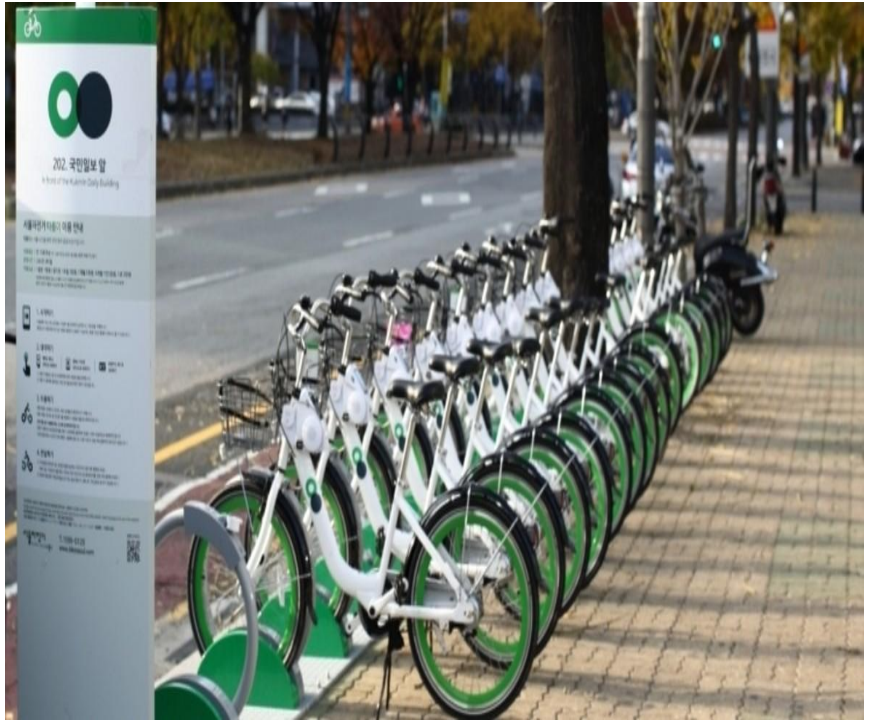
Capstone Project

Regression Analysis on Seoul Bike Sharing Demand Prediction

Rohan Jagadale
Data science trainee at Almabetter

Flow of the Presentation



- Introduction
- Problem Statement
- Exploratory Data Analysis
- Data Preparation
- Feature Selection
- Model Implementation
- Evaluation of model
- conclusion

Introduction

Bike sharing Demand

- Bike sharing systems are a means to renting bicycles where the process of obtaining membership, rental and bike return Is automated via a network of kiosk location throughout a city.

Methodology

Supervised machine learning (Regression)

Database

- Seoul Bike Data
- From 2016 to 2017
- 8761 rows and 14 columns

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 problem here is to make Prediction of bike count required at each hour for the stable supply of rental bikes.

Data Pipeline

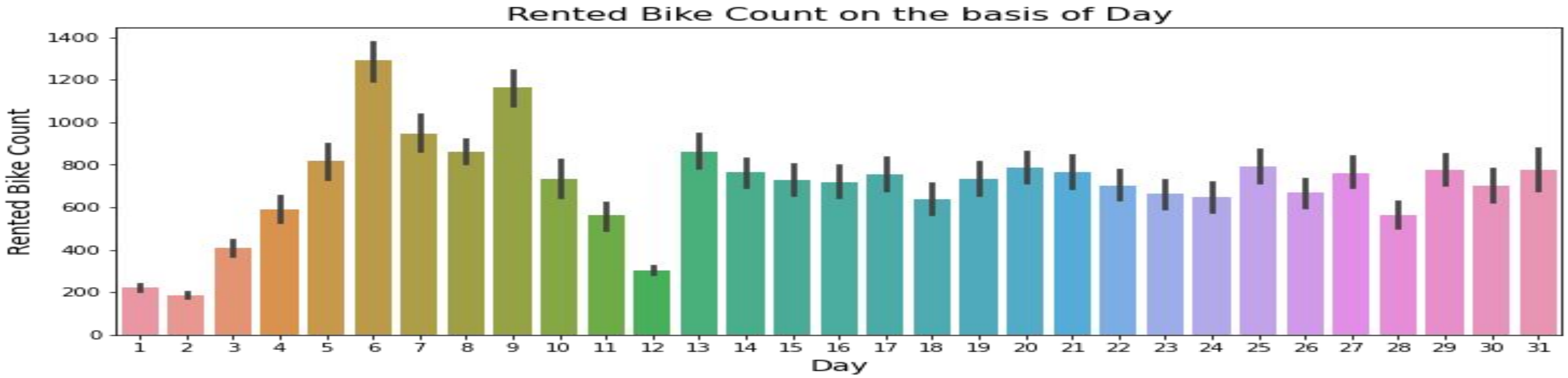
- ❖ Data processing-: At first phase checked for null values and changed the datetime containing column in dataset.
- ❖ Separate Features : Separate dependent and Independent variables. Divide them in different columns.
- ❖ EDA: Exploratory Data analysis was done on the features selected in the phase
- ❖ Data processing-2: Preparing the new dataframe with selected columns And dropped dummy variables.
- ❖ Feature Selection: Used VIF to check the correlation among the variables.
- ❖ Create a model: Finally in this part created models, trained and tested it on the available dataset.

