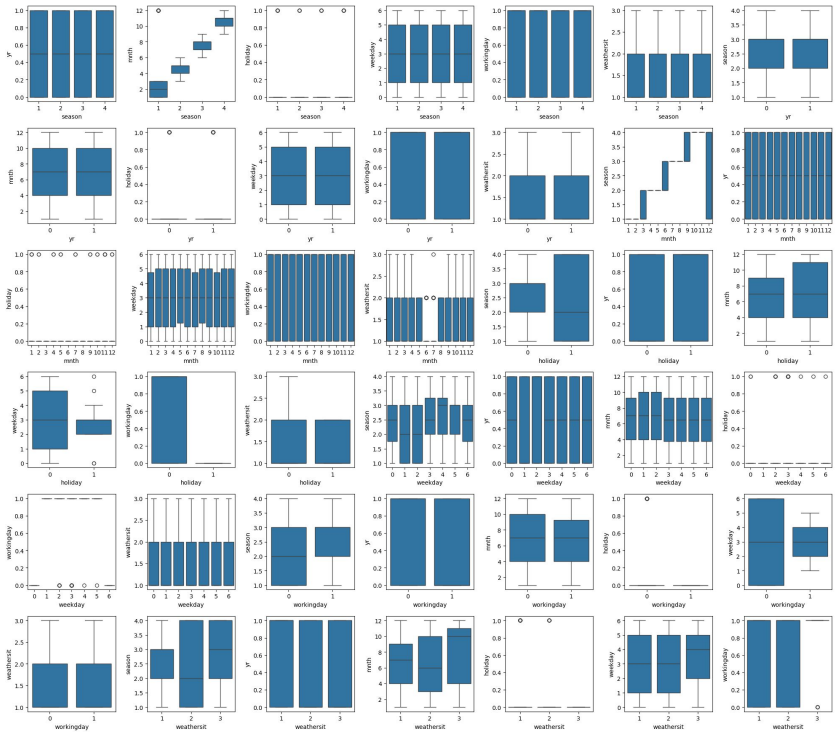
Count plots for categorical variables

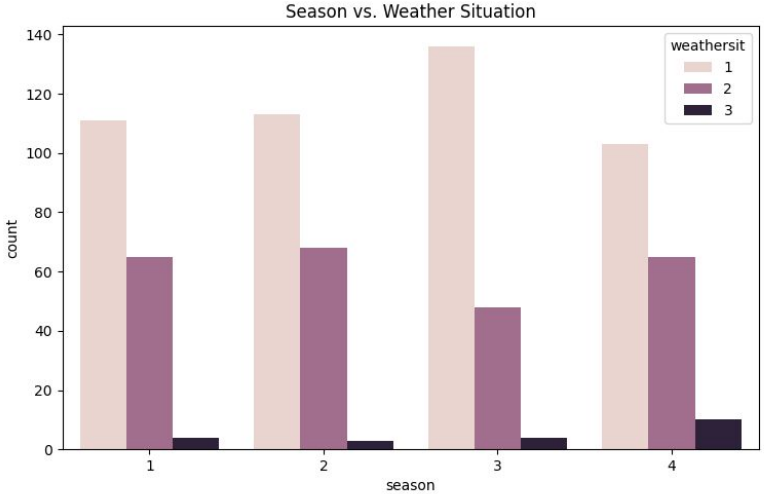


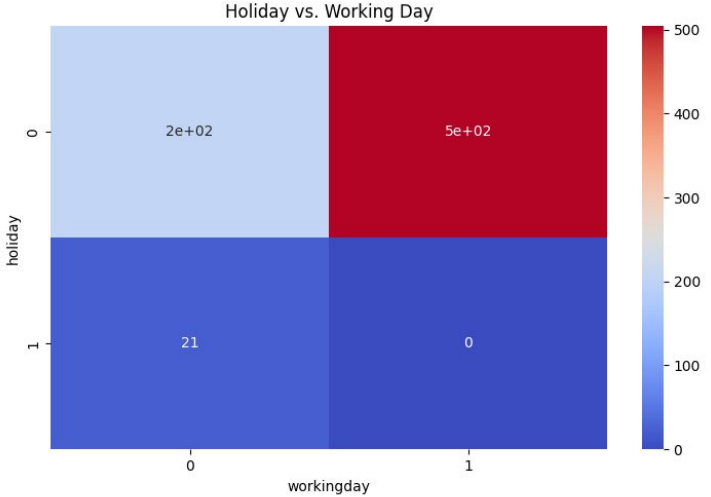
Bivariate Analysis

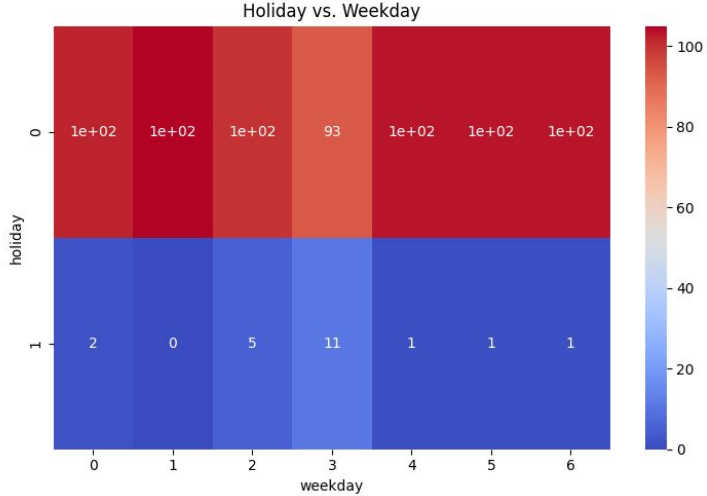
Categorical vs. Categorical

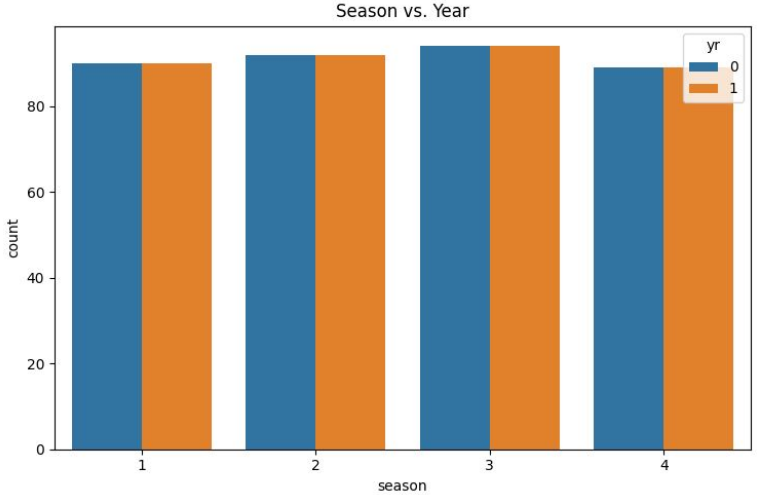




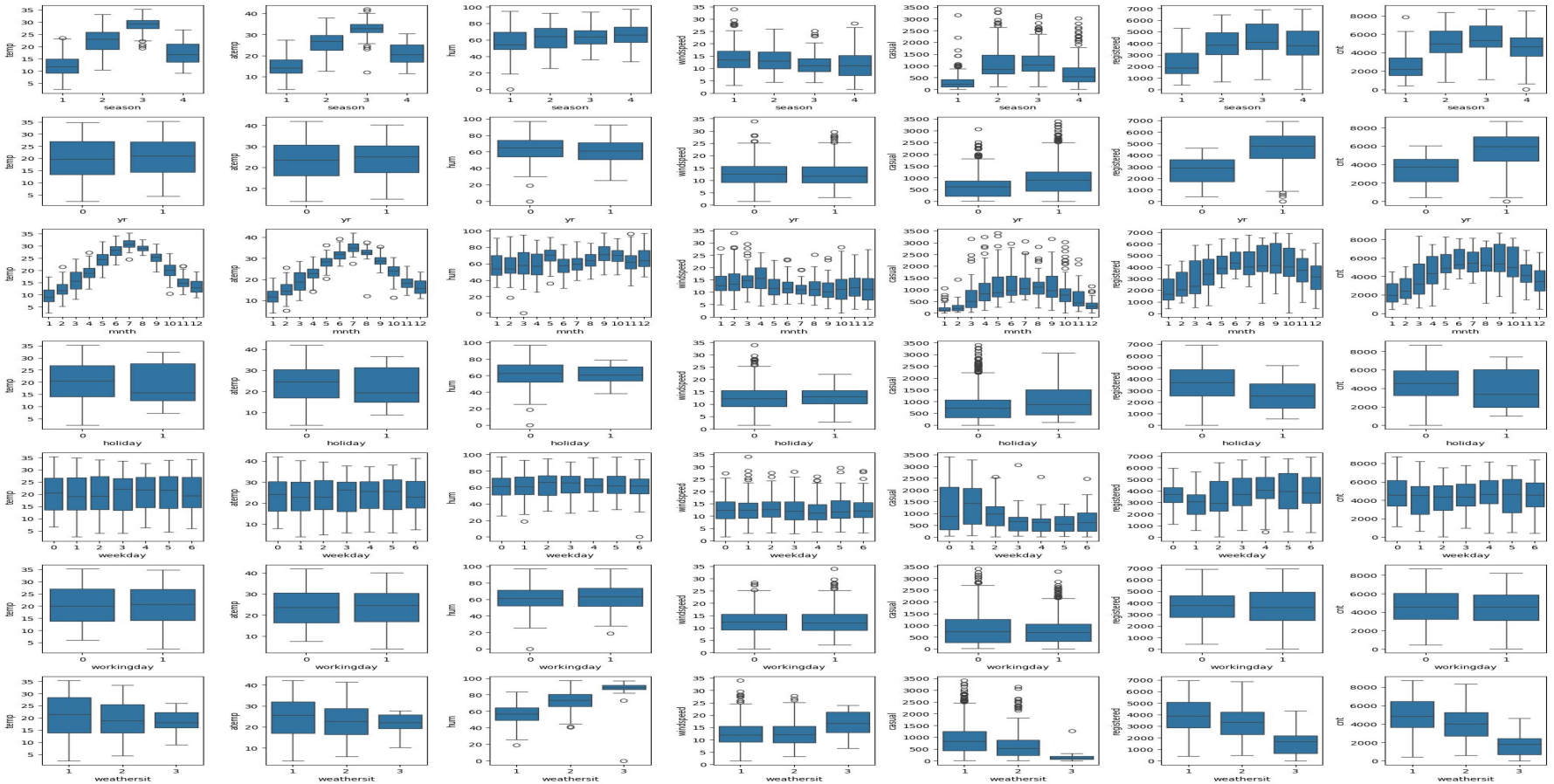

Season vs. Weather Situation
weathersit
1
2
3
count
season


Holiday vs. Working Day
