# Regression Modeling on Bike Sharing Data Set

T. Sai Harish Sarma

# Content

- Business Context
- Problem Statement
- Data Cleaning & EDA
- Feature Engineering
- Pre Processing Data
- Target Conditioning
- Modelling, Evaluation & Tuning
- Model Explainability &Conclusion

# Business Context

✓Seoul is the official capital city of South Korea.

✓Seoul Metropolitan Government provides Rental Bike Service to the public.

✓These Public Bikes are designed to be used by all women, the elderly and the infirm.

✓These Bikes are made light weight and durable.

# Problem Statement

Currently Rental bikes are introduced in many urban cities for the enhancement of mobility comfort. It is important to make the rental bike available and accessible to the public at the right time as it lessens the waiting time. Eventually, providing the city with a stable supply of rental bikes becomes a major concern. The crucial part is the prediction of bike count required at each hour for the stable supply of rental bikes.

# Data Cleaning

```
### Getting info about null / NaN values count in each column
data.isna().sum()
```

```
Date                           0
Rented Bike Count              0
Hour                           0
Temperature(°C)                0
Humidity(%)                    0
Wind speed (m/s)               0
Visibility (10m)               0
Dew point temperature(°C)      0
Solar Radiation (MJ/m2)        0
Rainfall(mm)                   0
Snowfall (cm)                  0
Seasons                        0
Holiday                        0
Functioning Day                0
dtype: int64
```

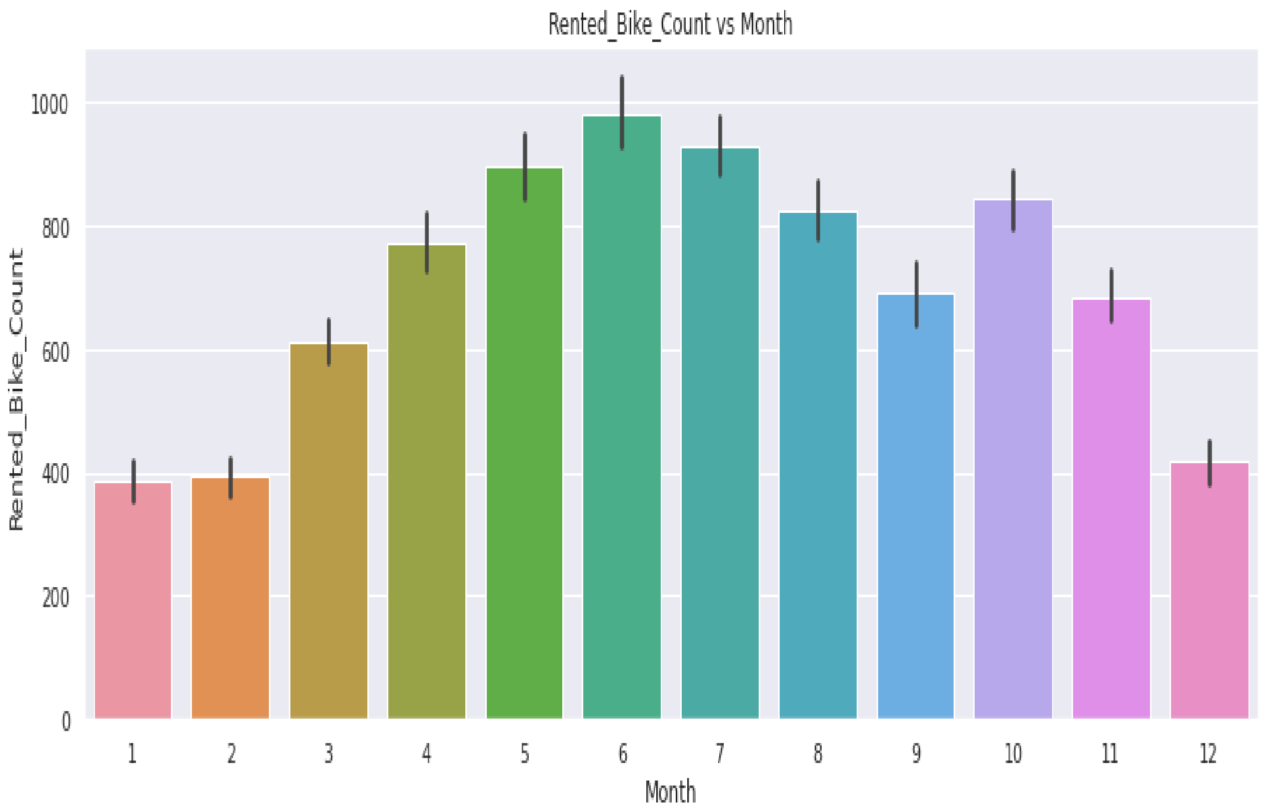
```
[8]  ## Convering Data type of Date Column to Date Time Format
     data['Date'] = pd.to_datetime(data['Date'])
```

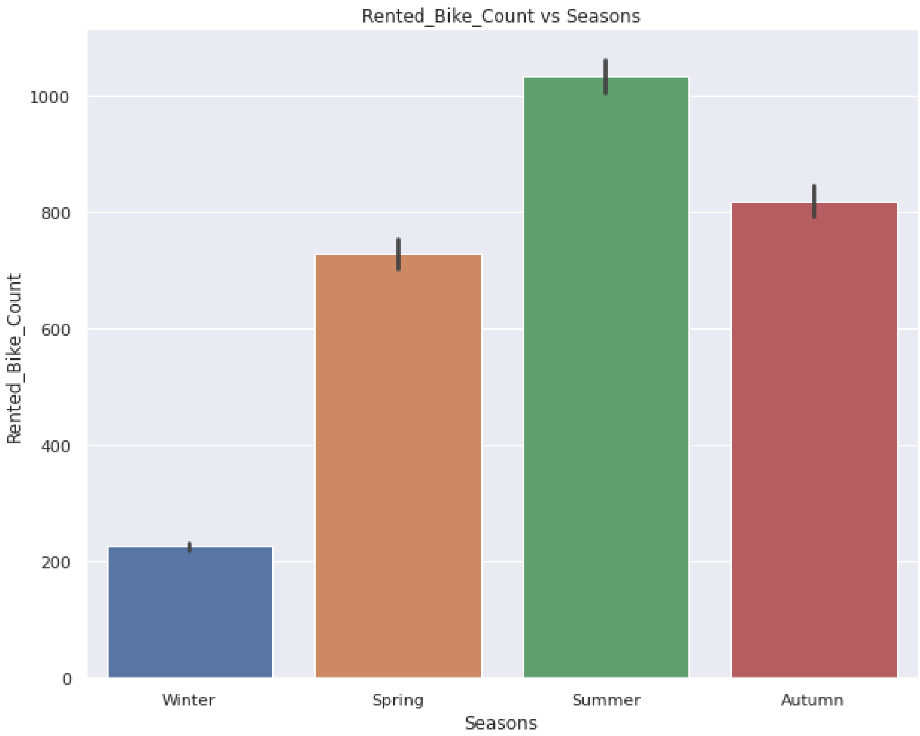
```
[9]  ## Extracting Week Day, Month, Year from Date
     data['Month'] = data['Date'].dt.month
     data['Year'] = data['Date'].dt.year
```

```
## Renaming Columns
data = data.rename(columns={'Rented Bike Count': 'Rented_Bike_Count', 'Temperature(°C)': 'Temperature',
                            'Humidity(%)': 'Humidity', 'Wind speed (m/s)': 'Wind_speed', 'Visibility (10m)': 'Visibility',
                            'Dew point temperature(°C)'  :'Dew_point',  'Solar Radiation (MJ/m2)': 'Solar_Radiation',
                            'Rainfall(mm)': 'Rainfall',  'Snowfall (cm)': 'Snowfall', 'Functioning Day': 'Functioning_Day'})
```