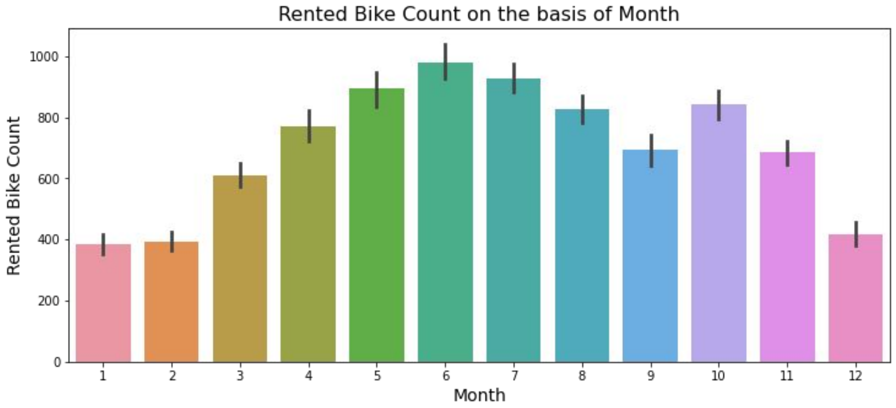
Data Description

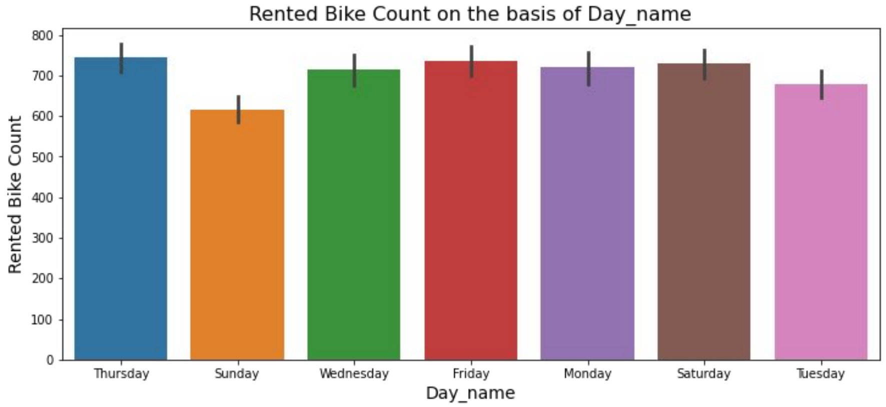
Attribute Information:

- Date : year-month-day
- Rented Bike count - Count of bikes rented at each hour
- Hour - Hour of he day
- Temperature-Temperature in Celsius. (%)
- Humidity - a quantity representing the amount of water vapour in the atmosphere or in a gas.
- Wind speed - The rate at which air is moving in a particular area
- Visibility - measure of the distance at which an object or light can be clearly discerned (10m)
- Dew point temperature - The temperature below which the water vapour in a volume of air at a constant pressure will condense into liquid water
- Solar radiation - The electromagnetic radiation emitted by the sun (MJ/m2)
- Rainfall - the quantity of rain falling within a given area in a given time (mm)
- Snowfall - the quantity of snow falling within a given area in a given time (cm)
- Seasons - Winter, Spring, Summer, Autumn
- Holiday - Holiday/No holiday
- Functional Day - NoFunc(Non Functional Hours), Fun(Functional hours

