Capstone Project

Bike Share Demand Prediction

Chandrashekhar Awate

Contents

Step 1

Exploratory Data Analysis

Data Exploration

Data Analysis

Finding insights

Checking Correlation

Finding multicollinearity

Removing multicollinearity by VIF

Step 2

Modelling And Evaluation

Linear Regression

Polynomial Regression

Random Forest Regressor

Regularization for regression

Hyperparameter Tuning

Introduction

- 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.
- The dataset contains weather information (Temperature, Humidity, Windspeed, Visibility, Dewpoint, Solar radiation, Snowfall, Rainfall), the number of bikes rented per hour and date information.

Information About Dataset

- Date : year-month-day
- Rented Bike count - Count of bikes rented at each hour
- Hour - Hour of he day
- Temperature-Temperature in Celsius
- Humidity - %
- Windspeed - m/s
- Visibility - 10m
- Dew point temperature - Celsius
- Solar radiation - MJ/m2
- Rainfall - mm
- Snowfall - cm
- Seasons - Winter, Spring, Summer, Autumn
- Holiday - Holiday/No holiday
- Functional Day - NoFunc(Non Functional Hours), Fun(Functional hours)

Problem Statement

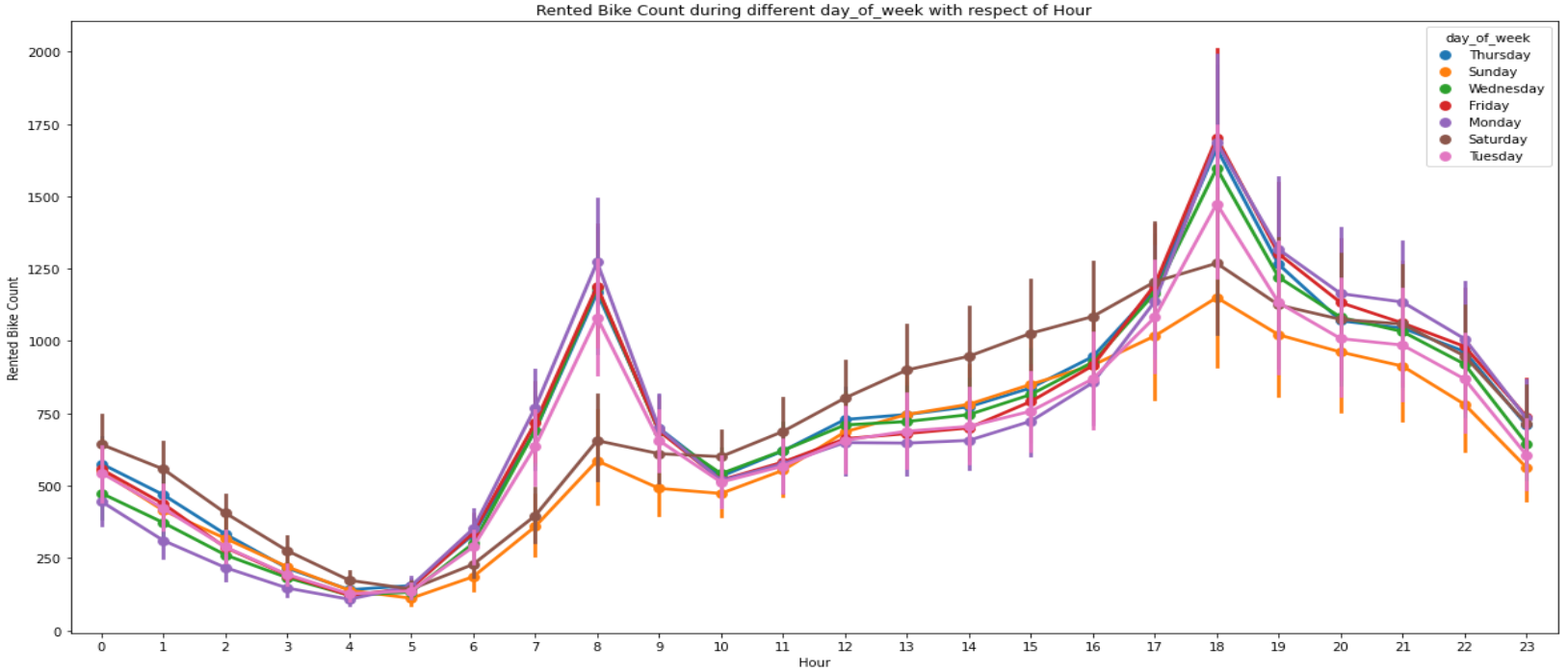
Our Goal is prediction of bike count required at each hour for the stable supply of rental bikes.