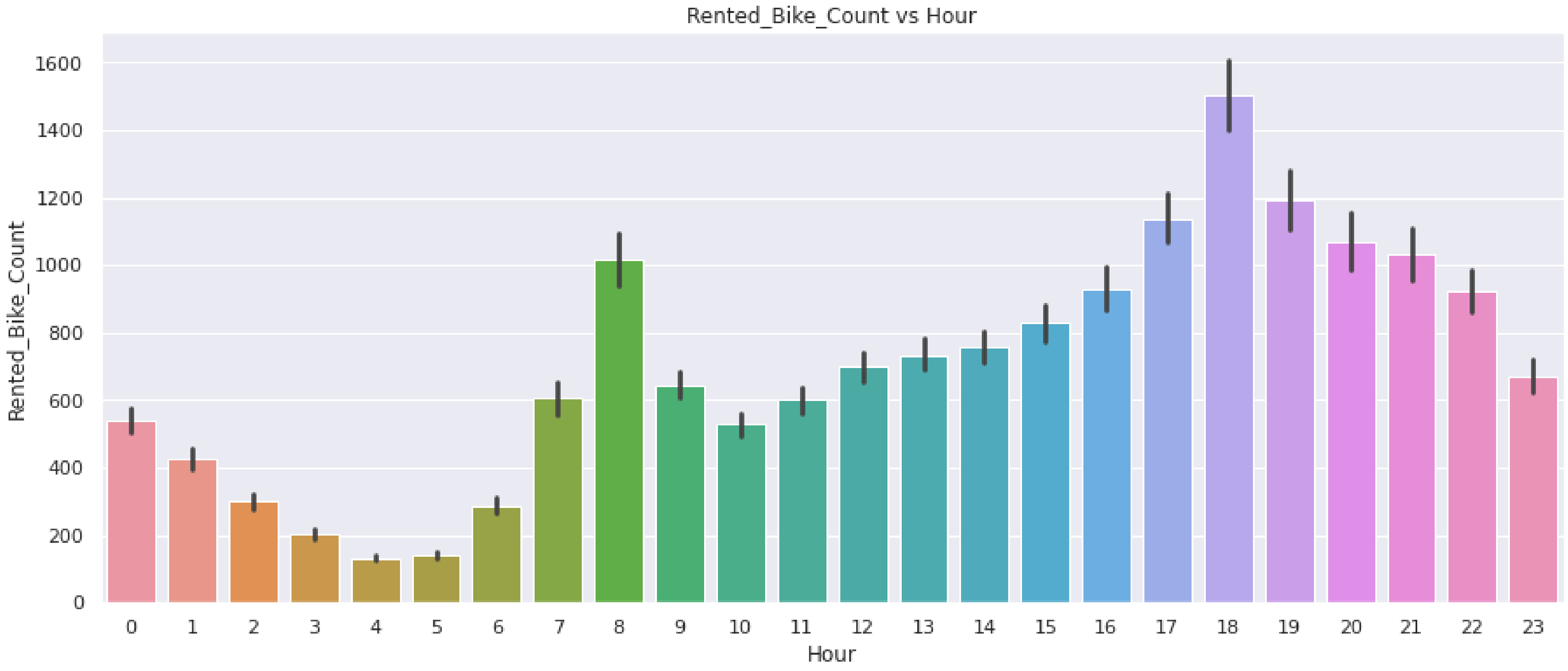
```
[11] ### Dropping Date Column after extraction
     data.drop(columns = ['Date'], inplace =True)
```

