**Bike Sharing Analysis**

Using Pyspark and Python

horizontal line

# 

# **Introduction**

With the cost-of-living increasing day by day, and congestion charges levied in many parts of London, the choice of using public transport makes sense to people. But what if you want to travel shorter distances, still have your independence, and maintain a healthy lifestyle – that’s where bike sharing started (is my assumption). Many organizations like Santander offer public bike-sharing schemes with several docking stations across London to help and encourage this lifestyle. However, it is also a challenge to maintain the requisite number of bikes.

**Objective**

The goal of this project is to try and predict the bike share numbers using Machine Learning

### **Aims and Objectives**

1. To identify the best prediction model for the data collected
2. Analysis of the previous data and visualize the hidden patterns using Python and Tableau
3. Check the suitable machine learning classification type (Classification or Regression)
4. Compare the models generated and evaluate the prediction performance

## 

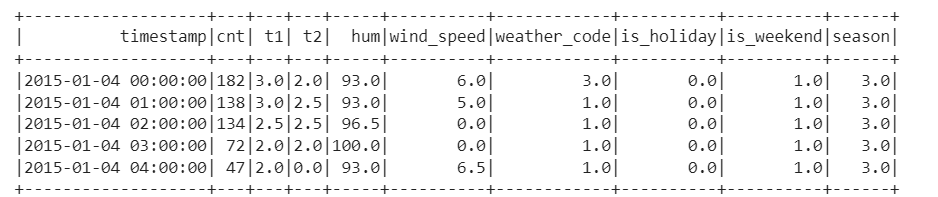
## **Structure of the Project**

1. Loading Data
2. Preprocessing and EDA
3. Feature Engineering for modeling
4. Training the model and model Evaluation
5. Conclusion
6. Analysis using Tableau

### **Loading Data**

To load data in PySpark, we first need to create a PySpark session. After creating the session, we can read the data using the read.csv() function of PySpark. Setting inferSchema and header parameters to True will automatically infer the schema of the data and load the header column as well.

To perform some basic exploration of the data, we can use the show() function to display the first few rows of the data and the printSchema() function to display the schema of the data. Here is an example code snippet:



The column description is as shown below:

* timestamp: This column represents the date and time when the data was collected. It is usually represented in the format of YYYY-MM-DD HH:MM:SS.
* cnt: This column represents the total number of bike rentals during the given timestamp. It is the target variable for prediction models.
* t1: This column represents the temperature in Celsius at the time of the data collection.
* t2: This column represents the temperature in Celsius that it feels like due to the wind speed, humidity, and temperature at the time of data collection.
* hum: This column represents the humidity percentage at the time of the data collection.
* wind\_speed: This column represents the wind speed in km/h at the time of the data collection.
* weather\_code: This column represents the weather code at the time of the data collection. It is a categorical variable that represents the type of weather condition such as clear sky, light rain, heavy rain, snow, etc.
* is\_holiday: This column represents whether the day of the data collection is a holiday or not. It is a binary variable where 1 indicates that it is a holiday and 0 indicates it is not.
* is\_weekend: This column represents whether the day of the data collection is a weekend or not. It is a binary variable where 1 indicates that it is a weekend and 0 indicates it is not.
* season: This column represents the season of the year at the time of the data collection. It is a categorical variable that has different seasons in numerical format.

The printschema() function shows the schema of the data as follows:

| root  |-- timestamp: timestamp (nullable = true)  |-- cnt: integer (nullable = true)  |-- t1: double (nullable = true)  |-- t2: double (nullable = true)  |-- hum: double (nullable = true)  |-- wind\_speed: double (nullable = true)  |-- weather\_code: double (nullable = true)  |-- is\_holiday: double (nullable = true)  |-- is\_weekend: double (nullable = true)  |-- season: double (nullable = true) |
| --- |

Inorder to obtain the shape of data, we have used topandas() to convert the Pyspark dataframe to pandas dataframe as shown below

| print('Without any conversion: ',(data.count(), len(data.columns)))  spark.conf.set("spark.sql.execution.arrow.enabled", "true") pandasDF=data.toPandas() print('By converting from spark to Pandas: ',pandasDF.shape) |
| --- |

So we can observe the output as:

| Without any conversion: (17414, 10) By converting from spark to Pandas: (17414, 10) |
| --- |

Thus, we observe that there are around 17414 rows in the data with 10 columns.