Exploratory Data Analysis

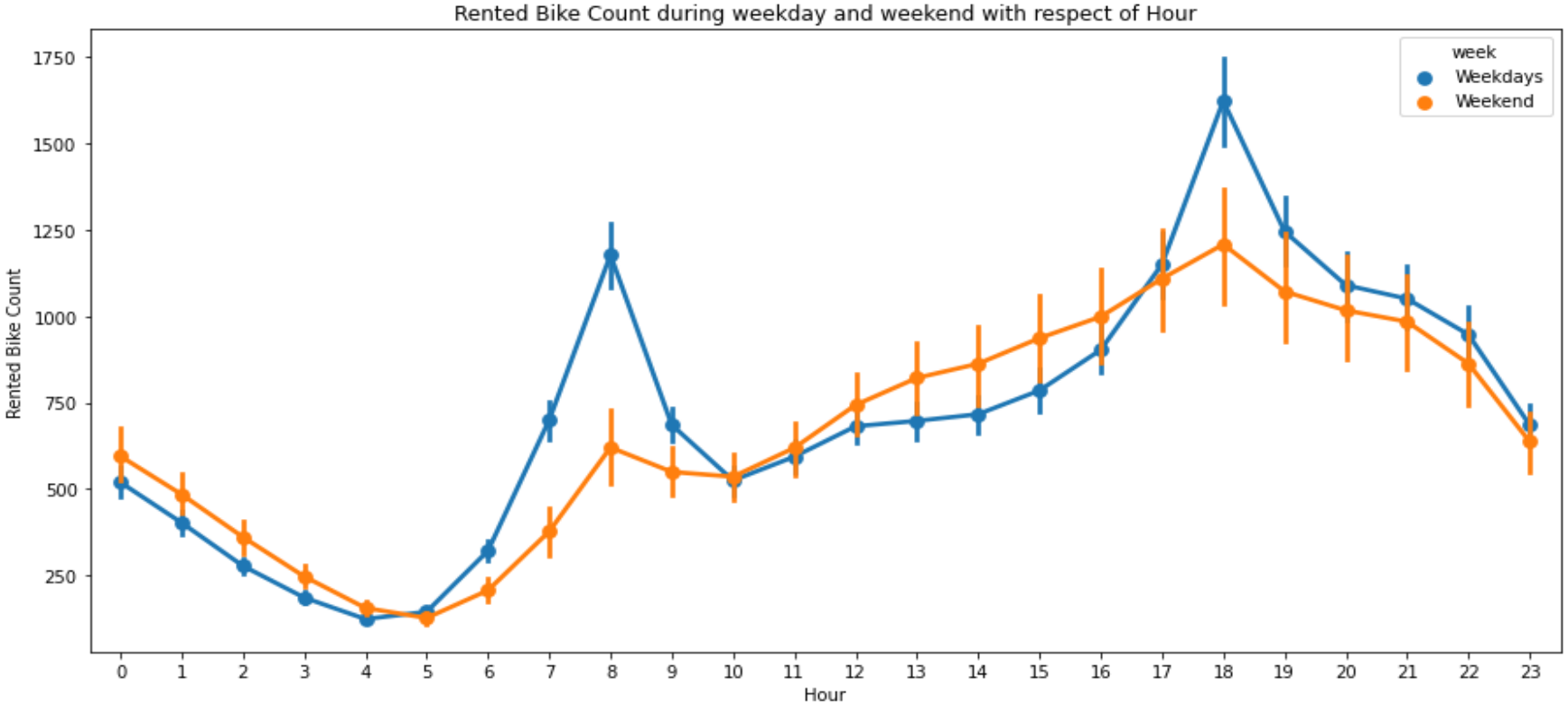
```
#lets see top 5 records
df.head(5)
```

	Date	Rented Bike Count	Hour	Temperature(°C)	Humidity(%)	Wind speed (m/s)	Visibility (10m)	Dew point temperature(°C)	Solar Radiation (MJ/m2)	Rainfall(mm)	Snowfall (cm)	Seasons	Holiday	Functioning Day
0	01/12/2017	254	0	-5.2	37	2.2	2000	-17.6	0.0	0.0	0.0	Winter	No Holiday	Yes
1	01/12/2017	204	1	-5.5	38	0.8	2000	-17.6	0.0	0.0	0.0	Winter	No Holiday	Yes
2	01/12/2017	173	2	-6.0	39	1.0	2000	-17.7	0.0	0.0	0.0	Winter	No Holiday	Yes
3	01/12/2017	107	3	-6.2	40	0.9	2000	-17.6	0.0	0.0	0.0	Winter	No Holiday	Yes
4	01/12/2017	78	4	-6.0	36	2.3	2000	-18.6	0.0	0.0	0.0	Winter	No Holiday	Yes