# Exploratory Data Analysis

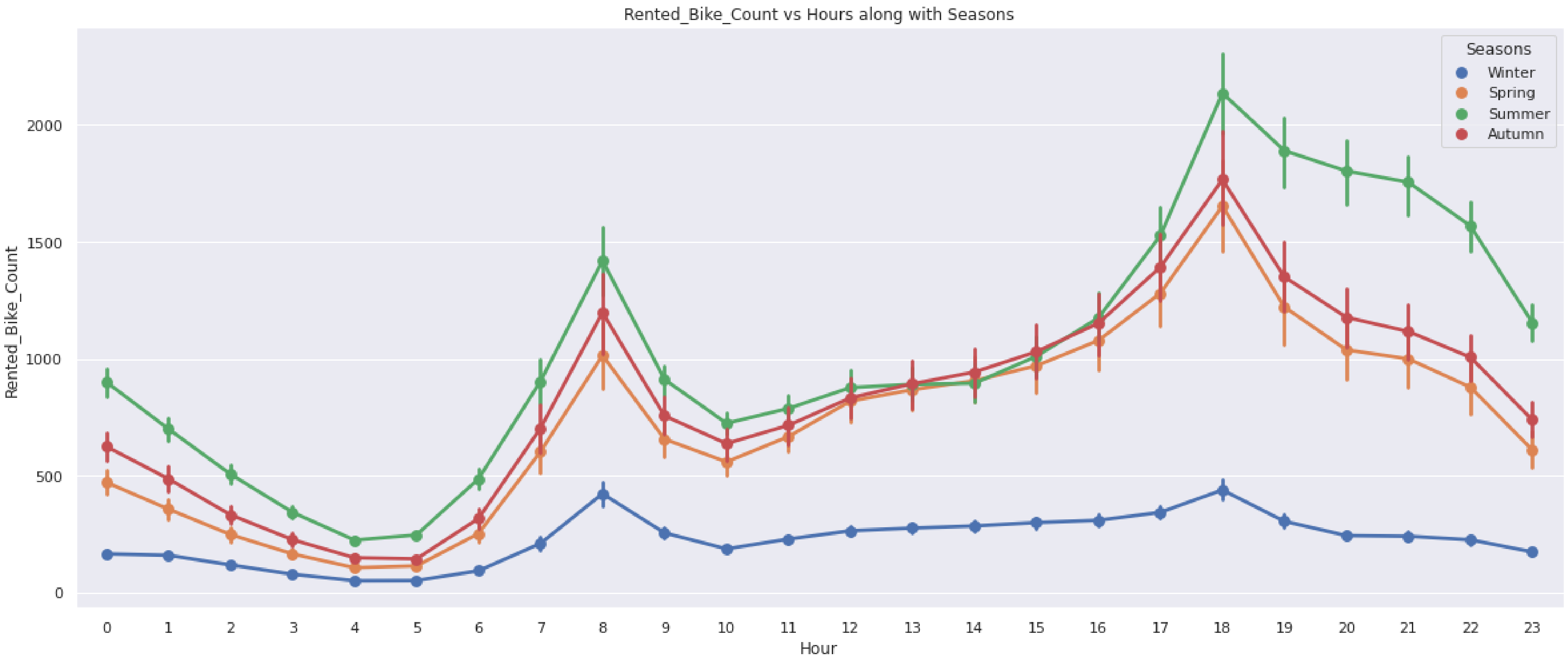


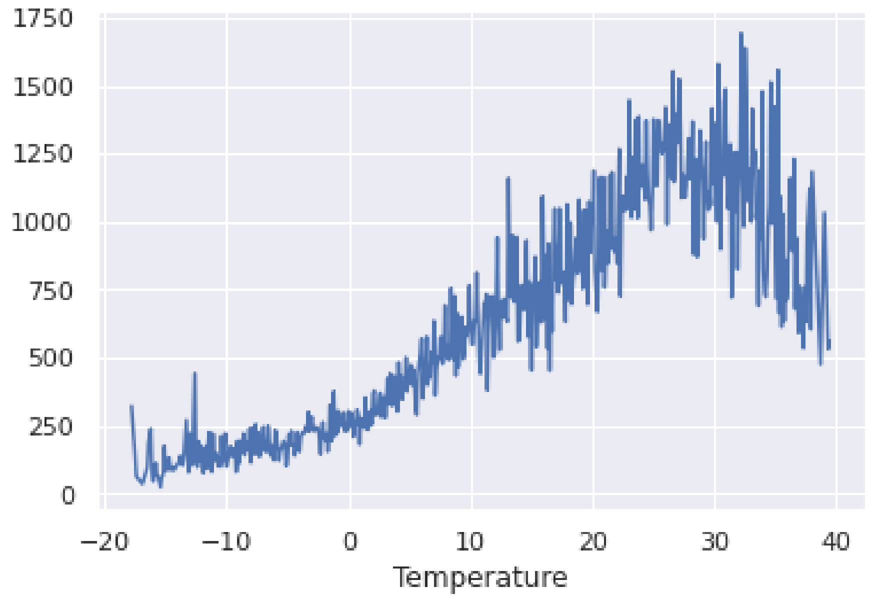
# Exploratory Data Analysis

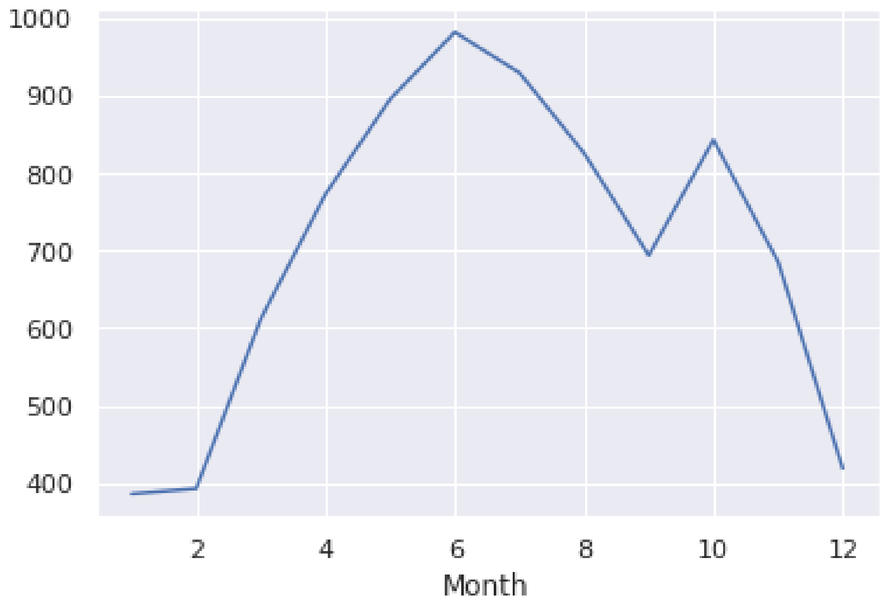


# Exploratory Data Analysis

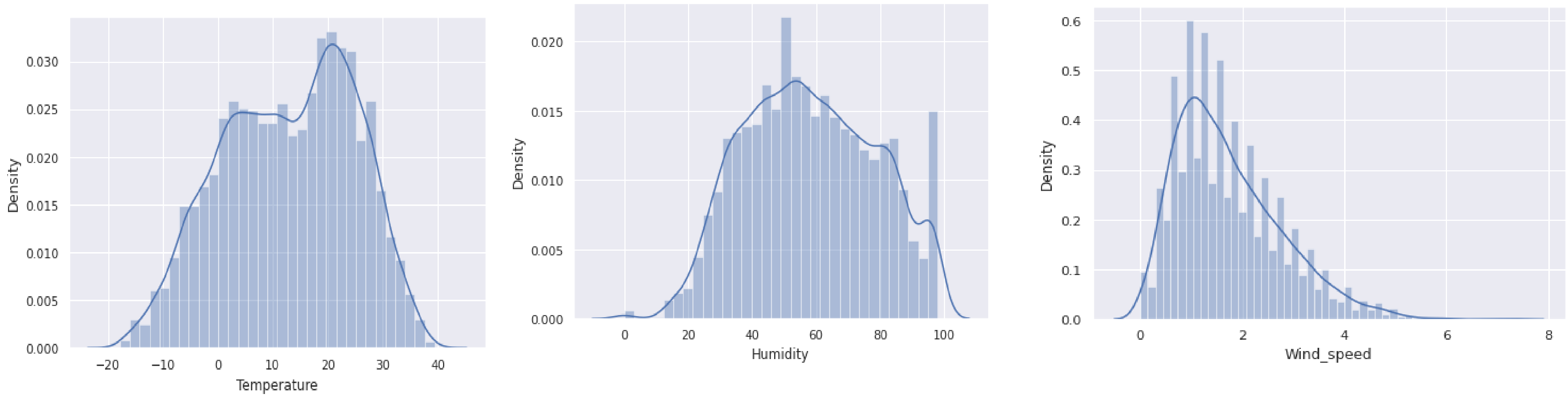


# Exploratory Data Analysis



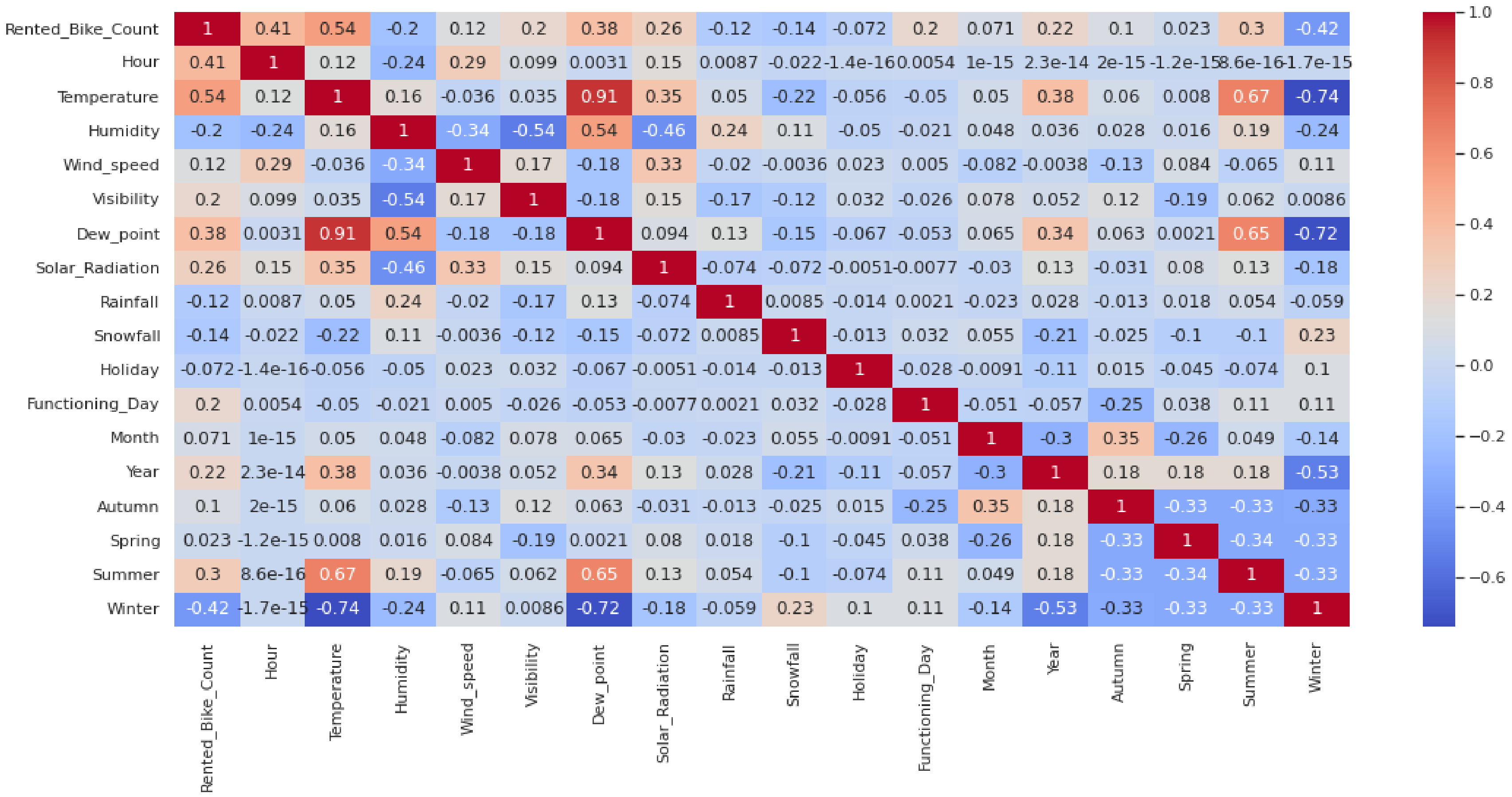
# Exploratory Data Analysis



# Feature Engineering

- Feature Encoding
- Correlation Check
- Outlier Treatment
- Multicollinearity Check
- Linearity Check