### **Preprocessing and EDA(Exploratory Data Analysis)**

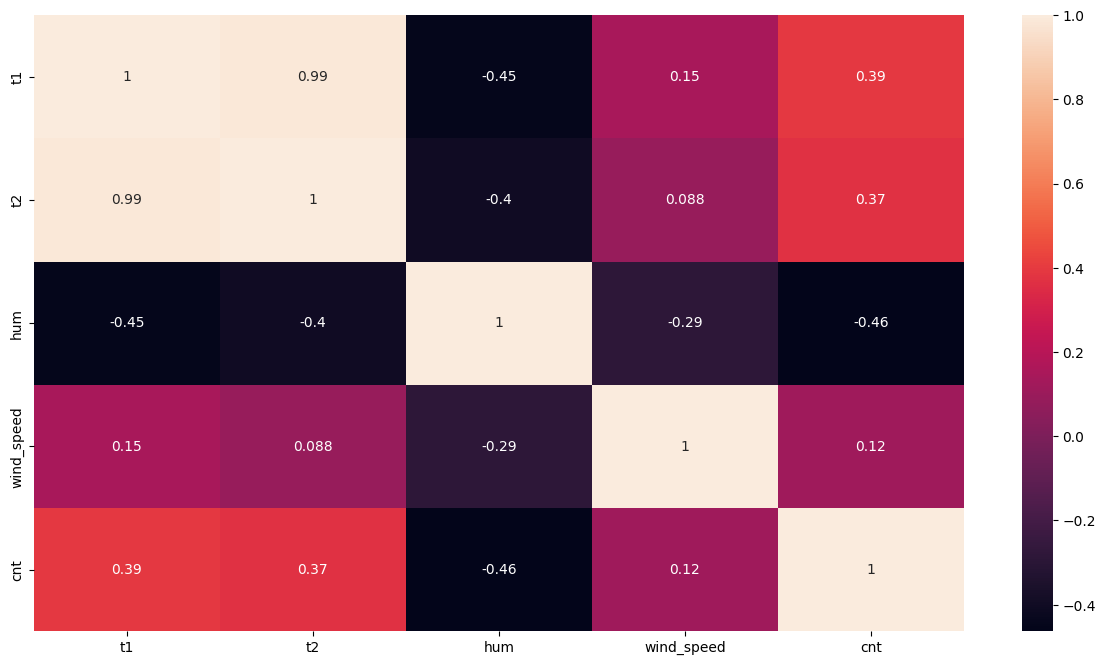
Exploratory Data Analysis (EDA) is a technique used for analyzing data that involves exploring and summarizing the key features of a dataset through statistical and visual means. This includes examining the distribution, central tendency, spread, correlations, outliers, and missing values in the data. The primary objectives of EDA are to comprehend the underlying patterns and relationships in the data, detect potential issues or errors in the data, and generate hypotheses for further investigation or modeling. To conduct EDA on the data, we need to convert the Pyspark dataframe to a pandas dataframe using the toPandas() function since the Pyspark dataframe is not suitable for plotting graphs and visualizations.

We have begun the analysis by checking for null values using the isNull() function and we observed that there are no null values in the data as observed below

| missing in timestamp 0 missing in cnt 0 missing in t1 0 missing in t2 0 missing in hum 0 missing in wind\_speed 0 missing in weather\_code 0 missing in is\_holiday 0 missing in is\_weekend 0 missing in season 0 |
| --- |

At First, we performed statistical analysis using the data.describe() function, we obtained some meaningful insights:

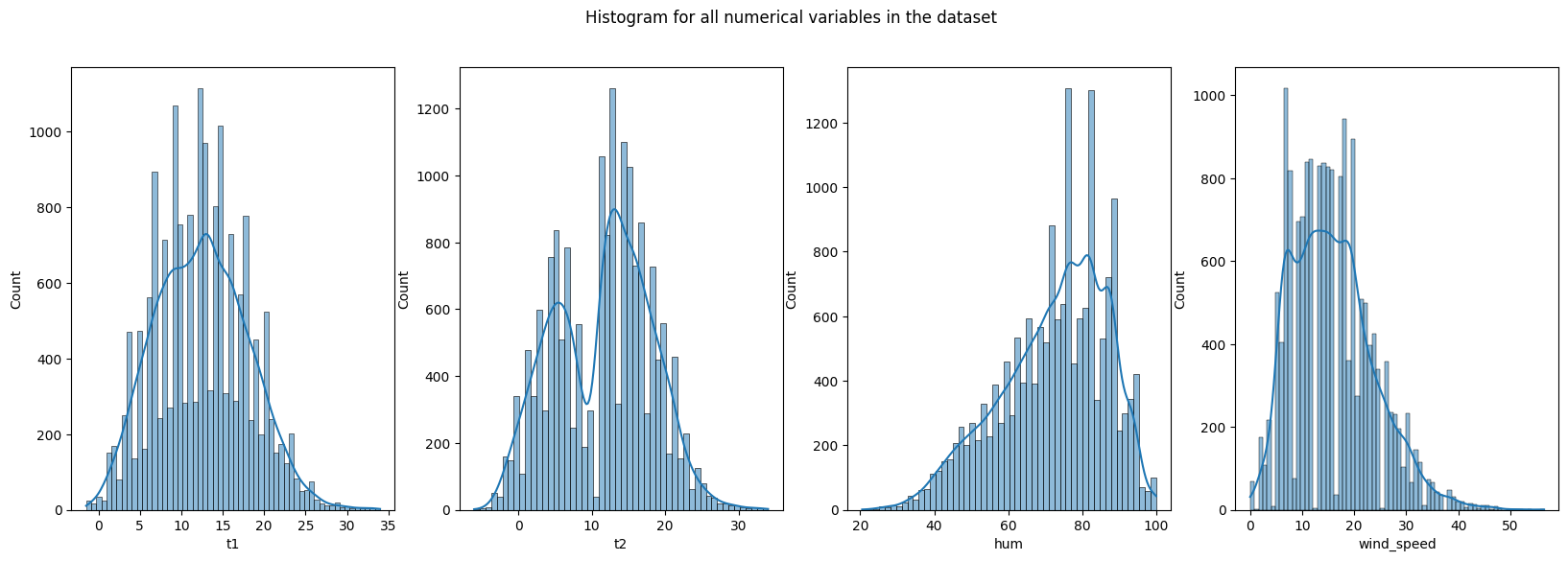
* There are 17414 records in the dataset for all columns.
* The average temperature in column t1 is 12.47 degrees Celsius, and the average temperature in column t2 is 11.52 degrees Celsius.
* The average humidity in column hum is 72.32%.
* The average wind speed in column wind\_speed is 15.91 km/h.
* The average count of bike rentals in column cnt is 1143.
* The standard deviation for t1, t2, hum, wind\_speed, and cnt columns is relatively high, which indicates that the values are spread out from the mean value.
* The minimum temperature in t1 is -1.5 degrees Celsius, while the maximum temperature is 34 degrees Celsius.
* The minimum temperature in t2 is -6.0 degrees Celsius, while the maximum temperature is 34 degrees Celsius.
* The minimum humidity is 20.5%, and the maximum humidity is 100%.
* The minimum wind speed is 0 km/h, and the maximum wind speed is 56.5 km/h.
* There are very few instances of holidays in the column is\_holiday, with an average of 0.022%.
* Around 28.5% of the days in the dataset are weekends, as indicated by the is\_weekend column.
* The dataset contains records from 4 seasons, with an average season value of 1.49.