Defining the Dependent Variable

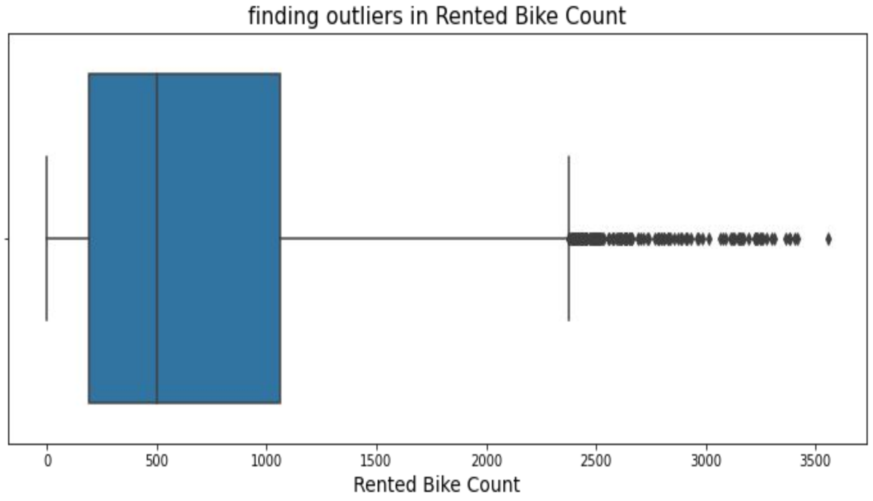


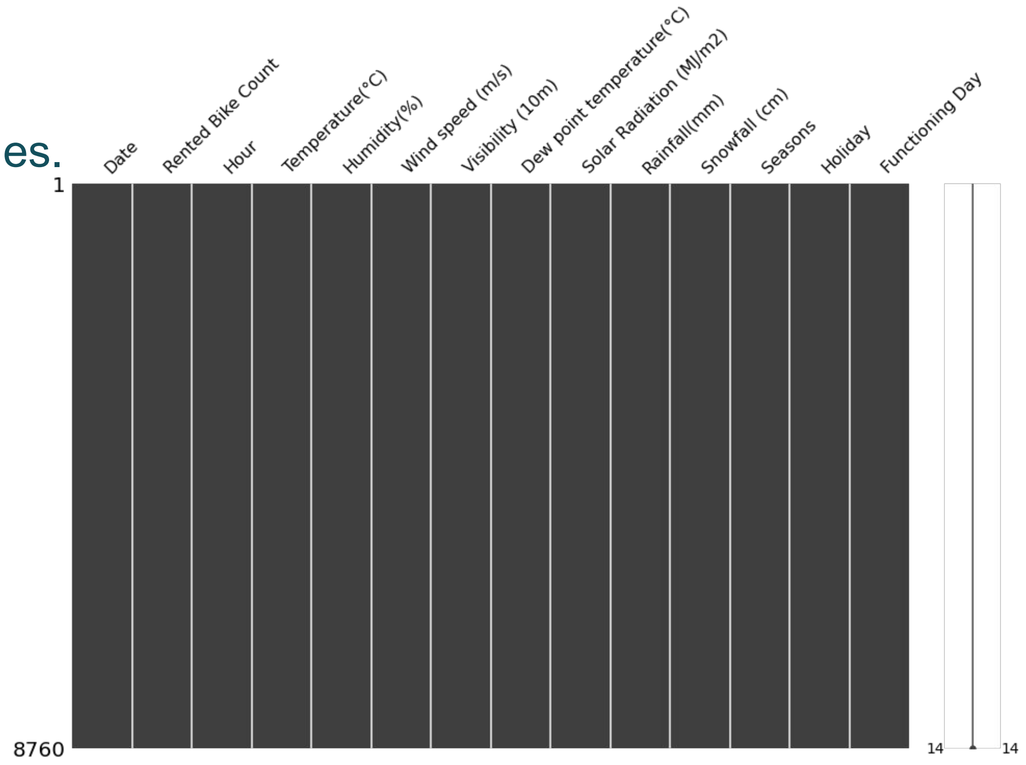




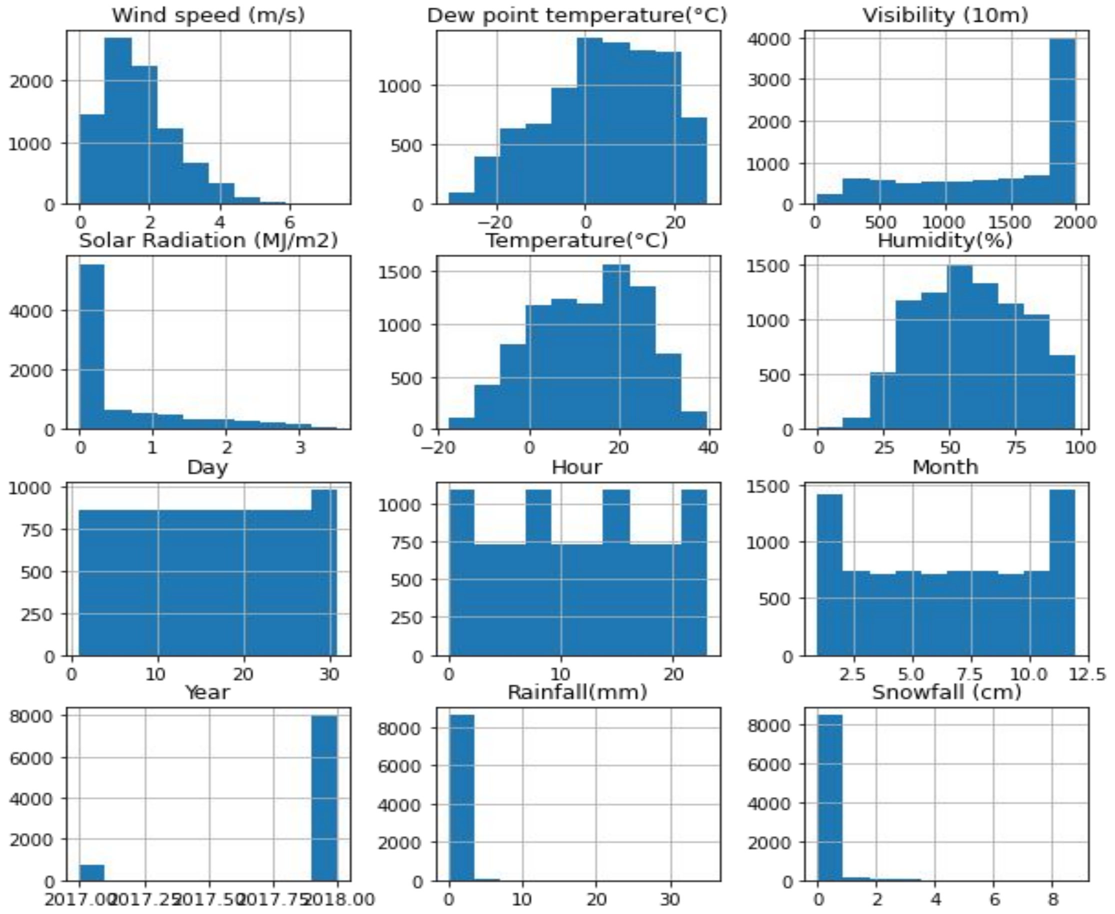
Exploratory Data Analysis

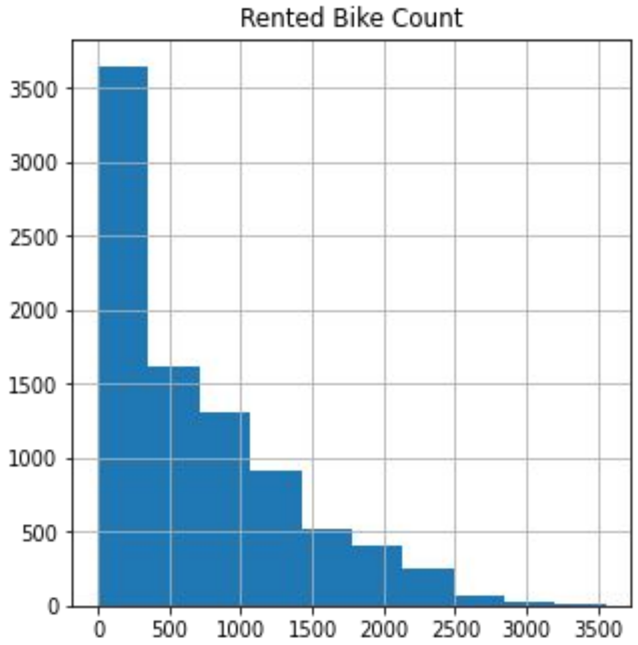
- Cleaning the null values.
- Finding outliers.
- Dependent and independent variables.
- Dropped some features.
- Value counts of features.
- Checking for Multicollinearity.