# Feature Engineering – Correlation



# Feature Engineering – Correlation

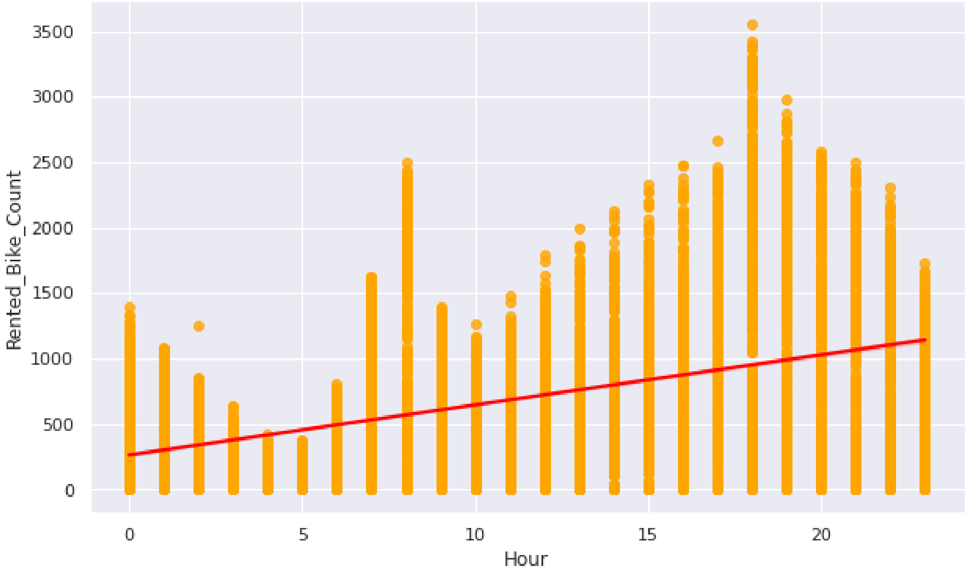
Positive Correlation:
1.Temperature
2.Dew Point Temperature
3.Solar Radiation

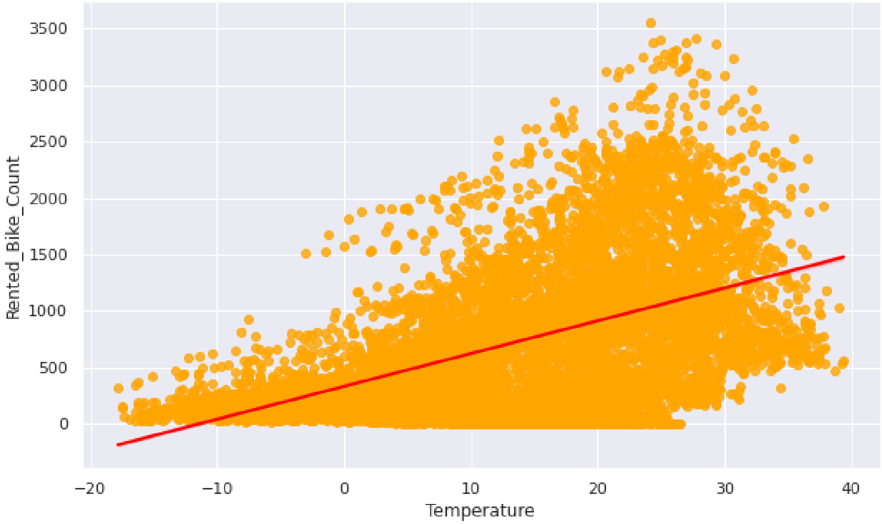
Negative correlation:
1.Winter
2.Humidity
3.Snowfall

The feature Dew Point Temperature is positively and highly correlated with Temperature.

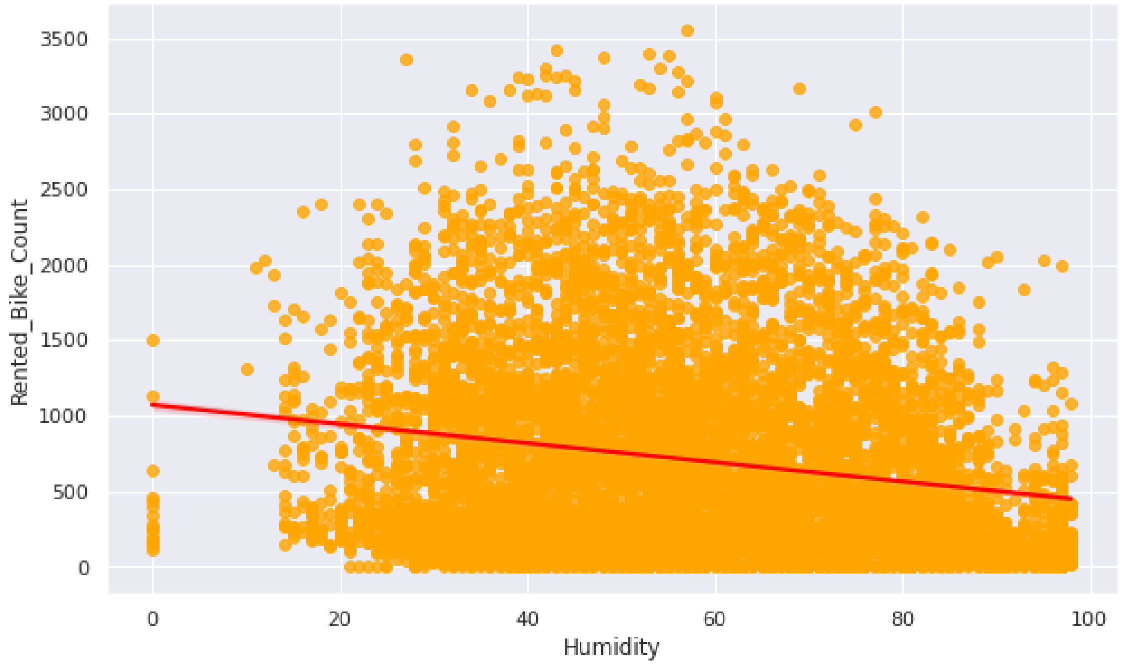
There will be no effect in our model if Dew Point Temp. variable is removed.