We have also obtained the correlation plot using corr() function to observe relations between different features in the data.   


Observations from the above correlation plot:

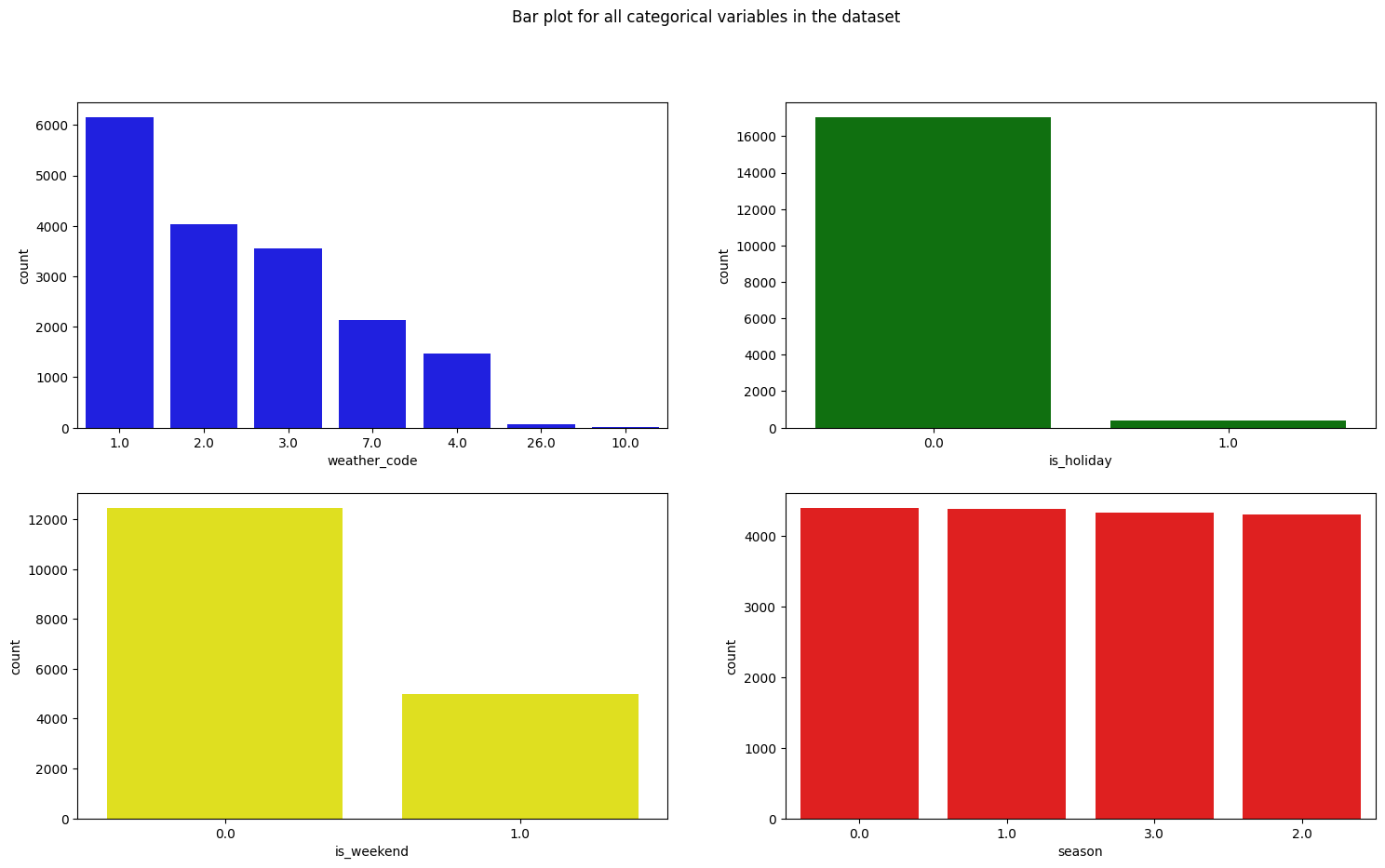
* The correlation coefficient between t1 and t2 is very high (0.988), indicating a strong positive correlation between these two variables.
* There is a negative correlation (-0.448) between t1 and hum, which suggests that as t1 increases, hum decreases.
* The correlation between t2 and hum is also negative (-0.403), indicating that as t2 increases, hum decreases.
* Wind speed has a weak positive correlation with t1 (0.145) and t2 (0.088), indicating that there is a small positive relationship between these variables.
* The correlation between hum and cnt is negative (-0.463), suggesting that as hum increases, cnt decreases.
* There is a positive correlation (0.389) between cnt and t1, which means that as t1 increases, cnt also increases.