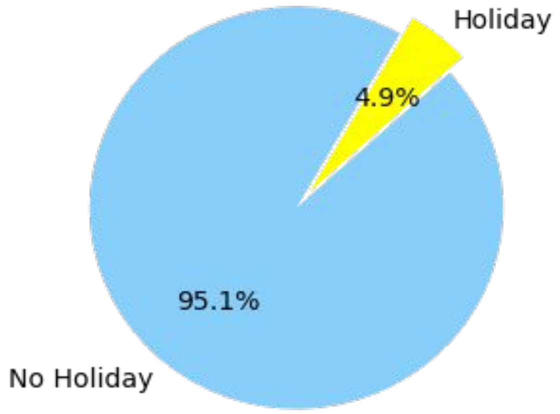


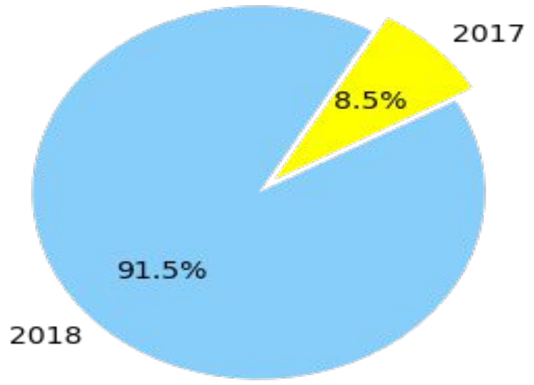
Exploratory Data Analysis

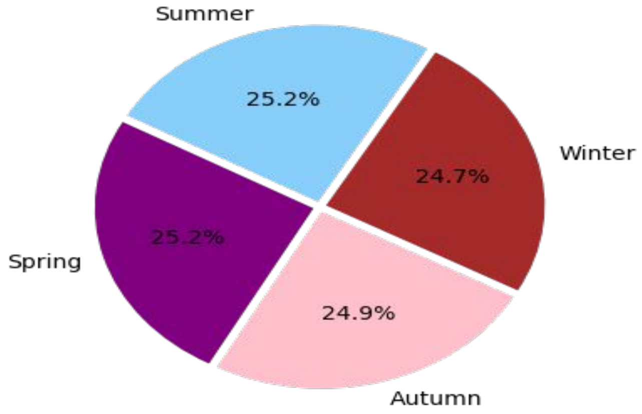


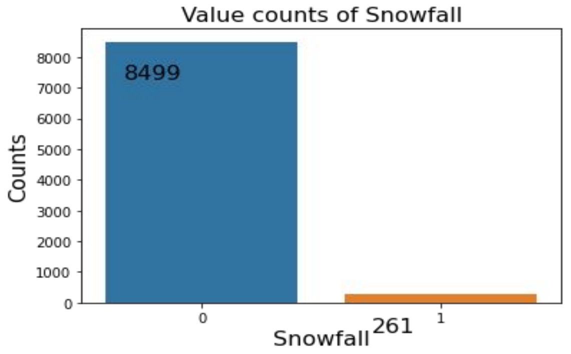


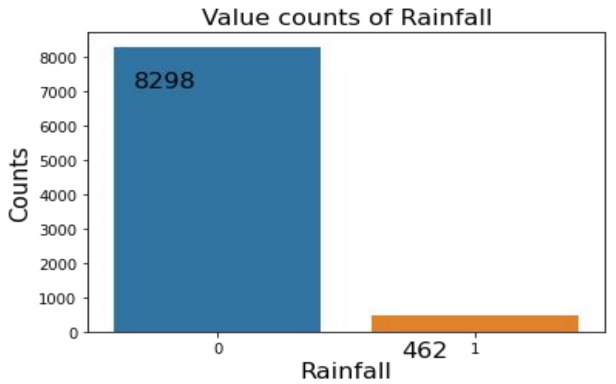
Exploratory Data Analysis

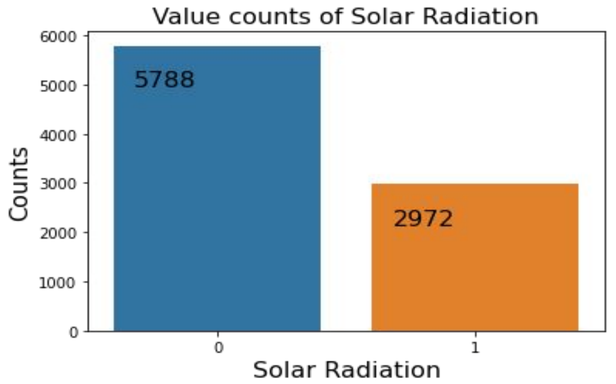






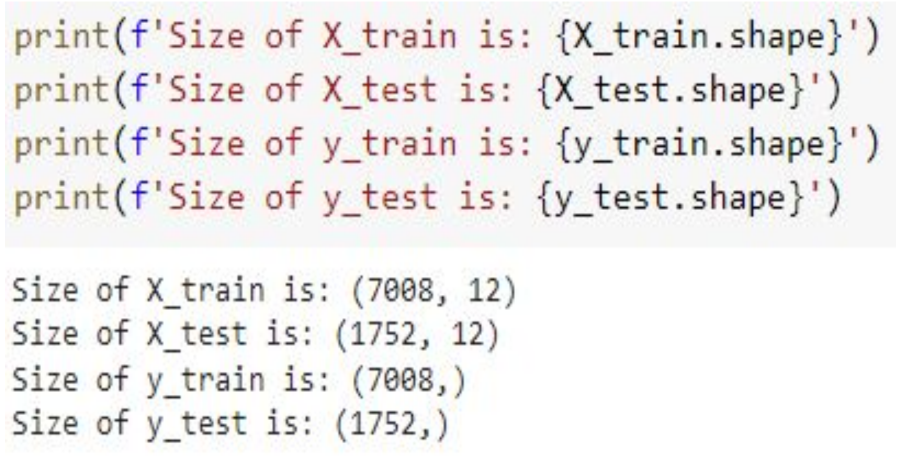






Data Preparation

- ➢ Preparing the new Dataframe with selected columns
- ➢ Variance inflation factor was used to check the correlation among the variables.
- ➢ Few features are dropped.
- ➢ Divided the dataset into train and test set in the ratio of 80:20
- ➢ StandardScaler was used to scale the data.