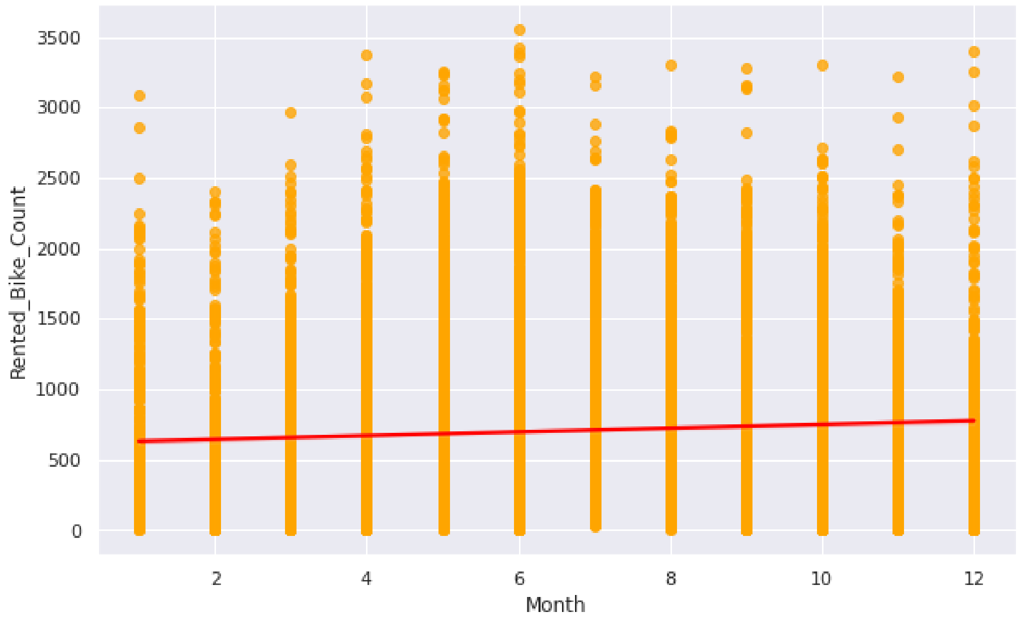
# Linearity Check



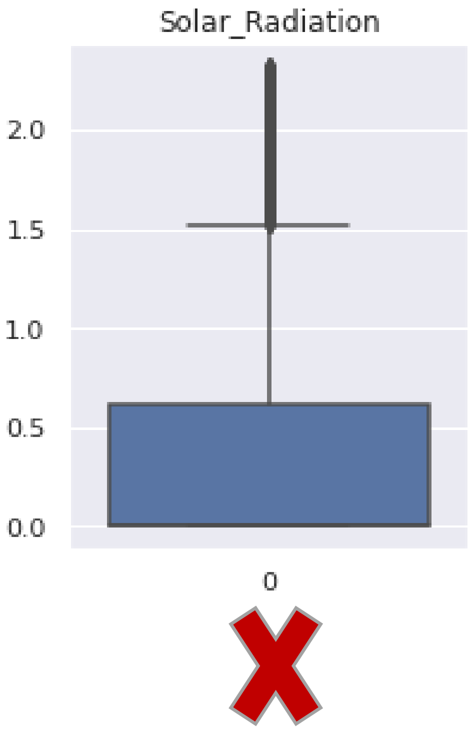



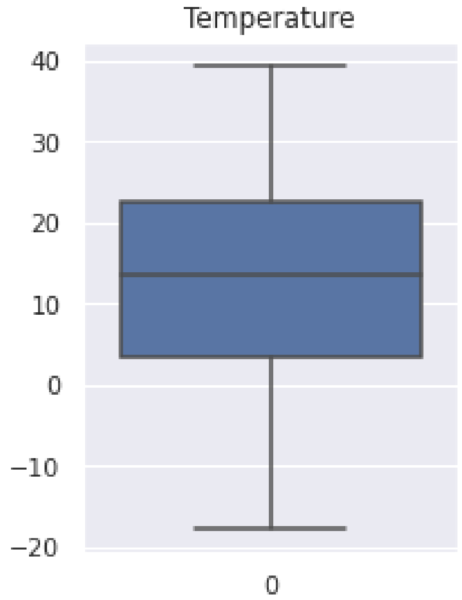
# Linearity Check

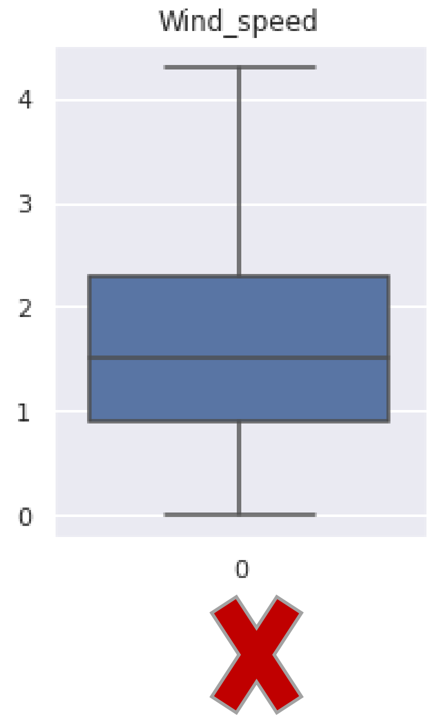


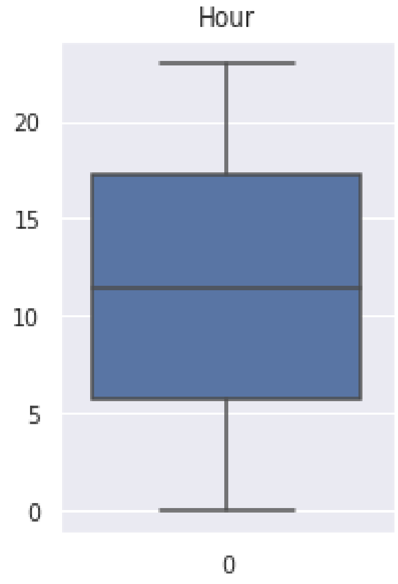



# Pre Processing Data - Outliers



# Outlier Treatment

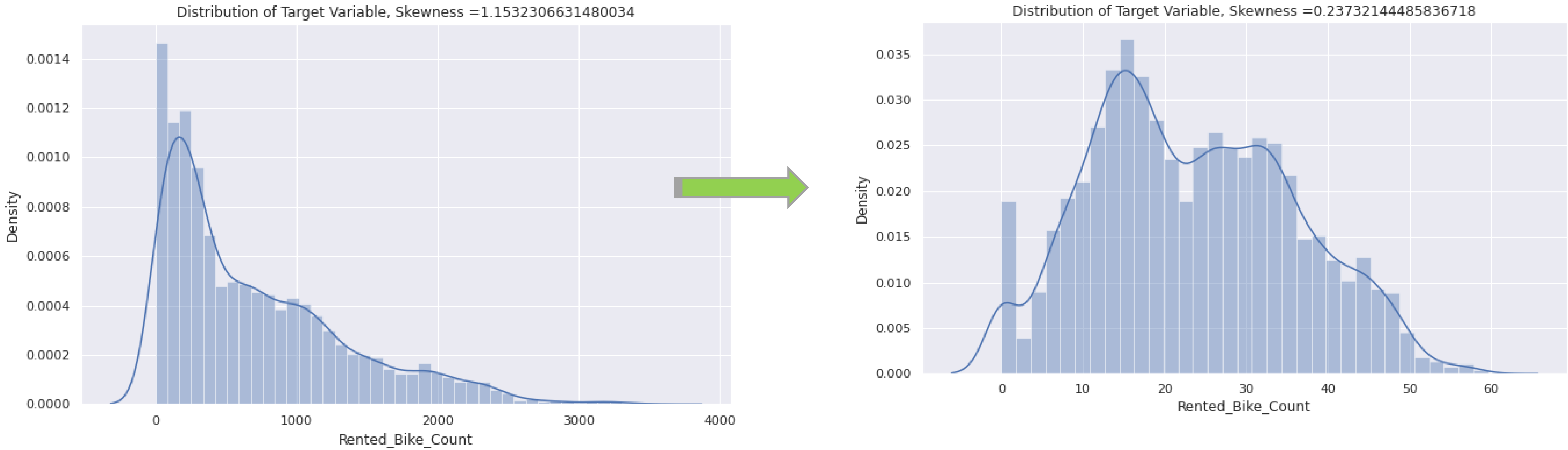
**Q1 – 25%  & Q3 – 75%**

**IQR =Q3-Q1**

**u_lim= Q3+ 1.5*(IQR)**

**l_lim= Q1- 1.5*(IQR)**

# Target Feature Conditioning



# Creating Input and Out Features

**5.1. Creating X and Y Variables**

```
[56] ## x as Independant variable
     ## y as dependant variable

     x=data.drop(['Rented_Bike_Count'], axis=1)
     y=data['Rented_Bike_Count']
```

```
[63] ### Creating Test and Train Set of input and output variables
     x_train,x_test,y_train,y_test= train_test_split(x,y, test_size=0.30, random_state=42)
```

```
[64] x_train.shape, x_test.shape, y_train.shape, y_test.shape

     ((6132, 16), (2628, 16), (6132,), (2628,))
```

# Feature Scaling

```
[65] scaler= StandardScaler()
     x_train= scaler.fit_transform(x_train)
     x_test= scaler.fit_transform(x_test)
```

```
[66] x_train
```

```
array([[-1.09359218, -2.02557075, -0.40155174, ..., -0.5853851 ,
        -0.57986105,  1.76898076],
       [ 1.23323198, -1.58026093, -0.69489404, ..., -0.5853851 ,
        -0.57986105,  1.76898076],
       [-0.51188614, -0.13510455,  0.3806944 , ..., -0.5853851 ,
        -0.57986105, -0.56529727],
       ...,
       [ 0.36067292,  1.38567048,  0.18513286, ..., -0.5853851 ,
         1.72455109, -0.56529727],
       [ 1.23323198, -1.37861121, -0.35266136, ..., -0.5853851 ,
        -0.57986105,  1.76898076],
       [ 1.524085  ,  0.52865914, -0.15709982, ..., -0.5853851 ,
        -0.57986105, -0.56529727]])
```

# Modeling, Evaluating & Tuning

1. Linear Regression
2. Lasso Regression with GridSearchCV
3. Ridge Regression with GridSearchCV
4. Random Forest with GridSearchCV
5. Gradient Boosting Regressor with GridSearchCV