Overall, these correlations suggest that temperature (t1 and t2) and humidity (hum) are important factors that influence the bike rental count (cnt), while wind speed has a weaker relationship with cnt.

Performing Univariate analysis  


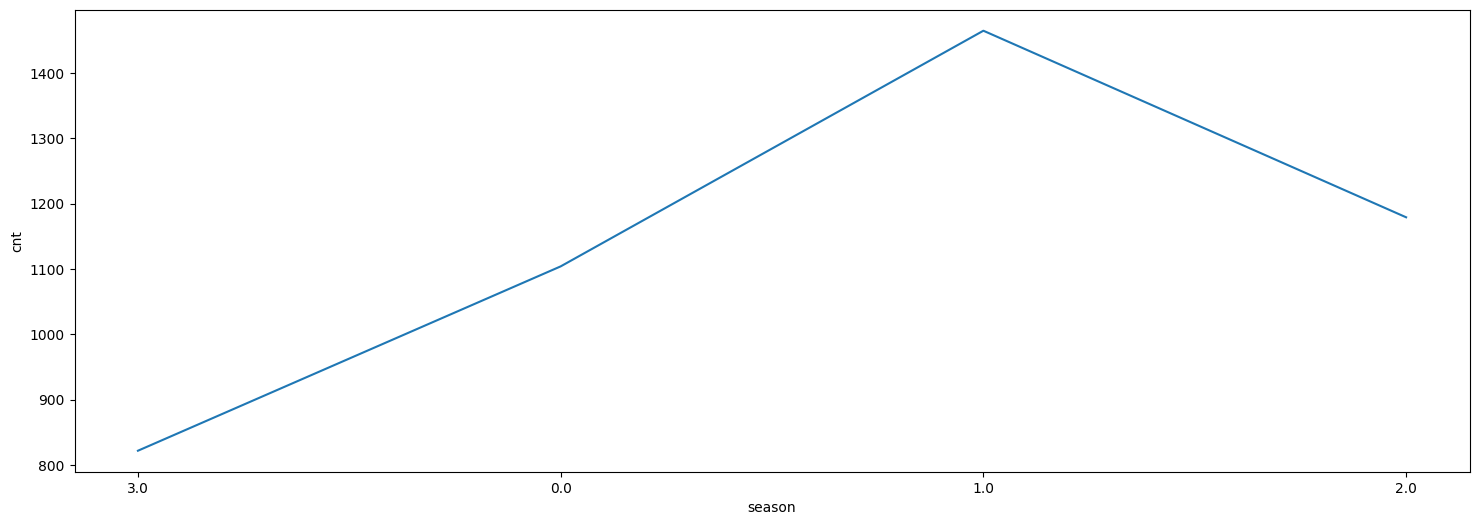
Observations from the above plots

* t1: The t1 seems to follow a gaussian distribution..
* t2: This even seems to follow normal distribution with two different peak regions.
* hum: It seems to have a left skewed distribution and the humidity seems to be mostly high.
* Wind\_speed: The wind\_speed seems to have a right skewed distribution and it is always lower than 50 km/hr.



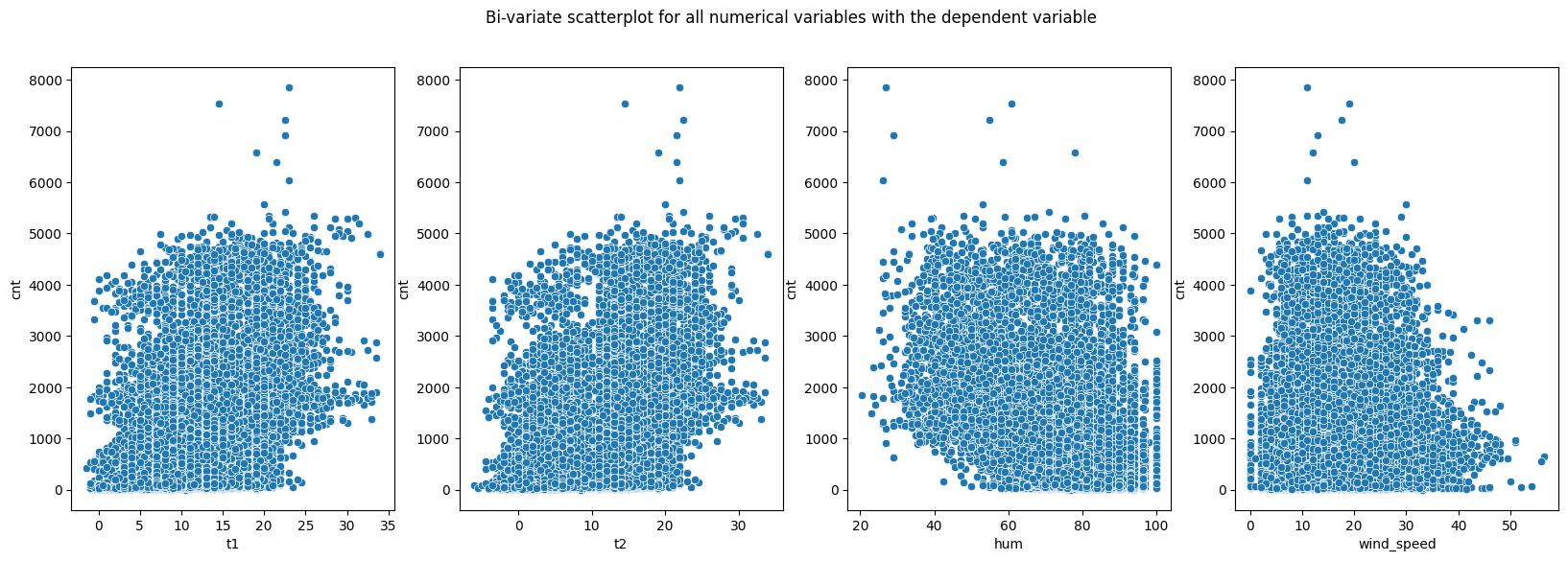
Observations from the above plots

* From all the different weather\_codes, it can be seen that 1 is the most repeating weather code while 26 is the least repeating one.
* We can also observe that there are very few holidays present in the data.
* There are some good number of weekend days given in the dataset.
* The seasons feature seems to be well-balanced with all the 4 different seasons data balanced.