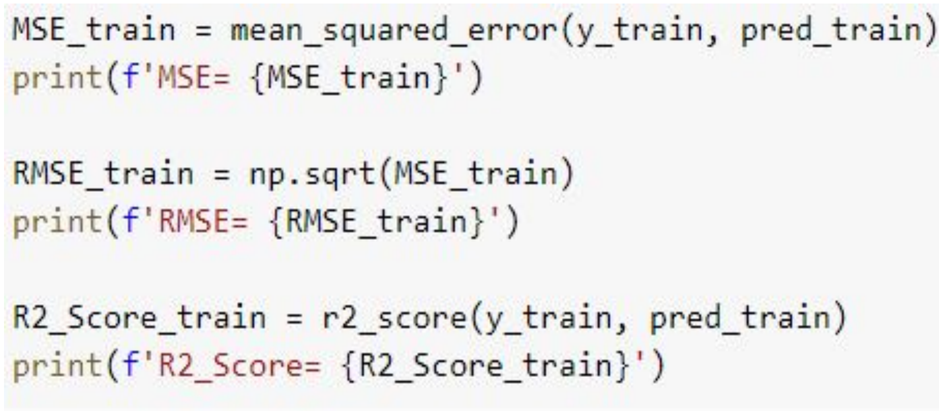


```
print(f'Size of X_train is: {X_train.shape}')
print(f'Size of X_test is: {X_test.shape}')
print(f'Size of y_train is: {y_train.shape}')
print(f'Size of y_test is: {y_test.shape}')
```

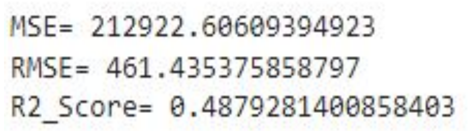
```
Size of X_train is: (7008, 12)
Size of X_test is: (1752, 12)
Size of y_train is: (7008,)
Size of y_test is: (1752,)
```

Linear Regression Model Implementation

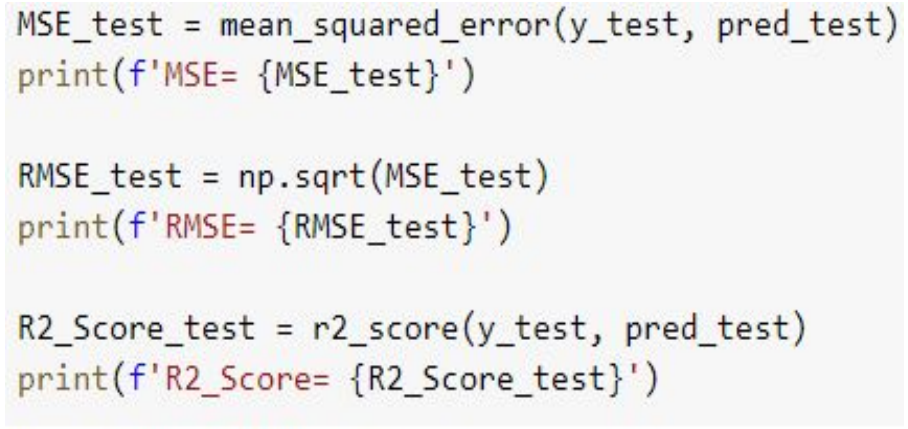
- ➔ The Scaled data was used for the model implementation.
- ➔ Fit the training dataset to the model.
- ➔ The regressor score of the model is 48.79%
- ➔ Defining the predicted values form the model.



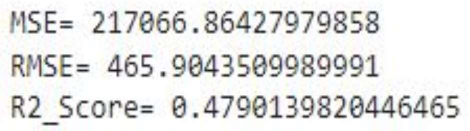
```python
MSE_train = mean_squared_error(y_train, pred_train)
print(f'MSE= {MSE_train}')

RMSE_train = np.sqrt(MSE_train)
print(f'RMSE= {RMSE_train}')

R2_Score_train = r2_score(y_train, pred_train)
print(f'R2_Score= {R2_Score_train}')
```



```
MSE= 212922.60609394923
RMSE= 461.435375858797
R2_Score= 0.4879281400858403
```