2e+02
5e+02
21
0
holiday
workingday

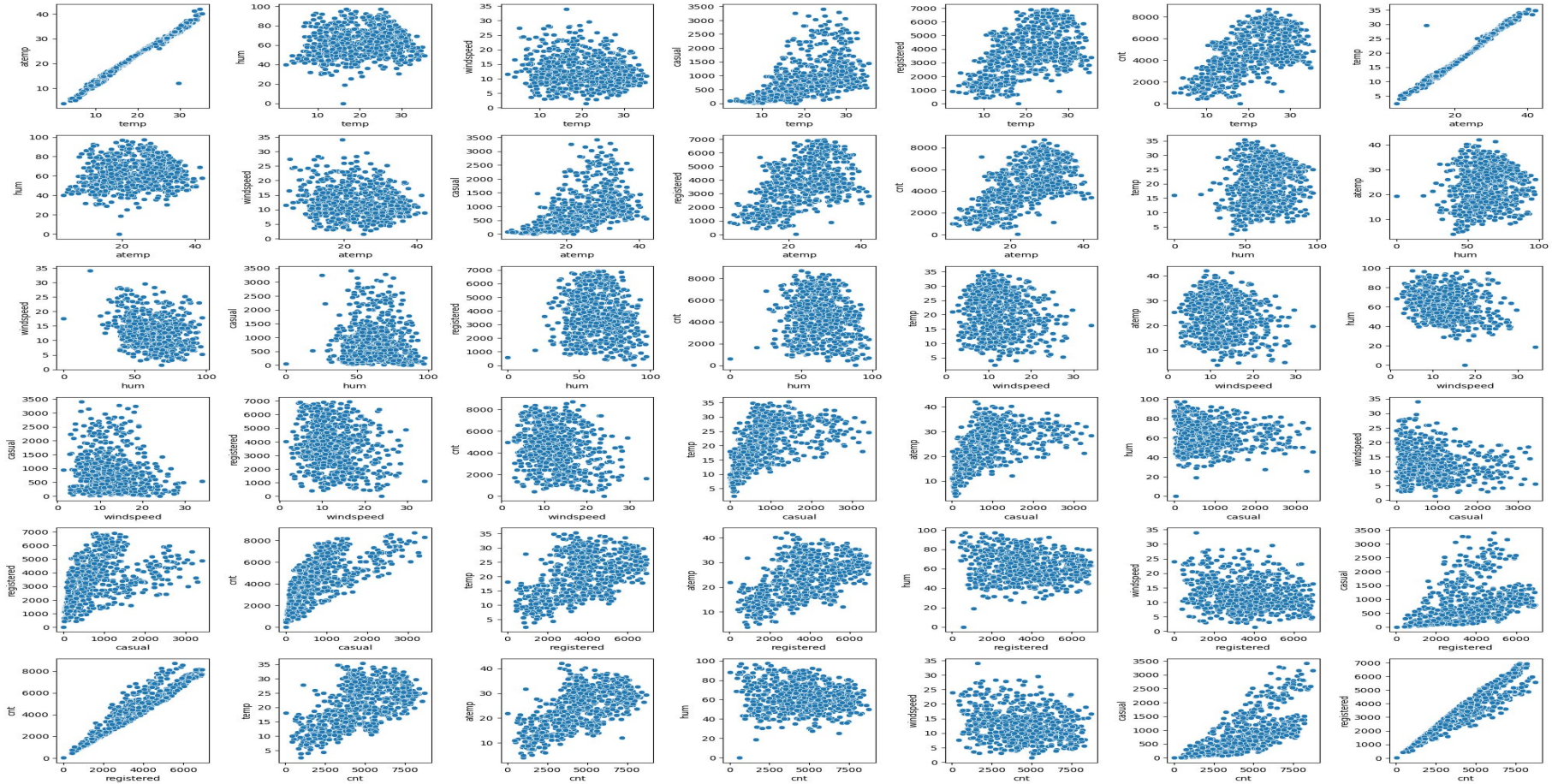

Holiday vs. Weekday
1e+02
93
2
0
5
11
1
holiday
weekday


Season vs. Year
yr
0
1
count
season

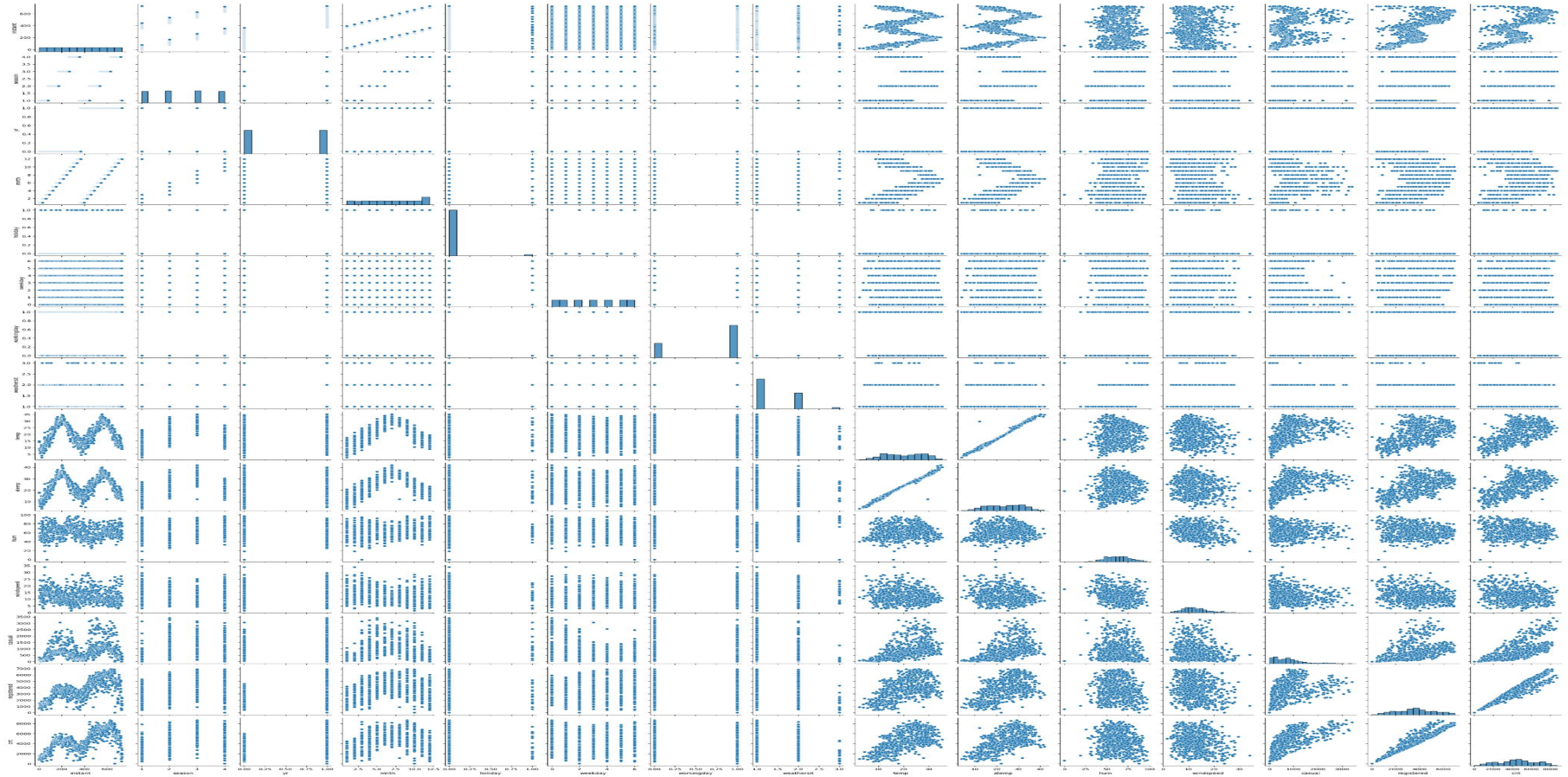
Categorical vs. Numerical



Numerical vs. Numerical



Multivariate Analysis



Step 3 : Pre Processing

In the preprocessing phase, it's crucial to identify and remove any columns that are not relevant to the prediction task or those that do not contribute meaningful information to the model. Based on my statement, it seems four columns that are not correct or necessary for this feature.

That four columns are (instant, dteday, casual, registered)

One more time check ,Missing Values and null values ,Categorical to encode numerical format

Step 4 : Dummy Variables Creation

Convert the selected columns (season, mnth, weekday, and weathersit) to categorical data type. This step is necessary for the next step, where we will create dummy variables.

Generate dummy variables for the categorical columns specified in columns_to_encode. The drop_first=True parameter is used to drop the first category in each column to avoid multicollinearity. This process creates new binary columns representing the presence of each category in the original columns.

Concatenate the original DataFrame boomBikesAfterDrop with the newly created dummy variables (status). This step appends the dummy variable columns to the original DataFrame.

Drop the original categorical columns (season, mnth, weekday, and weathersit) from the combined DataFrame (bick). This step ensures that only the dummy variables remain in the dataset, **preventing redundancy and multicollinearity.**