Performing Bivariate analysis:  


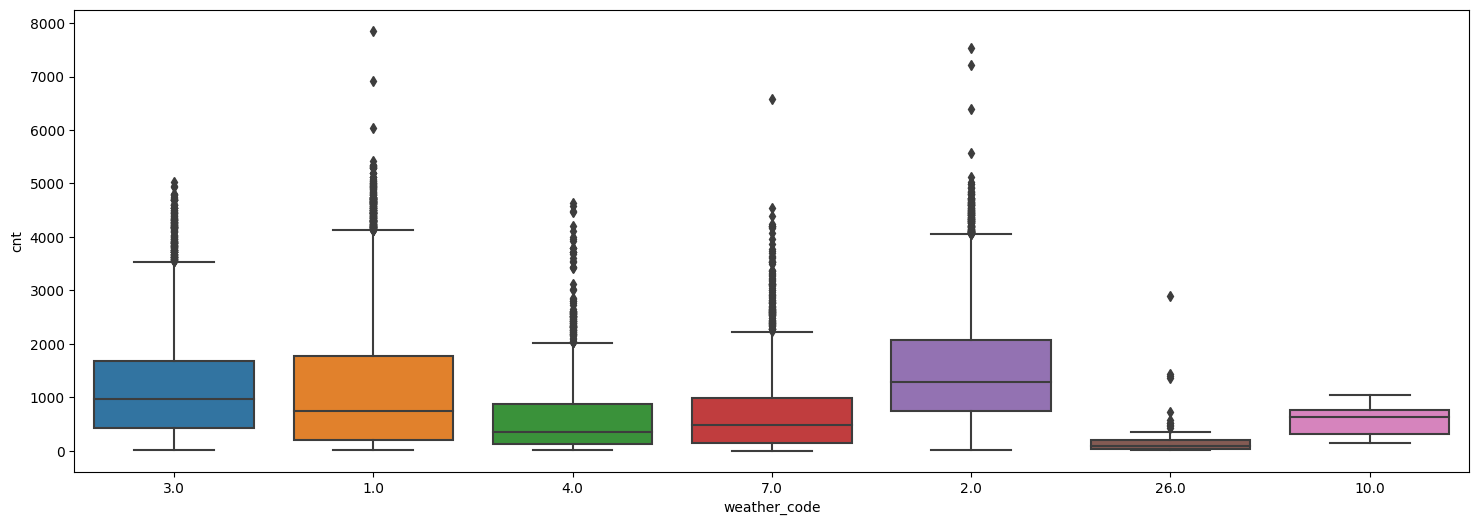
Observations,

* We can observe that there is a peak in cnt values in the 1 season as compared to all other seasons.

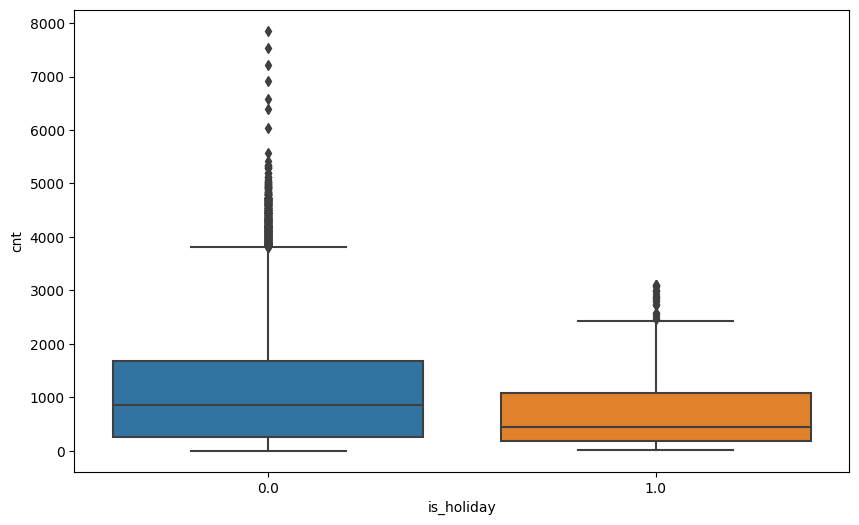


Observations,

* We can observe that there is some noise in the plots and the t1, t2 and wind\_speed seems to have a positive and rightward relation whereas the hum seems to have a negative relation with cnt.



From the above boxplot, we can observe that there are some outliers in the weather\_code column and there are most outliers in the 1 and 2 categories.



From the above boxplot, we can observe that there are some outliers in the 0 category of the is\_holiday category.

### **Feature Engineering**

In machine learning projects, feature engineering plays a critical role in enabling the model to comprehend the data. To achieve this, various feature techniques were applied to both categorical and numerical attributes.