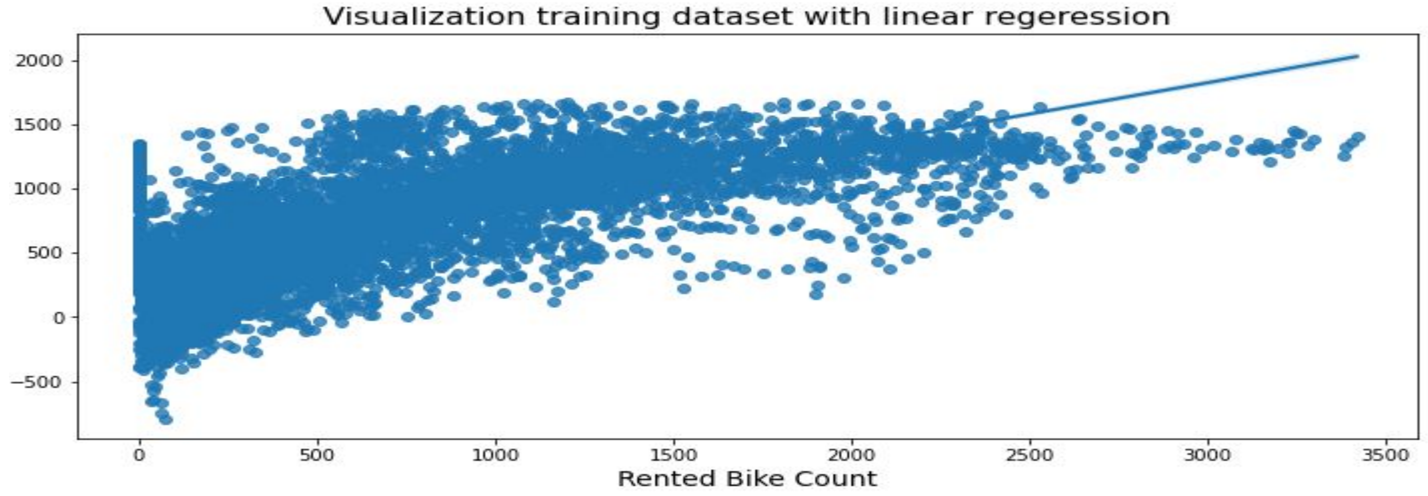


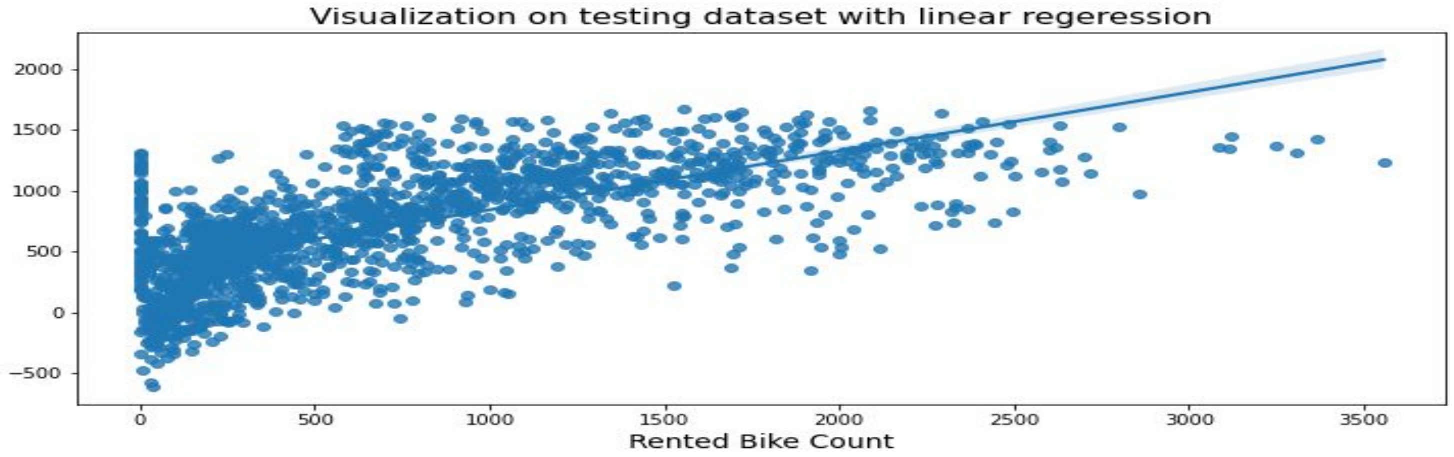
```python
MSE_test = mean_squared_error(y_test, pred_test)
print(f'MSE= {MSE_test}')

RMSE_test = np.sqrt(MSE_test)
print(f'RMSE= {RMSE_test}')

R2_Score_test = r2_score(y_test, pred_test)
print(f'R2_Score= {R2_Score_test}')
```



```
MSE= 217066.86427979858
RMSE= 465.9043509989991
R2_Score= 0.4790139820446465
```

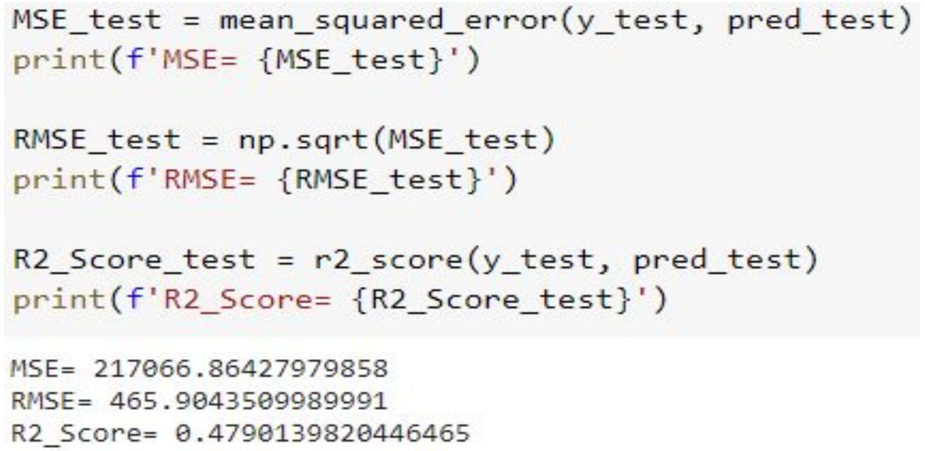
Linear Regression





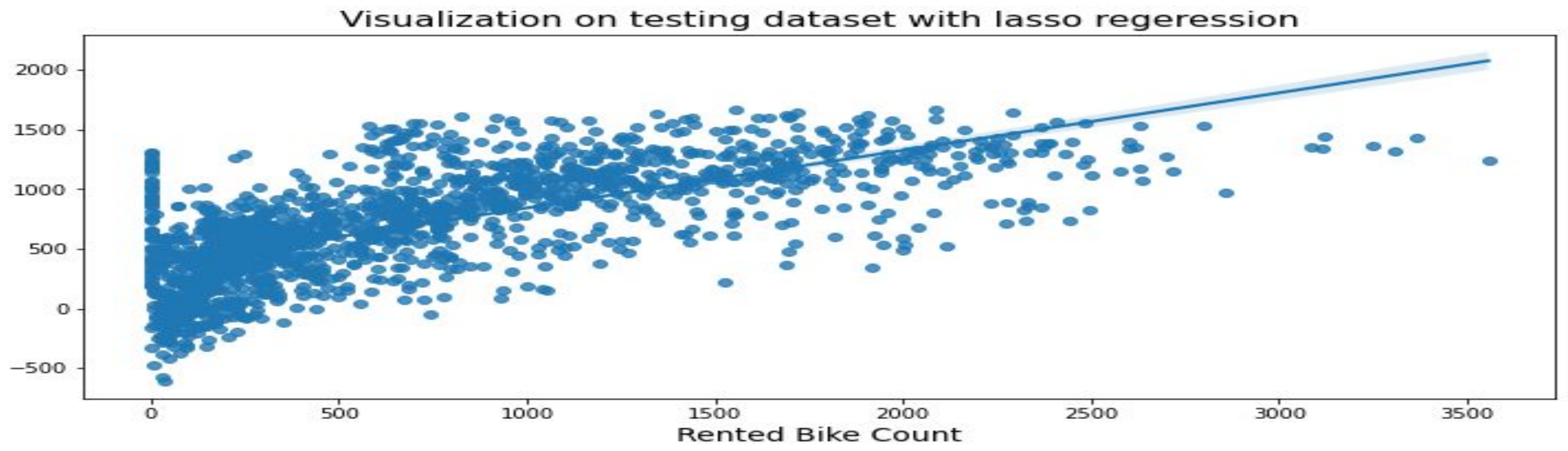
Lasso Regression

- ➔ The Scaled data was used for the model implementation.
- ➔ Fit the training dataset to the model.
- ➔ The regressor score of the model is **48.79%**
- ➔ Defining the predicted values form the model.