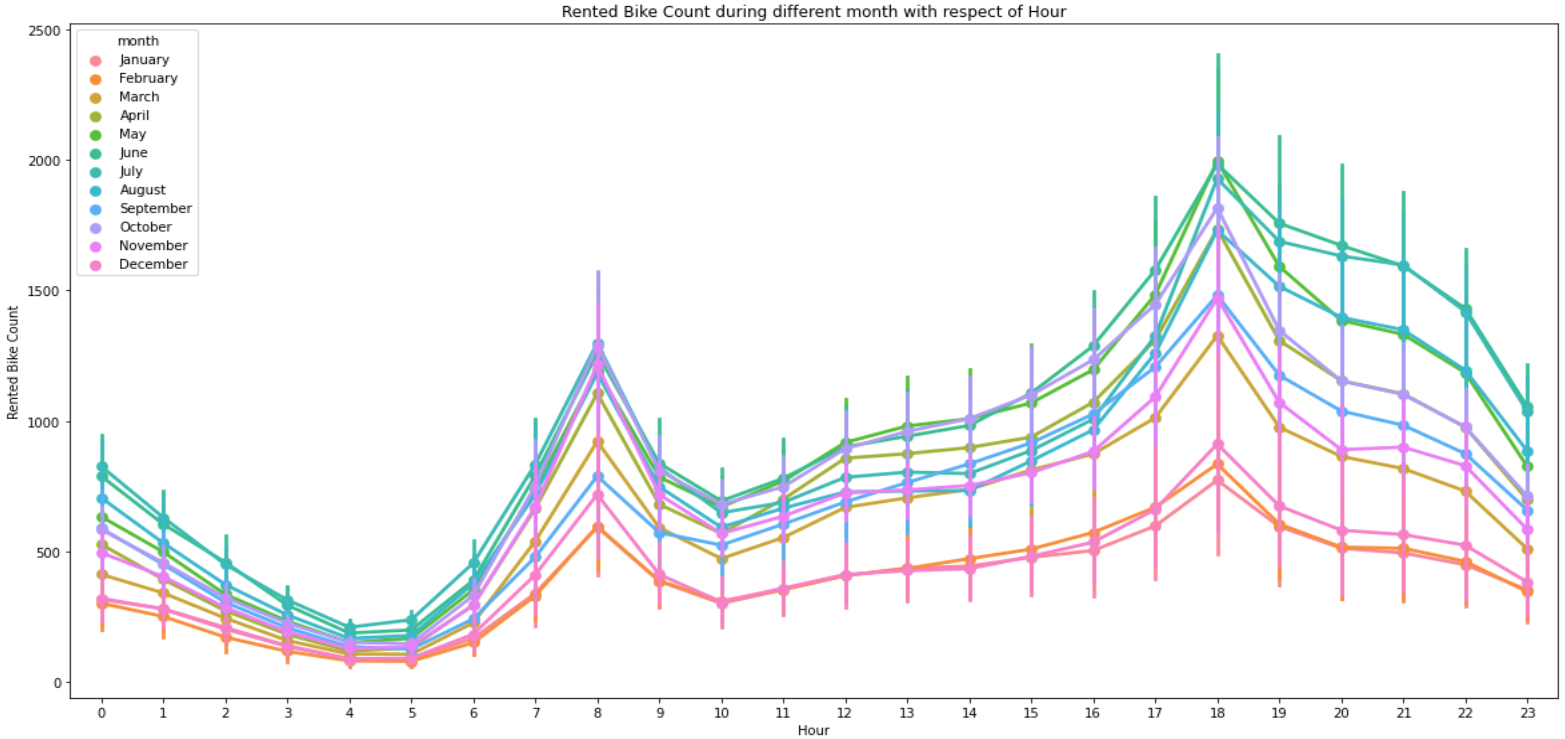
Exploratory Data Analysis



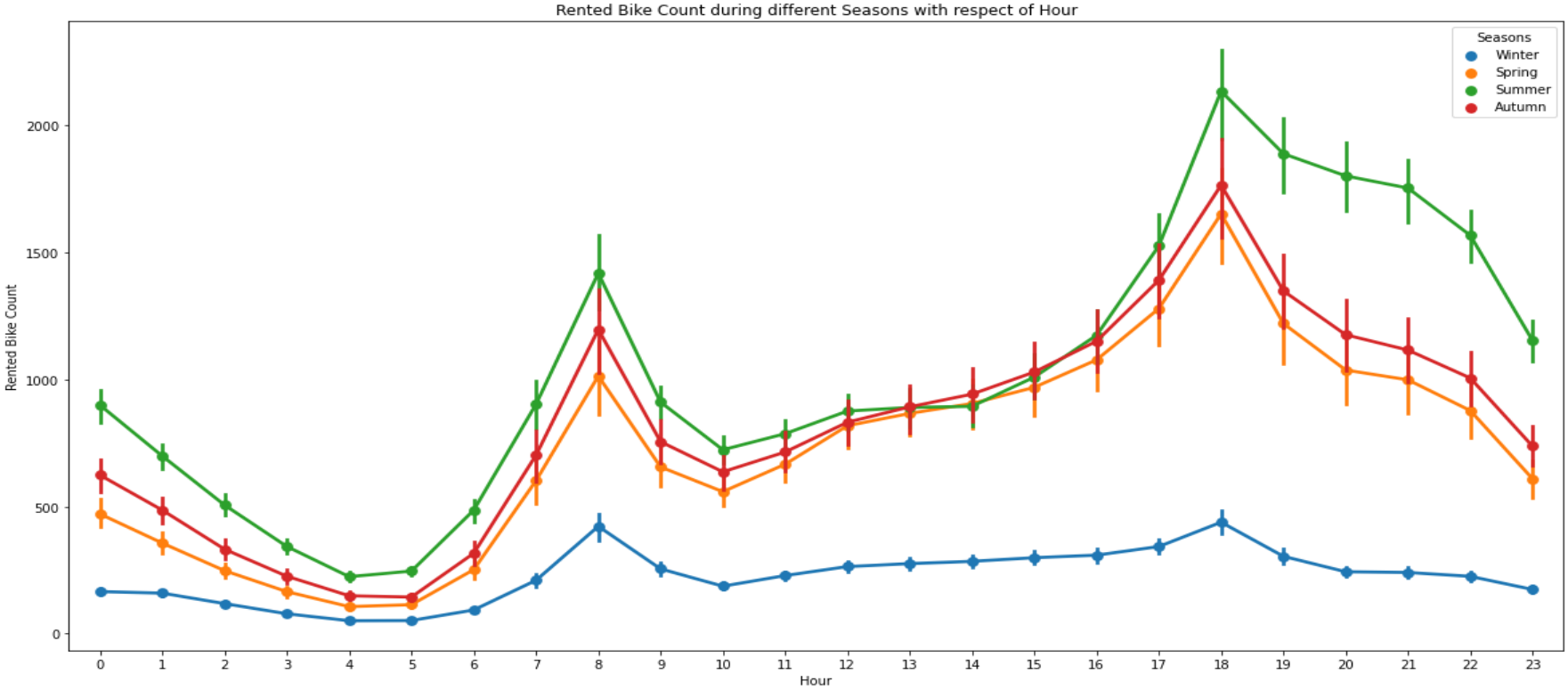
Exploratory Data Analysis



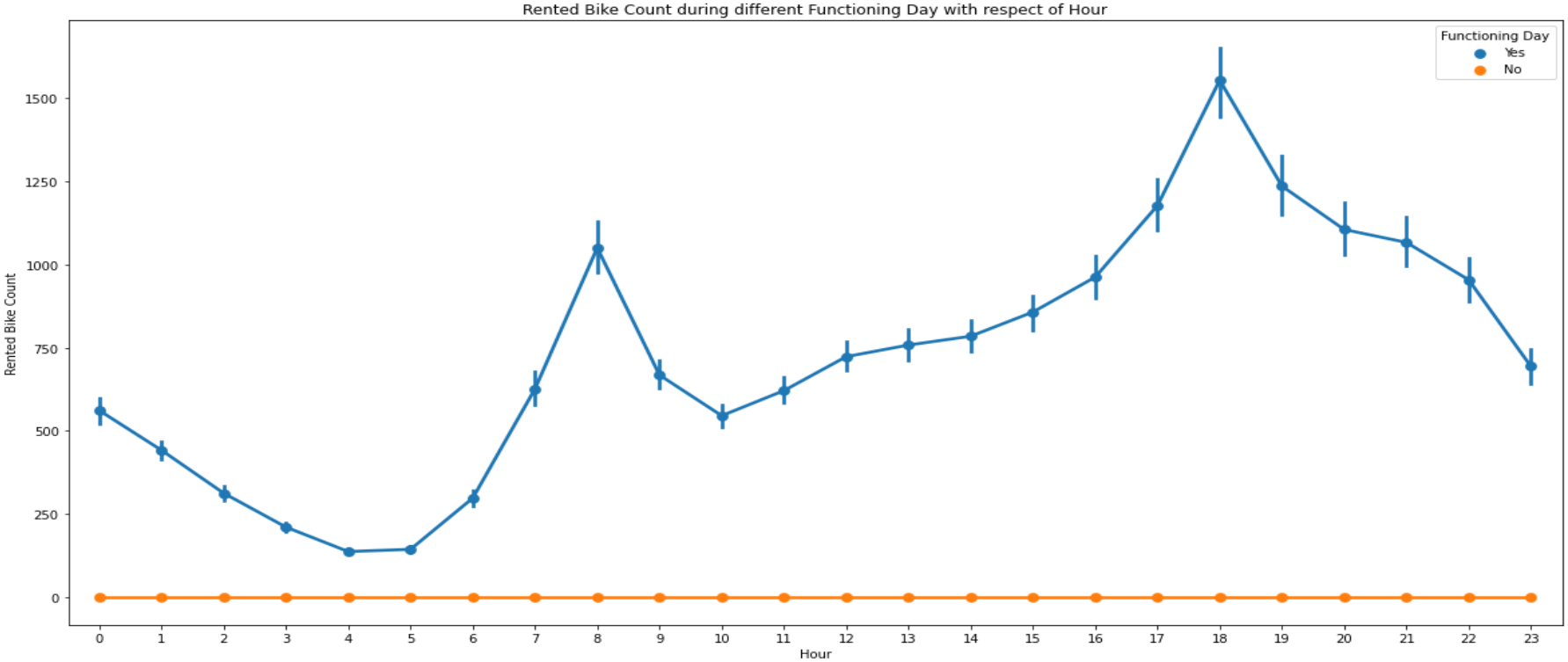
Exploratory Data Analysis



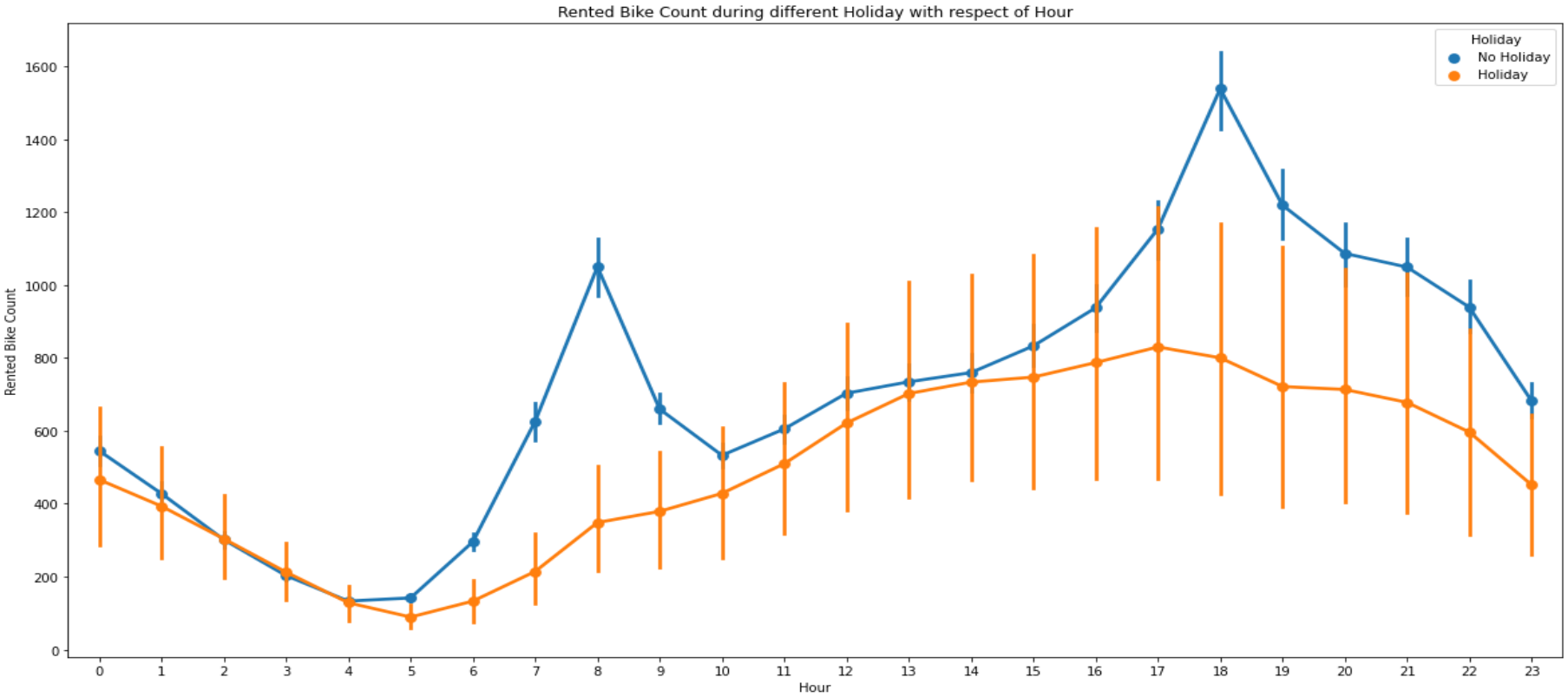
Exploratory Data Analysis



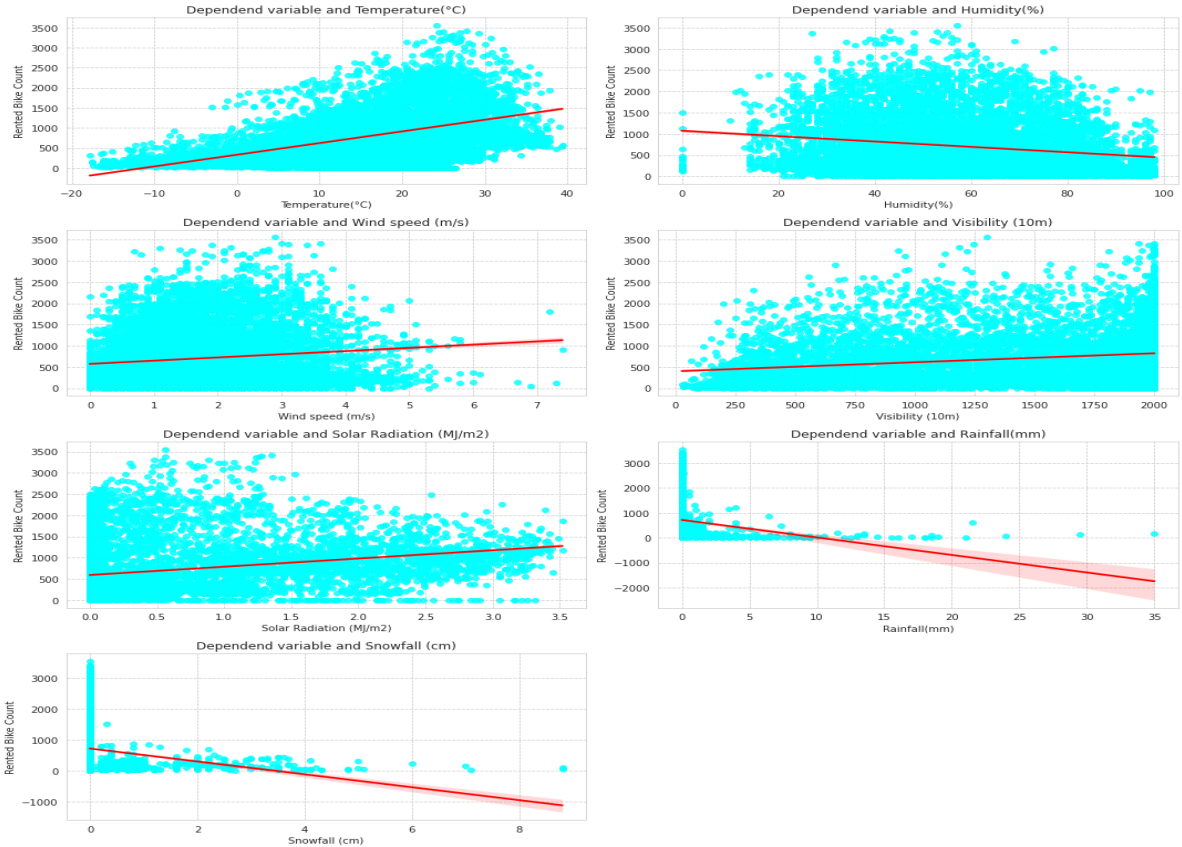
Exploratory Data Analysis



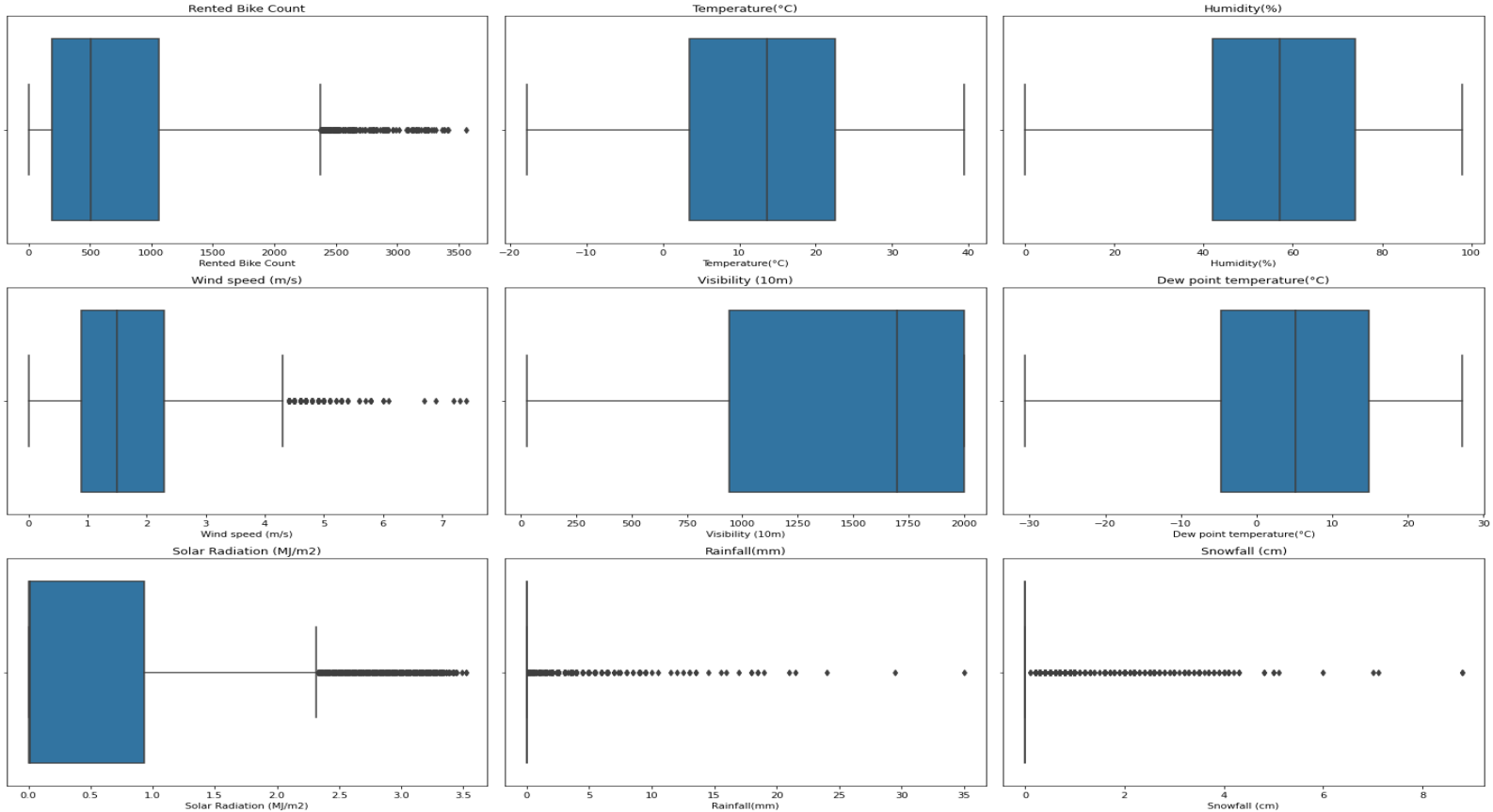
Exploratory Data Analysis



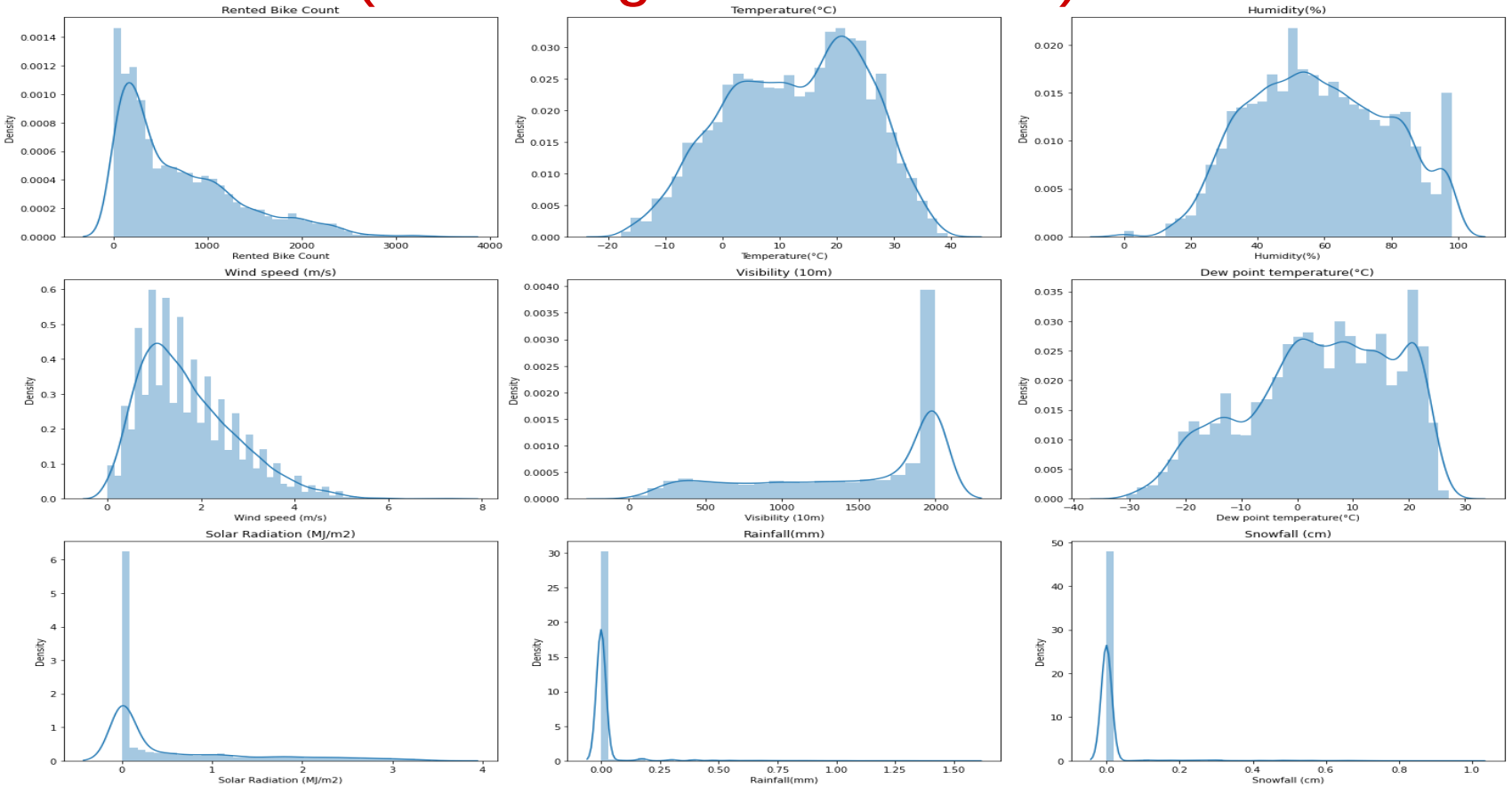
Exploratory Data Analysis



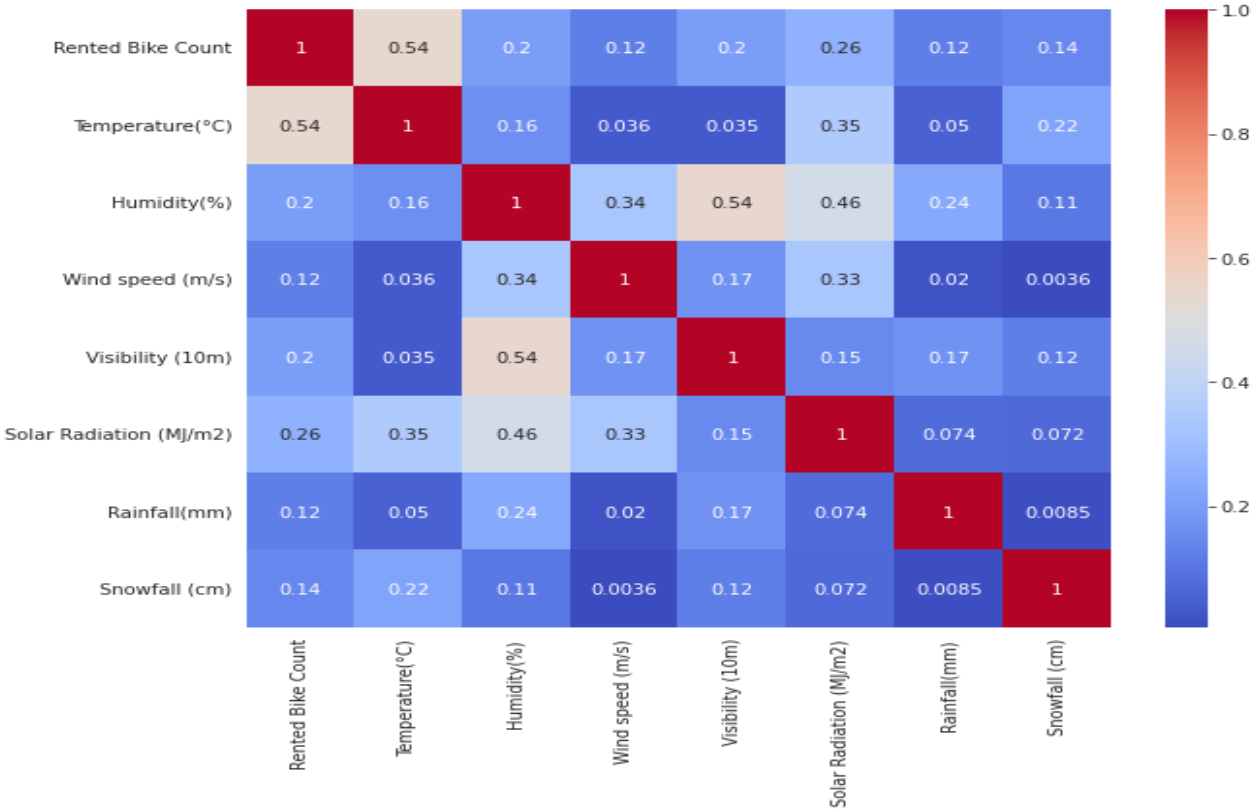
Outlier detections and there treatment



Distribution (Checking for skewness)



Correlation