Step 5 : Splitting the Data into Training and Testing Sets

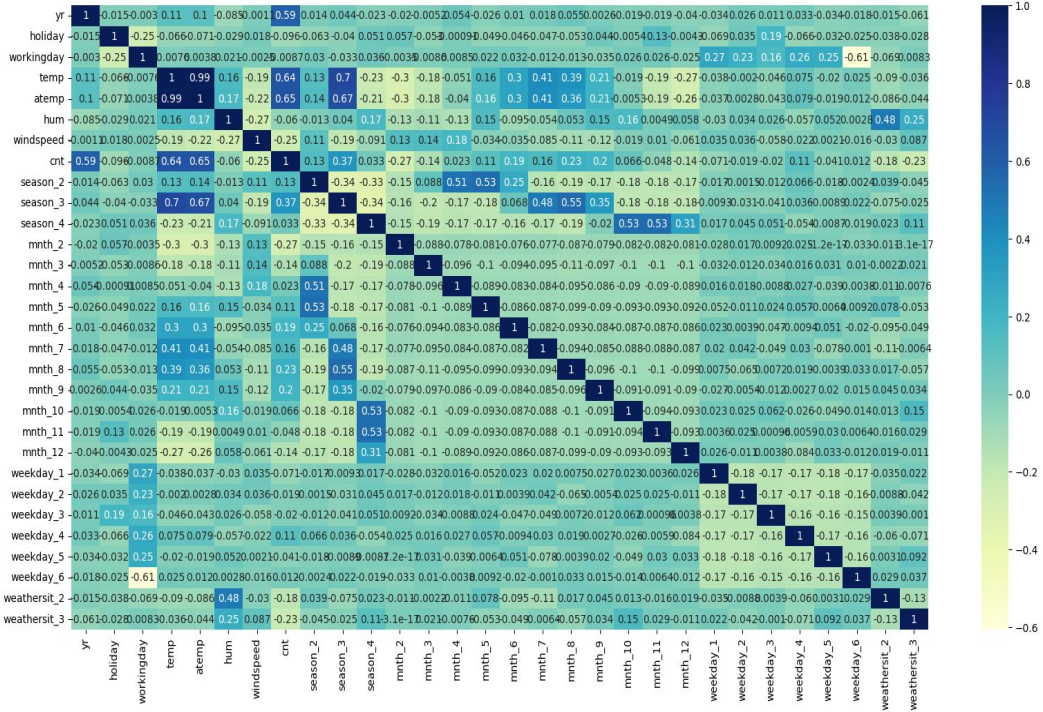
To effectively train and evaluate a machine learning model, it is crucial to divide the dataset into training and testing sets. This allows the model to learn from the training set and be evaluated on unseen data in the testing set, helping to ensure that the model generalizes well to new data.

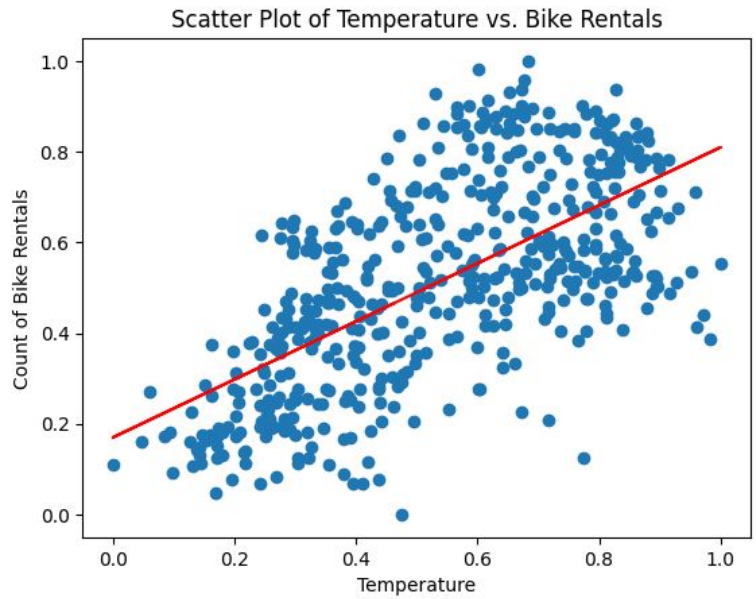
- Import the train_test_split function from the sklearn.model_selection module. This function is used to split the dataset into training and testing sets.
- Set the random seed using np.random.seed(0). This ensures that the splitting of data is reproducible, meaning that every time you run the code, the training and testing sets will have the same rows.
- Use the train_test_split function to split the bickDf DataFrame into training and testing sets. The train_size=0.7 parameter specifies that 70% of the data should be used for training, while the test_size=0.3 parameter specifies that 30% should be used for testing. The random_state=100 parameter ensures that the split is consistent every time the code is run.