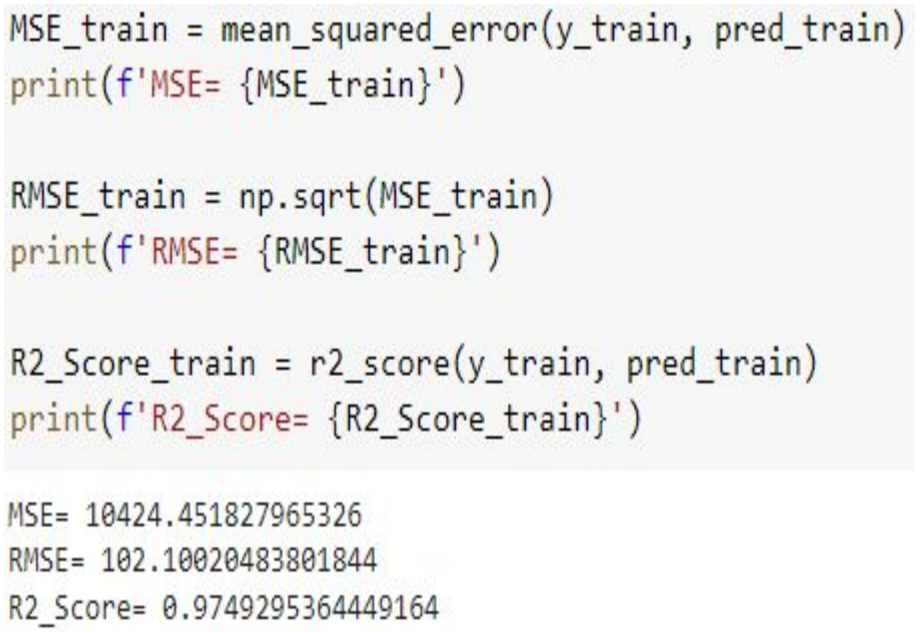
```
MSE_test = mean_squared_error(y_test, pred_test)
print(f'MSE= {MSE_test}')

RMSE_test = np.sqrt(MSE_test)
print(f'RMSE= {RMSE_test}')

R2_Score_test = r2_score(y_test, pred_test)
print(f'R2_Score= {R2_Score_test}')
```

```
MSE= 217066.86427979858
RMSE= 465.9043509989991
R2_Score= 0.4790139820446465
```



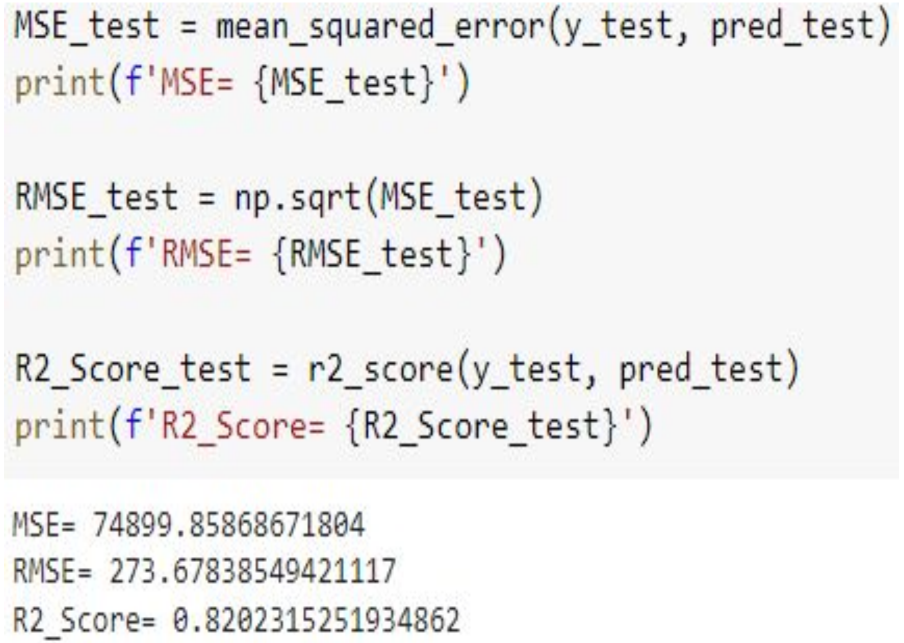
Random forest Regressor

- ➔ The Scaled data was stored in the dataframe.
- ➔ Fit the training dataset to the model.



```
MSE_train = mean_squared_error(y_train, pred_train)
print(f'MSE= {MSE_train}')

RMSE_train = np.sqrt(MSE_train)
print(f'RMSE= {RMSE_train}')

R2_Score_train = r2_score(y_train, pred_train)
print(f'R2_Score= {R2_Score_train}')
```