### 1. Linear Regression

Fitting the Model

```
[69] model_1=LinearRegression().fit(x_train,y_train)
```

```
model_1.score(x_train,y_train), model_1.score(x_test,y_test)
```

```
(0.6388208407560204, 0.622310483063715)
```

```
[71] model_1.coef_
```

```
array([ 3.44347399e+00,  5.09673318e+00, -3.14430579e+00,  9.02250273e-02,
        3.29599814e-01,  2.47844904e-01, -4.44089210e-16, -9.99200722e-16,
       -6.46957344e-01,  5.22892019e+00,  8.34607438e-02, -6.76237225e-01,
        1.64634705e+00,  2.41078939e-01,  3.95201814e-01, -2.31191860e+00])
```

```
[72] model_1.intercept_
```

```
23.458551814426663
```

```
[73] ### Predicting  of Target with Training and Testing Data

     y_train_predict = model_1.predict(x_train)
     y_test_predict = model_1.predict(x_test)
```

```
MSE = mean_squared_error(y_train, y_train_predict)
print("MSE :",MSE)

RMSE= np.sqrt(MSE)
print("RMSE :",RMSE)

MAE= mean_absolute_error(y_train, y_train_predict)
print("MAE :",MAE)

R2_Score = r2_score(y_train, y_train_predict)
print("R2_Score :",R2_Score)

Adj_R2_Score = (1-(1-R2_Score)* (x_train.shape[0]-1) /(x_train.shape[0] - x_train.shape[1] - 1))
print("Adj_R2_Score :",Adj_R2_Score)
```

```
MSE : 56.464645536297915
RMSE : 7.51429607723158
MAE : 5.781371705477361
R2_Score : 0.6388143766387069
Adj_R2_Score : 0.6378693284009668
```

```
[92] Dict= {'Model': 'Lasso regression',
            'MSE' : round(MSE,3), 'RMSE': round(RMSE,3),
            'MAE': round(MAE,3), 'R2_Score': round(R2_Score,3),
            'Adj_R2_Score': round(Adj_R2_Score,3)}
```

```
[93] Training_DF= Training_DF.append(Dict,ignore_index=True)
     Training_DF
```

| | Model | MSE | RMSE | MAE | R2_Score | Adj_R2_Score |
|---|---|---|---|---|---|---|
| 0 | Linear regression | 56.464 | 7.514 | 5.783 | 0.639 | 0.638 |
| 1 | Lasso regression | 56.465 | 7.514 | 5.781 | 0.639 | 0.638 |

# Modeling, Evaluating & Tuning

Evaluation Metrics:

- MSE
- MAE
- RMSE
- R2 SCORE
- ADJUSTED R2 SCORE

# Modeling, Evaluating & Tuning

```
[140] Training_DF
```

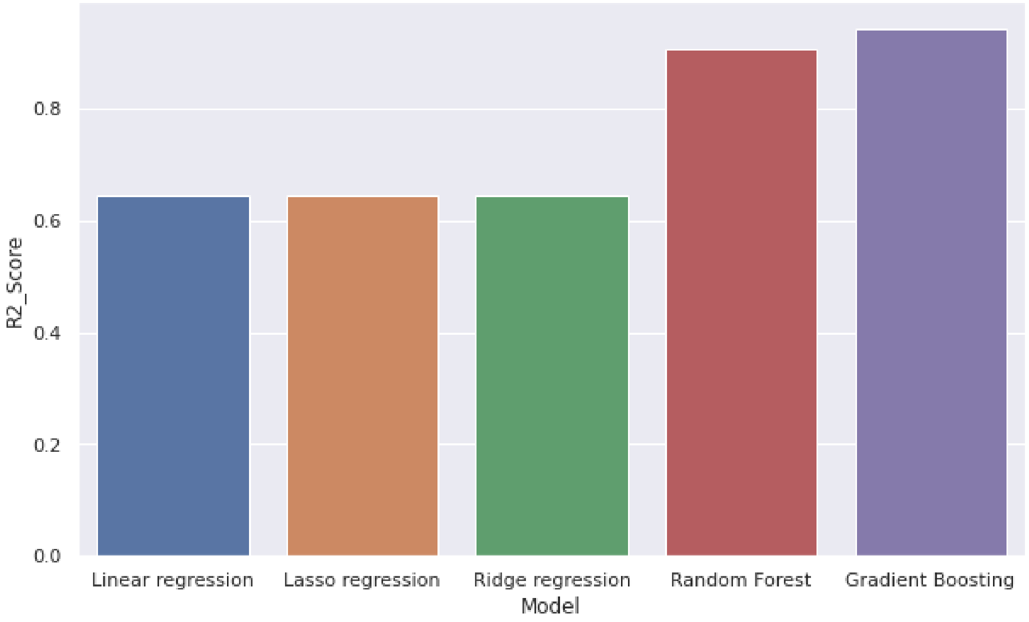
| | Model | MSE | RMSE | MAE | R2_Score | Adj_R2_Score |
|---|---|---|---|---|---|---|
| 0 | Linear regression | 56.464 | 7.514 | 5.783 | 0.639 | 0.638 |
| 1 | Lasso regression | 56.465 | 7.514 | 5.781 | 0.639 | 0.638 |
| 2 | Ridge regression | 56.464 | 7.514 | 5.783 | 0.639 | 0.638 |
| 3 | Random Forest | 14.243 | 3.774 | 2.579 | 0.909 | 0.909 |
| 4 | Gradient Boosting | 8.728 | 2.954 | 2.040 | 0.944 | 0.944 |

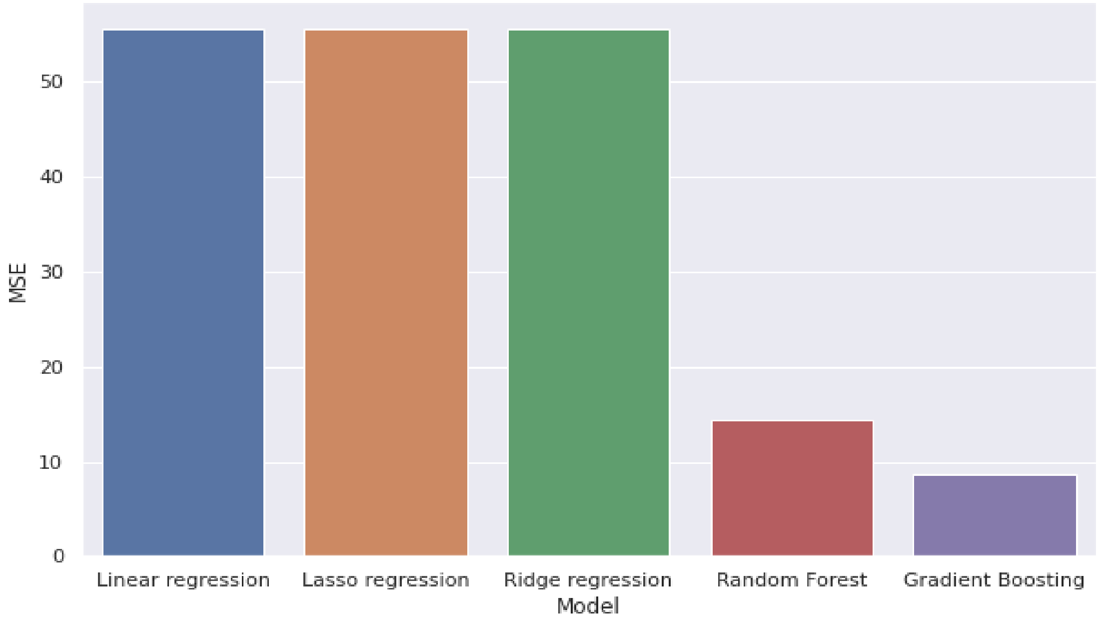
# Modeling, Evaluating & Tuning

```
Testing_DF
```