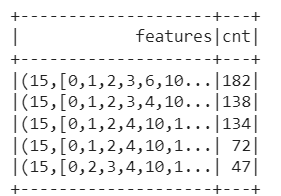
For Categorical attributes - We are using stringindexer to handle the categorical features and convert them into numerical values and then onehotencode the numerical columns. In Pyspark, StringIndexer is a commonly used method for data preprocessing. Its purpose is to convert categorical string features into numerical features. To achieve this, it assigns a unique numerical index to each distinct category present in the feature column. This index is determined based on the frequency of occurrence of each category. This technique is particularly beneficial for machine learning models that require numerical inputs. By converting categorical data into numerical data, it enables such models to make use of the categorical information during training and inference. Code for handling the string attributes is shown below:

| # Handling categorical and string variables from pyspark.ml.feature import StringIndexer, OneHotEncoder  # create object of StringIndexer class and specify input and output column SI\_weather\_code = StringIndexer(inputCol='weather\_code',outputCol='weather\_code\_Index') SI\_is\_holiday = StringIndexer(inputCol='is\_holiday',outputCol='is\_holiday\_Index') SI\_is\_weekend = StringIndexer(inputCol='is\_weekend',outputCol='is\_weekend\_Index') SI\_season = StringIndexer(inputCol='season',outputCol='season\_Index')  # Handle issues SI\_weather\_code.setHandleInvalid("error") SI\_is\_holiday.setHandleInvalid("error") SI\_is\_weekend.setHandleInvalid("error") SI\_season.setHandleInvalid("error")  # transform the data data = SI\_weather\_code.fit(data).transform(data) data = SI\_is\_holiday.fit(data).transform(data) data = SI\_is\_weekend.fit(data).transform(data) data = SI\_season.fit(data).transform(data)  # one hot encoding the columns # create object and specify input and output column OHE\_weather\_code = OneHotEncoder(inputCols=['weather\_code\_Index'],outputCols=['weather\_code\_OHE']) OHE\_is\_holiday = OneHotEncoder(inputCols=['is\_holiday\_Index'],outputCols=['is\_holiday\_OHE']) OHE\_is\_weekend = OneHotEncoder(inputCols=['is\_weekend\_Index'],outputCols=['is\_weekend\_OHE']) OHE\_season = OneHotEncoder(inputCols=['season\_Index'],outputCols=['season\_OHE'])  # transform the data data = OHE\_weather\_code.fit(data).transform(data) data = OHE\_is\_holiday.fit(data).transform(data) data = OHE\_is\_weekend.fit(data).transform(data) data = OHE\_season.fit(data).transform(data) |
| --- |

Feature Engineering on Numerical attributes - To process the numerical features, we utilized a Pyspark function called VectorAssembler. This feature enables the combination of various input columns into a vector column. By providing a list of input column names, VectorAssembler generates a vector column in which each vector element corresponds to the value of a specific input column. This is beneficial when preparing data for machine learning models that require input data in vector format.

| assembler=VectorAssembler(inputCols=[  't1',  't2',  'hum',  'wind\_speed',  'weather\_code\_OHE',  'is\_holiday\_OHE',  'is\_weekend\_OHE',  'season\_OHE'  ],outputCol="features") data=assembler.transform(data) |
| --- |

And the output of the vectorassembler method is as shown below:



Then we have created a final data with the features and the cnt features and then we split the data into train and test with 80% of training data and 20% testing data.

### **Training the Model and Model Evaluation**

We will be using the features and the cnt that were created using the vectorassembler. We will be using different algorithms and then compare them to choose the best model, we would start with the simplest linear regression model

#### Linear Regression:

Linear regression is a statistical method that is commonly used to study the relationship between a dependent variable and one or more independent variables. The basic idea behind linear regression is to find the best-fit line that summarizes the relationship between the variables. This line is obtained by minimizing the sum of the squared differences between the predicted values and the actual values.

The equation for a simple linear regression model with one independent variable can be written as y = mx + b, where y is the dependent variable, x is the independent variable, m is the slope of the line, and b is the intercept. The slope of the line represents the change in the dependent variable for a one-unit increase in the independent variable.

Linear regression can be used for various purposes, such as predicting future values of the dependent variable or understanding the strength and direction of the relationship between variables. It is also commonly used in machine learning and data analysis, as it provides a simple and effective way to model and analyze data. The code for linear regression in our project is as follows,

| from pyspark.ml.regression import LinearRegression lr = LinearRegression(labelCol="cnt",maxIter=50) model=lr.fit(train\_data) |
| --- |

Here we have used the linear regression model. This model initializer requires one main input which is the name of the label column to show the labels. We would then fit this model on the training data. It is a bit different from sklearn implementation of linear regression as there we can initialize the model directly without any parameters but here we need to initialize the model with some parameters as shown above. We can also evaluate the model and obtain various scores as shown below

| **Metric** | **Score** |
| --- | --- |
| R2 | 0.2740178895553639 |
| Mean Absolute Error | 674.5295175120749 |
| RMSE | 913.6362866137125 |

Based on the scores, it appears that the linear regression model has a relatively low R2 value of 0.274. This suggests that only about 27.4% of the variability in the dependent variable is explained by the independent variable(s) included in the model. Generally, a higher R2 value indicates a better fit of the model to the data.

Additionally, the mean absolute error (MAE) of 674.53 suggests that, on average, the model's predictions deviate from the actual values by about 674.53 units. This indicates that the model may not be accurately capturing the true relationship between the variables.

The root mean squared error (RMSE) score of 913.64 is also relatively high, which suggests that the model's predictions have a wide range of error. This could be due to factors such as the inclusion of irrelevant or non-linear variables in the model, or a lack of sufficient data for accurate modeling.