Rescaling the Features :

- Specify the list of numerical columns that need to be scaled.Import the MinMaxScaler class from the sklearn.preprocessing module to perform scaling.
- Create an instance of the MinMaxScaler.Fit the MinMaxScaler on the training set's numerical columns and transform the data, scaling the values to a range of [0, 1].
- Display summary statistics for the scaled numerical columns to understand their distribution after scaling.
- Create a heatmap to visualize the correlations between the scaled numerical features in the training set.
- This visualization helps identify strongly correlated features, which can inform feature selection and engineering decisions in the modeling process.

So, we pick temp as the first variable and we'll try to **fit a regression line** to that.





Step 6 : Adding Variable and Features selection

➔ Add Numerical Columns One by One:

- Added the 'temp' column as the first numerical feature.
- Evaluated the model's performance with the added feature using Ordinary Least Squares (OLS) regression.
- Obtained an R-squared value of 0.414, indicating that the 'temp' feature alone explains 41.4% of the variance in the target variable.
- Repeated the process by adding all numerical columns to the feature set.
- Evaluated the final model's performance, which achieved an impressive **R-squared value of 0.851.**
- The final model includes multiple numerical features along with categorical features like season, month, weekday, holiday, and weather conditions.

Step 7: Variance Inflation Factor (VIF) Checking

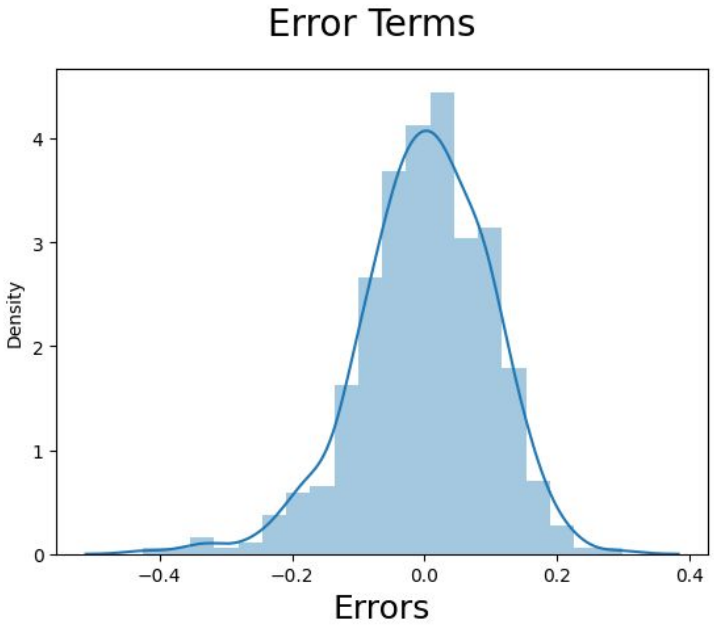
- The code you provided calculates the Variance Inflation Factor (VIF) for each feature variable in your dataset. The VIF helps assess multicollinearity among predictor variables in a regression analysis.
- **Variables with high VIF values, typically above 5,** indicate strong multicollinearity, suggesting that those variables are highly correlated with other predictor variables in the model.
- Based on the results of the VIF calculation, you should identify variables with high VIF values and consider dropping them from your model to address multicollinearity issues
- Look for variables with high VIF values, typically above 5, indicating multicollinearity.

Stop 8: Dropping the variable and updating the model

- Look for variables with high VIF values, typically above 5, indicating multicollinearity.
- ['weekday_1', 'weekday_2', 'weekday_5','weekday_6', 'weekday_4', 'weekday_3','mnth_3','mnth_4','mnth_5','mnth_6','mnth_8','mnth_11','mnth_12','mnth_7','atemp','season_3','season_4','workingday']
- Variables with p-values less than the chosen significance level (usually 0.05) are considered statistically significant.

Stop 9: Residual Analysis of the train data

- Scatter plot of residuals against predicted values.
- Histogram or density plot of residuals to check for normality.
- Q-Q plot to compare the distribution of residuals to a **normal distribution.**



Step 10: Making Predictions Using the Final Model

My final model exhibits a good R-squared value, indicating strong explanatory power, and features low p-values and VIF, suggesting that the selected variables are statistically significant and exhibit low multicollinearity, respectively.

Step 11: Model Evaluation

In the final step of model evaluation, my predictions align closely with the training sample data, indicating that the model performs well and is suitable for further feature development.