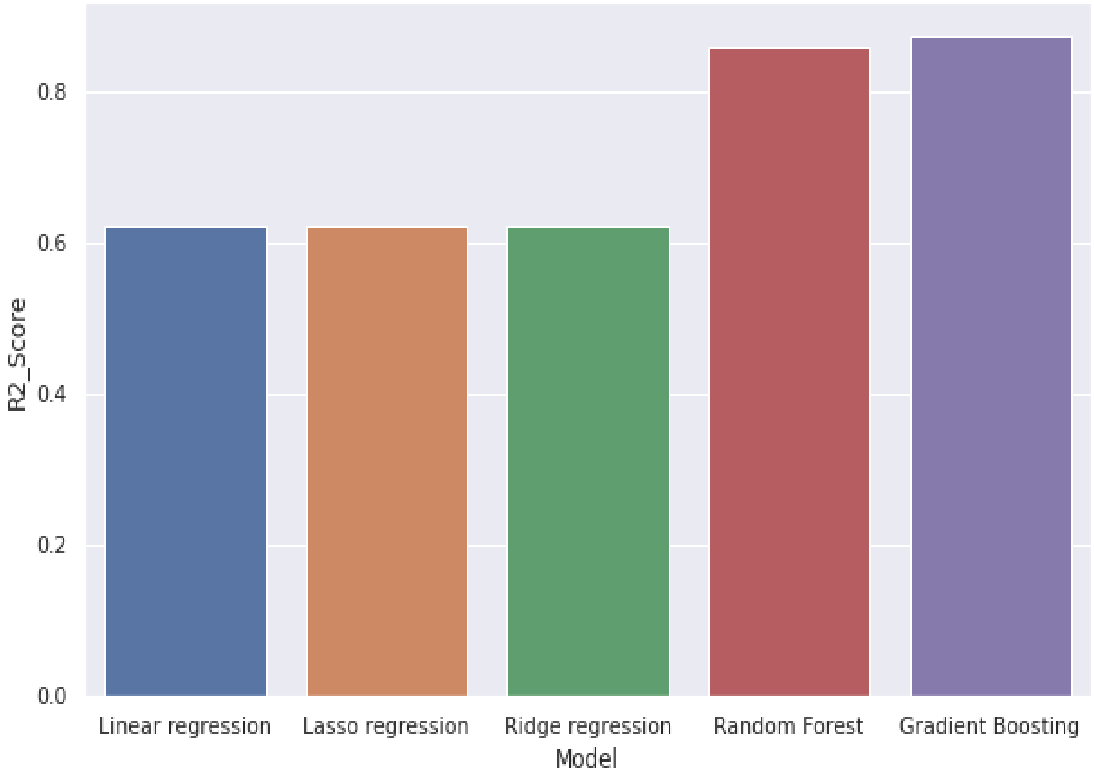
| | Model | MSE | RMSE | MAE | R2_Score | Adj_R2_Score |
|---|---|---|---|---|---|---|
| 0 | Linear regression | 57.380 | 7.575 | 5.823 | 0.622 | 0.620 |
| 1 | Lasso regression | 57.368 | 7.574 | 5.822 | 0.622 | 0.620 |
| 2 | Ridge regression | 57.379 | 7.575 | 5.823 | 0.622 | 0.620 |
| 3 | Random Forest | 21.527 | 4.640 | 3.141 | 0.858 | 0.857 |
| 4 | Gradient Boosting | 19.367 | 4.401 | 3.040 | 0.873 | 0.872 |