correlation

```
# Correlation with Rented Bike Count
df.corr()['Rented Bike Count']
```

```
Rented Bike Count            1.000000
Temperature(°C)              0.538558
Humidity(%)                 -0.199780
Wind speed (m/s)             0.121108
Visibility (10m)             0.199280
Dew point temperature(°C)    0.379788
Solar Radiation (MJ/m2)      0.261837
Rainfall(mm)                -0.180882
Snowfall (cm)               -0.163479
Name: Rented Bike Count, dtype: float64
```

Modelling And Evaluation

1)Linear Regression

Linear regression is a linear model, e.g. a model that assumes a linear relationship between the input variables (x) and the single output variable (y). More specifically, that y can be calculated from a linear combination of the input variables (x).

Different techniques can be used to prepare or train the linear regression equation from data, the most common of which is called Ordinary Least Squares. It is common to therefore refer to a model prepared this way as Ordinary Least Squares Linear Regression or just Least Squares Regression.

```
# Fitting the model
regression = LinearRegression()
regression.fit(X_train,y_train)
```

```
LinearRegression()
```

```
regression.score(X_train,y_train)
```

```
0.7945303123108239
```

```
#predicting
reg_pred = regression.predict(X_test)
```

```
#Sores of Linear Regression model
score_metrics(y_test,reg_pred)
```

```
mean absolute error is   : 207.53386831963152
mean squared error is   : 93982.68863138056
Root mean squared error is   : 306.56596130585103
R2 score is   : 0.7703263372002714
```

Modelling and Evaluation

2) **Polynomial Regression**

Polynomial regression is a special case of linear regression where we fit a polynomial equation on the data with a curvilinear relationship between the target variable and the independent variables.

```
from sklearn.preprocessing import PolynomialFeatures
```

```
poly = PolynomialFeatures()
poly_X_train = poly.fit_transform(X_train)
poly_X_test = poly.transform(X_test)
```

Modelling and Evaluation

3)Random Forest

Random forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both regression and classification problems.

It is based on the concept of ensemble learning, which is process of combining multiple classifier to solve complex problem and improve the model performance.

```
random = RandomForestRegressor(n_estimators=100,random_state=0)
random.fit(X_train,y_train)
```

```
RandomForestRegressor(random_state=0)
```