#### Random Forest Regressor:

Random forest regressor is a popular machine learning algorithm that can be used for regression tasks. It is an ensemble method that combines multiple decision trees to improve the accuracy and reduce overfitting. Each tree in the forest is trained on a random subset of features and samples, and the final prediction is obtained by averaging the predictions of all the trees.

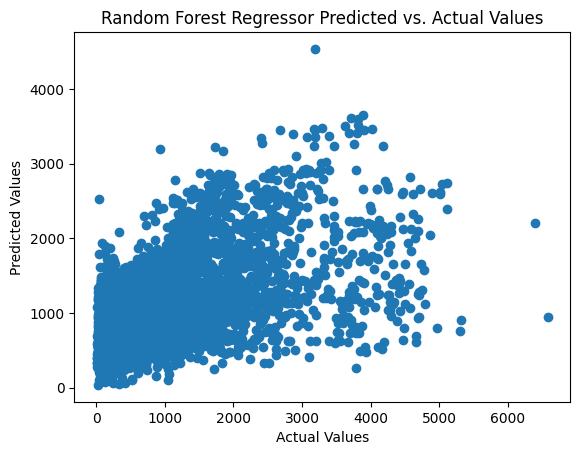
One of the main advantages of a random forest regressor is its ability to handle high-dimensional datasets with complex interactions between features. It is also a robust algorithm that can handle noisy data and outliers. Moreover, it provides an estimate of feature importance, which can be useful for feature selection and interpretation of the results. We have used this same model to fit on the data and evaluate it as shown below

| from pyspark.ml.regression import RandomForestRegressor from pyspark.ml.evaluation import RegressionEvaluator  rf = RandomForestRegressor(numTrees=10, maxDepth=5, seed=42,labelCol="cnt")  rfModel = rf.fit(train\_data) predictions = rfModel.transform(test\_data)  *# Evaluate the Random Forest Regressor model using RMSE metric* evaluator = RegressionEvaluator(labelCol="cnt", predictionCol="prediction", metricName="rmse") rmse = evaluator.evaluate(predictions) print("Root Mean Squared Error (RMSE) on test data = %g" % rmse) |
| --- |

We can observe that this model even requires the label column name as the mandatory initializer else it would not be able to understand the target feature and return an error that the label column is not found.

We can also observe that the RMSE of this model is 905.73 which is slightly better than the linear regression model. We can also look at the feature importance obtained by this model as

| Feature Importances (in descending order): t2: 0.5133200995828704 timestamp: 0.1551162352481937 t1: 0.1437324258761748 weather\_code: 0.048645384850686826 season\_Index: 0.0340480156397658 is\_holiday\_Index: 0.03254128473083003 is\_holiday: 0.016091484377524785 weather\_code\_OHE: 0.014106912424329609 is\_weekend: 0.01340815460499609 wind\_speed: 0.010102092309053517 season: 0.009980805906650063 is\_weekend\_Index: 0.004528658501719446 hum: 0.0043677599496178676 weather\_code\_Index: 1.068599758711567e-05 cnt: 0.0 |
| --- |

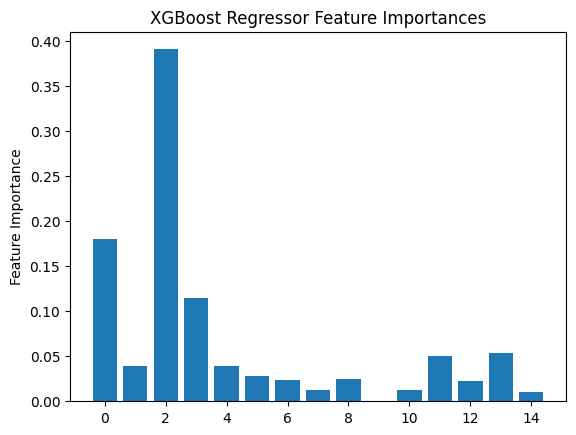
We can also look at the original values and predicted values using a scatter plot as shown below  


#### XGBoost Regressor:

XGBoost Regressor is an advanced machine learning algorithm that uses gradient boosting for regression tasks. It is designed to handle large datasets and can handle missing values effectively. The algorithm works by creating multiple decision trees iteratively and gradually improving the model's accuracy by combining the outputs of these trees. XGBoost Regressor is popular in various fields, including finance, healthcare, and e-commerce, due to its high performance and flexibility. We have also created the XGBoost model and evaluated it as shown below

| from pyspark.ml.regression import GBTRegressor from pyspark.ml.evaluation import RegressionEvaluator  gbt = GBTRegressor(labelCol="cnt",maxDepth=5, maxBins=32, minInstancesPerNode=1, minInfoGain=0.0, maxMemoryInMB=256, cacheNodeIds=False, checkpointInterval=10, lossType="squared", maxIter=20, stepSize=0.1, seed=42)  gbtModel = gbt.fit(train\_data) predictions = gbtModel.transform(test\_data)  *# Evaluate the XGBoost Regressor model using RMSE metric* evaluator = RegressionEvaluator(labelCol="cnt", predictionCol="prediction", metricName="rmse") rmse = evaluator.evaluate(predictions) print("Root Mean Squared Error (RMSE) on test data = %g" % rmse) |
| --- |

For this model, we can observe that the RMSE value is 888.499 which is better than the Random Forest model and Linear Regression model. For this model, We have plotted the feature importance as shown below