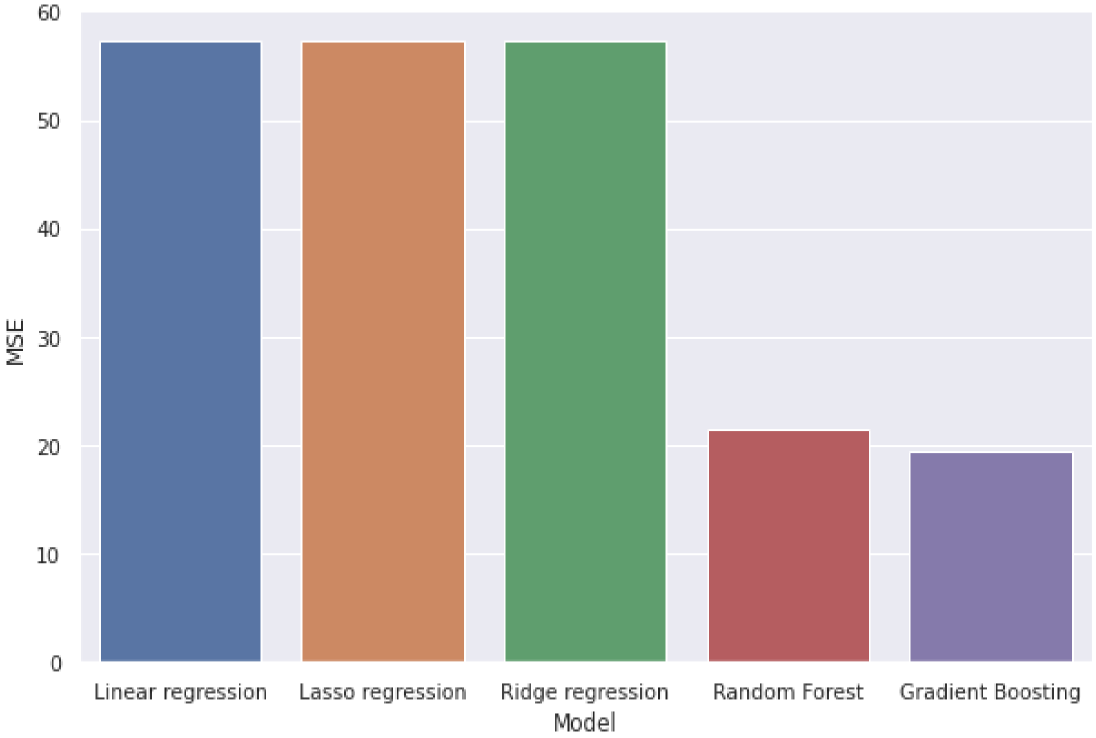
# Comparing Models - Training



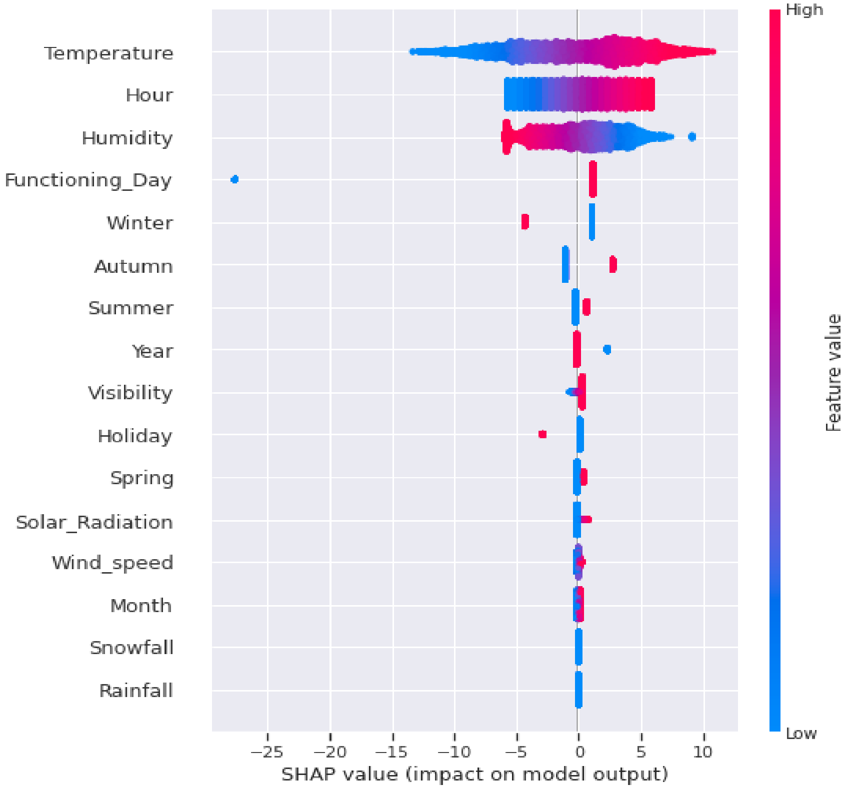
# Comparing Models - Testing



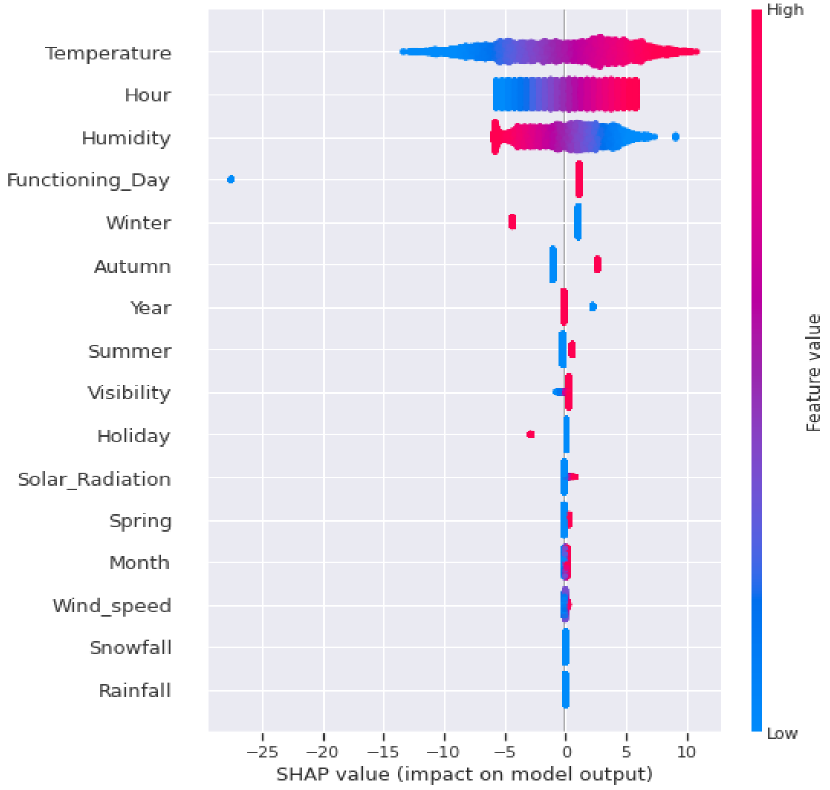



# Model Explainability

Linear Regression

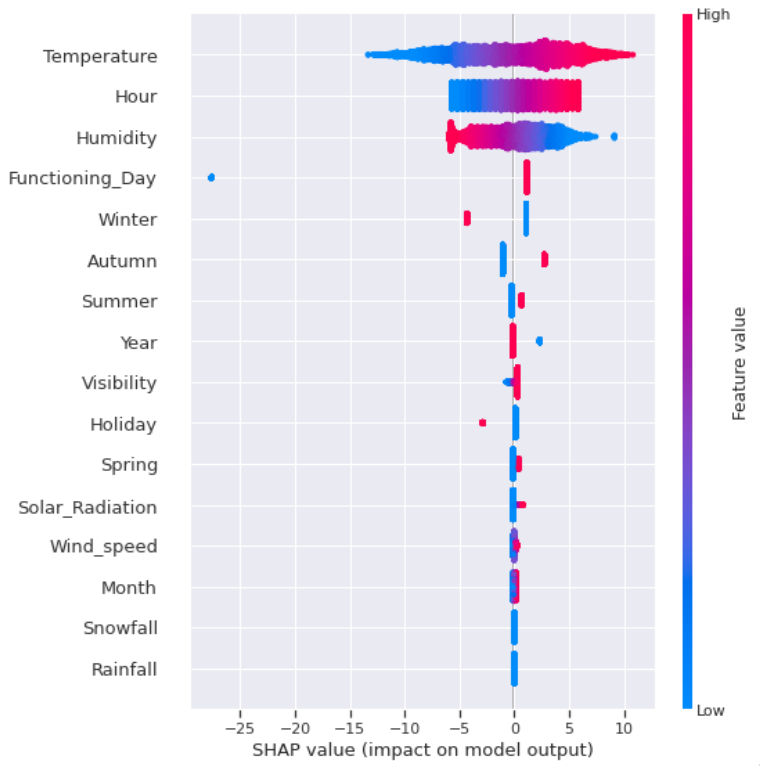



Lasso Regression

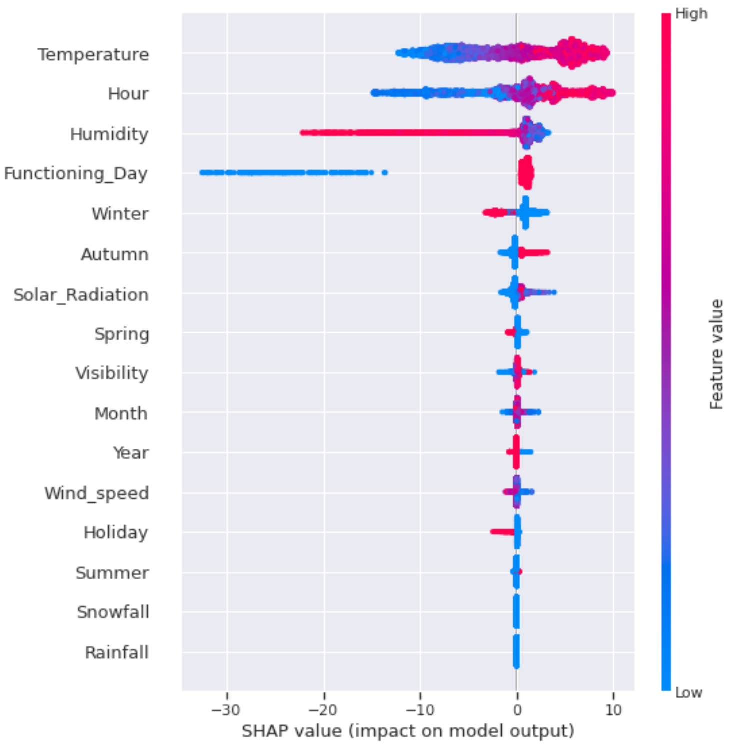


# Model Explainability

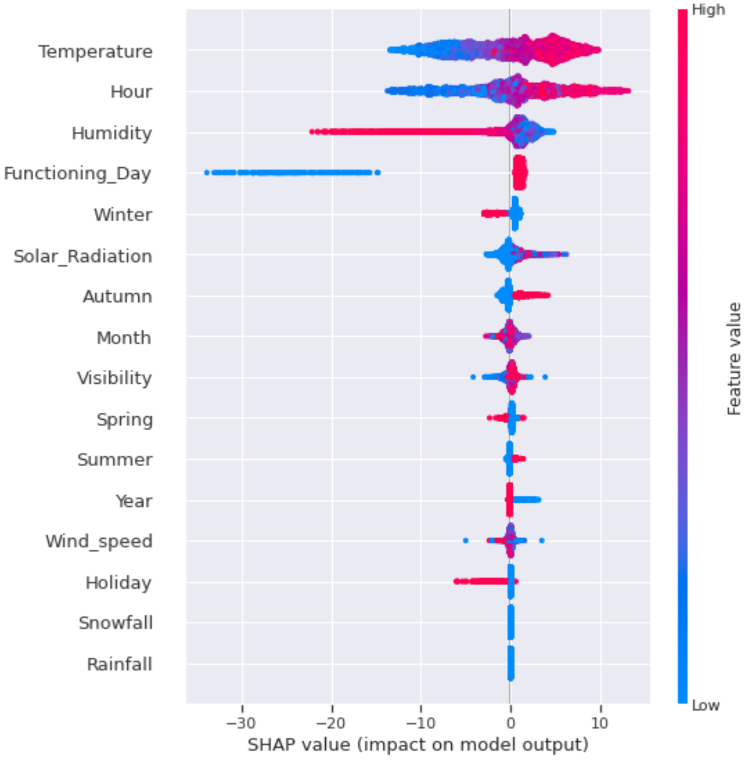
Ridge Regression




Random Forest Regression




Gradient Boost Regression



# Conclusions

1. EDA Outcomes

* The use of bikes is high in between the months of MAY and OCTOBER.
* The bikes are using while reaching and leaving office hours.
* The count of rental bikes is high in summer,  Demand is very low in Winter.
* The use of bikes is high, when the temperature ranges between 25 and 30  degrees.

2. Data Quality Issues

* Outliers available, they are removed.
* Multicollinearity
* Correlation
* Feature Encoding

# Conclusions

3. Model Outcomes

* Among all models, Random Forest and Gradient Boosting are worked better.

* Temperature, Humidity, Hour and Functionalday are the most influencing features on Count of Rental Bikes.

* Linear, Lasso and Ridge Models giving poor results as they are giving r2_score of 0.622, where as the Random forest and Gradient Boosting models giving as 0.858 and 0.873 respectively.

* Ensemble Modeling is preferred as the Mean Squared Error is drastically decreased with Random Forest and Gradient Boosting Regressor.

# Thankyou