Hyperparameter tuning and regularization

```
lasso = Lasso()
parameters = {'alpha': [1e-15,1e-13,1e-10,1e-8,1e-5,1e-4,1e-3,1e-2,1e-1,1,5,10,20,30,40,45,50,55,60,100,0.0014]}
lasso_regressor = GridSearchCV(lasso, parameters, scoring='neg_mean_squared_error', cv=5)
lasso_regressor.fit(X_train, y_train)
```

```
GridSearchCV(cv=5, estimator=Lasso(),
             param_grid={'alpha': [1e-15, 1e-13, 1e-10, 1e-08, 1e-05, 0.0001,
                                   0.001, 0.01, 0.1, 1, 5, 10, 20, 30, 40, 45,
                                   50, 55, 60, 100, 0.0014]},
             scoring='neg_mean_squared_error')
```

```
print("The best fit alpha value is found out to be :" ,lasso_regressor.best_params_)
print("\nUsing ",lasso_regressor.best_params_, " the negative mean squared error is: ", lasso_regressor.best_score_)
```

```
The best fit alpha value is found out to be : {'alpha': 0.0014}

Using  {'alpha': 0.0014}  the negative mean squared error is:  -32.67848428711776
```

```
[ ]  L2 = Ridge()
     parameters = {'alpha': [1e-15,1e-10,1e-8,1e-5,1e-4,1e-3,1e-2,1,5,10,20,30,40,45,50,55,60,100,0.5,1.5,1.6,1.7,1.8,1.9]}
     L2_regressor = GridSearchCV(L2, parameters, scoring='neg_mean_squared_error', cv=5)
     L2_regressor.fit(X_train,y_train)
```

```
     GridSearchCV(cv=5, estimator=Ridge(),
                  param_grid={'alpha': [1e-15, 1e-10, 1e-08, 1e-05, 0.0001, 0.001,
                                        0.01, 1, 5, 10, 20, 30, 40, 45, 50, 55, 60,
                                        100, 0.5, 1.5, 1.6, 1.7, 1.8, 1.9]},
                  scoring='neg_mean_squared_error')
```

```
[ ]  print("The best fit alpha value is found out to be :" ,L2_regressor.best_params_)
     print("\nUsing ",L2_regressor.best_params_, " the negative mean squared error is: ", L2_regressor.best_score_)
```

```
     The best fit alpha value is found out to be : {'alpha': 1.9}

     Using  {'alpha': 1.9}  the negative mean squared error is:  -32.680686843688136
```

Hyperparameter Tuning and Regularization

```
[ ]  random2 = RandomForestRegressor(n_estimators = 100,random_state = 0,criterion = 'absolute_error',max_features = "sqrt",max_depth = 15)
```

```
[ ]  random2.fit(X_train,y_train)
```

```
RandomForestRegressor(criterion='absolute_error', max_depth=15,
                      max_features='sqrt', random_state=0)
```

Evaluation

- 1) Mean Absolute Error(MAE) MAE is a very simple metric which calculates the absolute difference between actual and predicted values. To better understand, let’s take an example you have input data and output data and use Linear Regression, which draws a best-fit line. Now you have to find the MAE of your model which is basically a mistake made by the model known as an error. Now find the difference between the actual value and predicted value that is an absolute error but we have to find the mean absolute of the complete dataset. so, sum all the errors and divide them by a total number of observations And this is MAE. And we aim to get a minimum MAE because this is a loss.

- 2) Mean Squared Error(MSE) MSE is a most used and very simple metric with a little bit of change in mean absolute error. Mean squared error states that finding the squared difference between actual and predicted value.
- It represents the squared distance between actual and predicted values. we perform squared to avoid the cancellation of negative terms and it is the benefit of MSE.

Evaluation

- 3) Root Mean Squared Error(RMSE) As RMSE is clear by the name itself, that it is a simple square root of mean squared error.

- 4) R Squared (R2) R2 score is a metric that tells the performance of your model, not the loss in an absolute sense that how many wells did your model perform. In contrast, MAE and MSE depend on the context as we have seen whereas the R2 score is independent of context. So, with help of R squared we have a baseline model to compare a model which none of the other metrics provides. The same we have in classification problems which we call a threshold which is fixed at 0.5. So basically R2 squared calculates how must regression line is better than a mean line. Hence, R2 squared is also known as Coefficient of Determination or sometimes also known as Goodness of fit

Evaluation

```
#Sores of Linear Regression model
score_metrics(y_test,reg_pred)
```

```
mean absolute error is  : 207.41666780312968
mean squared error is  : 93868.92031647604
Root mean squared error is  : 306.3803523669167
R2 score is  : 0.7706043627172595
```

```
#Scores of ridge regression
L2_pred = L2_regressor.predict(X_test)
score_metrics(y_test,L2_pred)
```

```
mean absolute error is  : 207.43841458794412
mean squared error is  : 93898.28666463892
Root mean squared error is  : 306.4282732788196
R2 score is  : 0.7705325976204759
```

```
#Scores of lasso regression
L1_pred = L1.predict(X_test)
score_metrics(y_test,L1_pred)
```

```
mean absolute error is  : 207.4637026586552
mean squared error is  : 93937.02946963176
Root mean squared error is  : 306.49148351892546
R2 score is  : 0.770437918461373
```

```
#scores of polynomial regression
score_metrics(y_test,poly_pred)
```

```
mean absolute error is  : 132.8128809774486
mean squared error is  : 47669.689042533406
Root mean squared error is  : 218.33389348091012
R2 score is  : 0.8835054386466332
```

```
#scores of random forest regressor
score_metrics(y_test,y_random_test)
```

```
mean absolute error is  : 180.31377109194364
mean squared error is  : 84119.98191378884
Root mean squared error is  : 290.0344495293427
R2 score is  : 0.7944286906223295
```

Conclusion

1)We have used three models namely Linear Regression, Polynomial Regression and Random Forest Regressor.
2)We also used hyperparameter tuning and regularization technique to find optimal model.But the results were more or less same as of linear Regression.
3)The polynomial regression model tops in terms of R- squred value which was 88%and have least mean absolute error.
4) we are able to understand that the demand is low in the winter season.
5) The demand is low during holidays, but in no holidays the demand is high, it may be because people use bikes to go to their work
6) If there is no Functioning Day then there is no demand
7) We can observe from this column that the pattern of weekdays and weekends is different, in the weekend the demand becomes high in the afternoon. While the demand for office timings is high during weekdays, we can further change this column to weekdays and weekends.
8) We can clearly see that the demand is low in December January & February, It is cold in these months and we have already seen in season column that demand is less in winters.

Refferences

1)Analytics Vidhya

2)Machine Learning Mastery Etc.

Thankyou