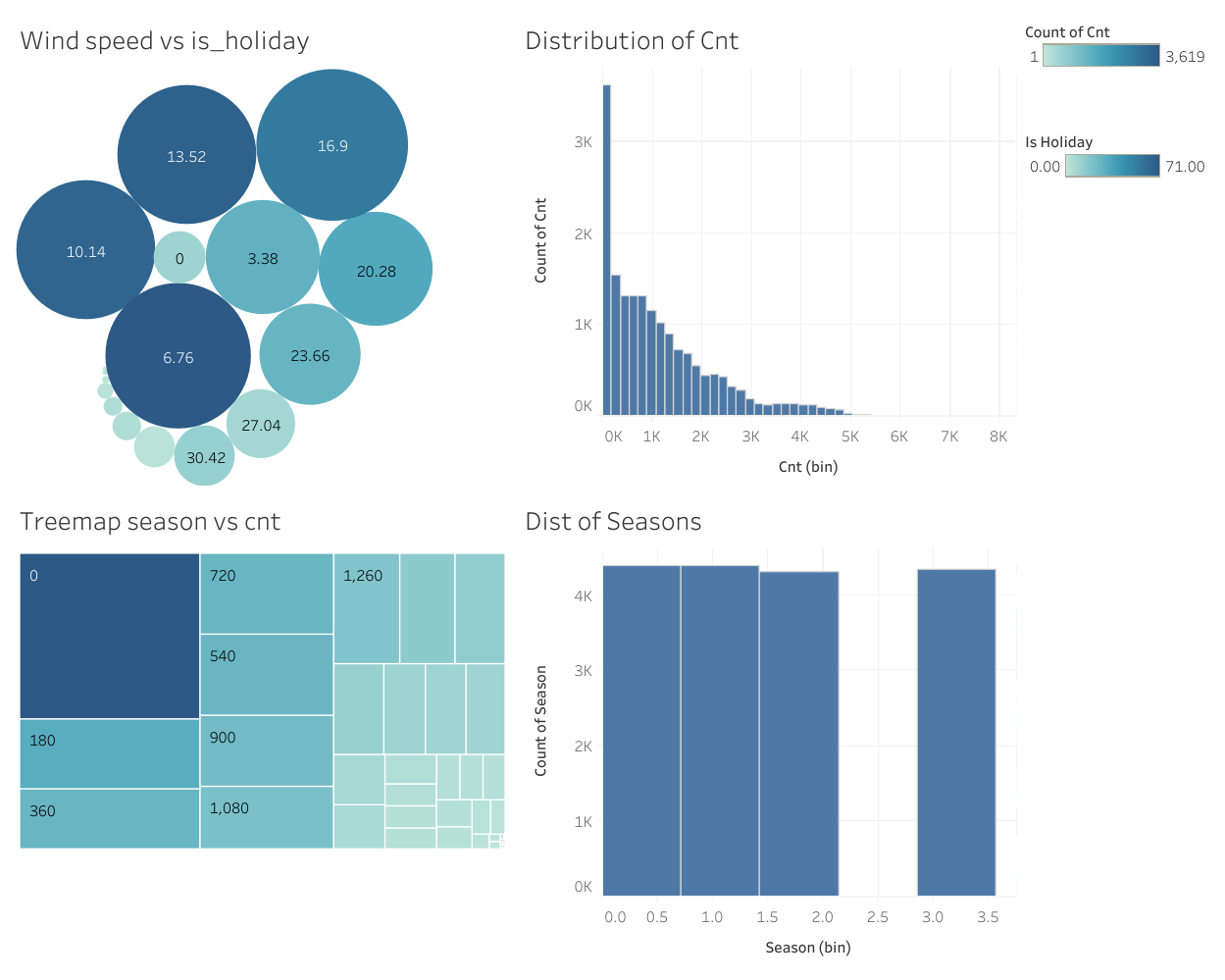
### **Insights**

* The data shows that there are 17414 rows and 10 columns which also includes a count of bike shares column(cnt).
* We have also observed that there are no null values in the data.
* We can also observe that the climate is relatively humid with a moderate wind\_speed as observed from the data.
* We have also observed that the holidays are infrequent in the data
* The weekends are also slightly overrepresented in the data.
* The cnt feature shows that the average bike rental count is 1,143, with a standard deviation of 1,085, indicating that the number of bike rentals varies widely across different observations in the dataset.
* We have also observed that out of the three models used, the XGBoost regressor model has the best performance over all the 3 other models.

### **Conclusion**

In conclusion, the bike sharing dataset was used to train three different regression models: linear regression, random forest regressor, and xgboost regressor. The performance of the models was evaluated using the RMSE metric, and the xgboost regressor outperformed the other models with an RMSE of 888.499. The dataset consisted of various features such as temperature, humidity, wind speed, weather code, holiday, weekend, and season, with the cnt column being the target variable. The dataset showed that temperature, humidity, and wind speed had a significant impact on the number of bikes rented. Overall, the xgboost regressor was the best model for predicting the number of bikes rented based on the given features. Therefore, the bike sharing company can use the xgboost regressor to forecast the demand for bikes based on weather conditions, holidays, weekends, and seasons.

### **Some Comparison Plots using Tableau**



From the above tableau dashboard, we can observe that the seasons are equally distributed and have about the same number of images, we have also observed that the cnt feature is right-skewed. We have also observed that there are higher wind speeds and that are common and there is no correct relation between them and a holiday and finally we can observe a treemap between season and cnt features.

## **Technology:**

Jupyter notebook, Python, Pyspark, Pandas, ML Algorithms Tableau,.