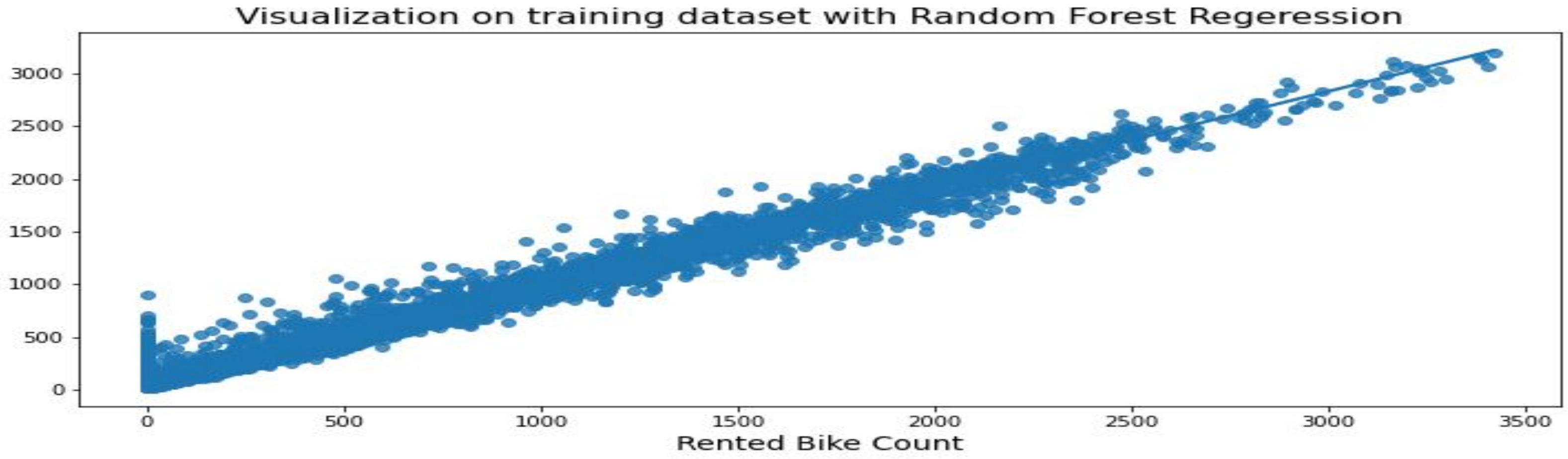
```
MSE= 10424.451827965326
RMSE= 102.10020483801844
R2_Score= 0.9749295364449164
```

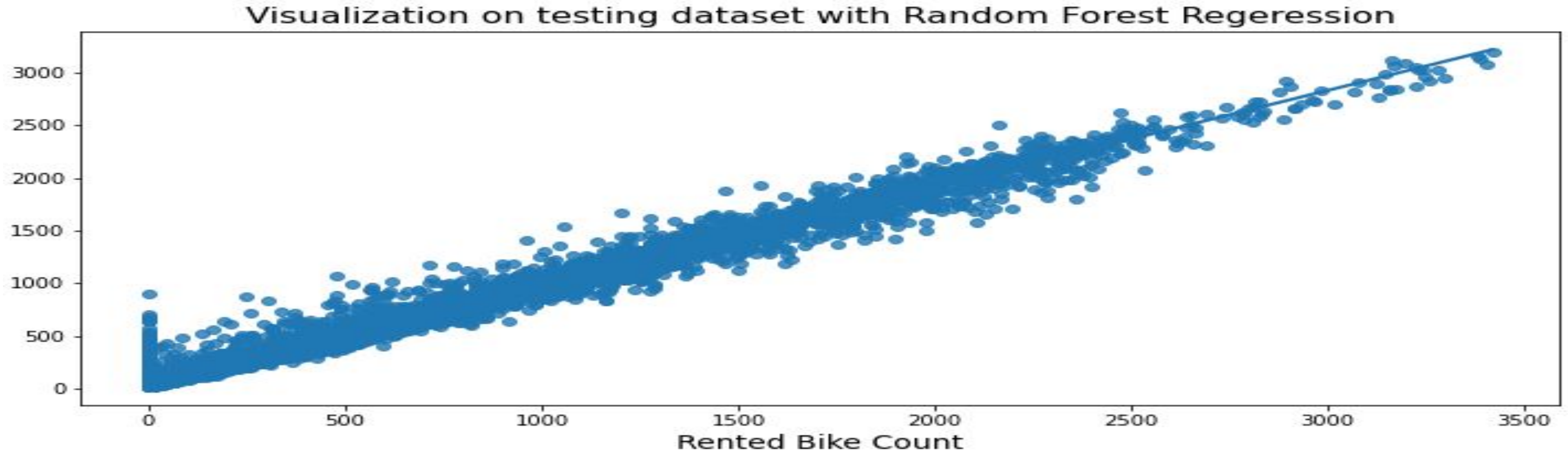


```
MSE_test = mean_squared_error(y_test, pred_test)
print(f'MSE= {MSE_test}')

RMSE_test = np.sqrt(MSE_test)
print(f'RMSE= {RMSE_test}')

R2_Score_test = r2_score(y_test, pred_test)
print(f'R2_Score= {R2_Score_test}')
```

```
MSE= 74899.85868671804
RMSE= 273.67838549421117
R2_Score= 0.8202315251934862
```

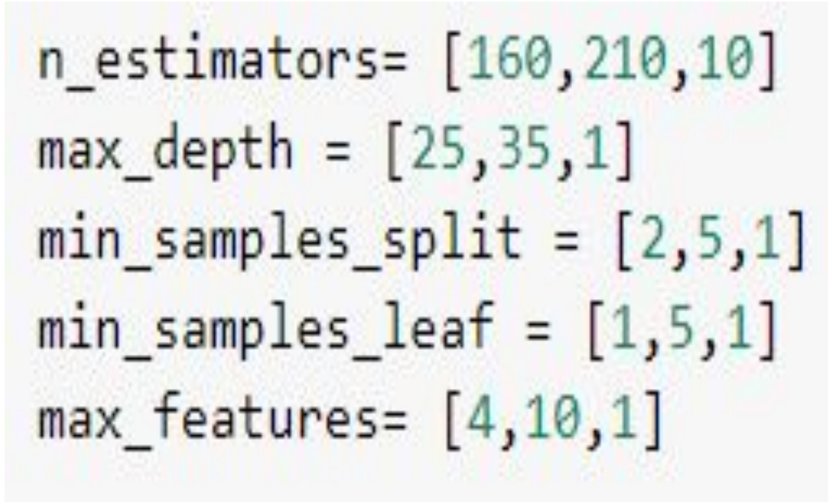
Random Forest Regressor





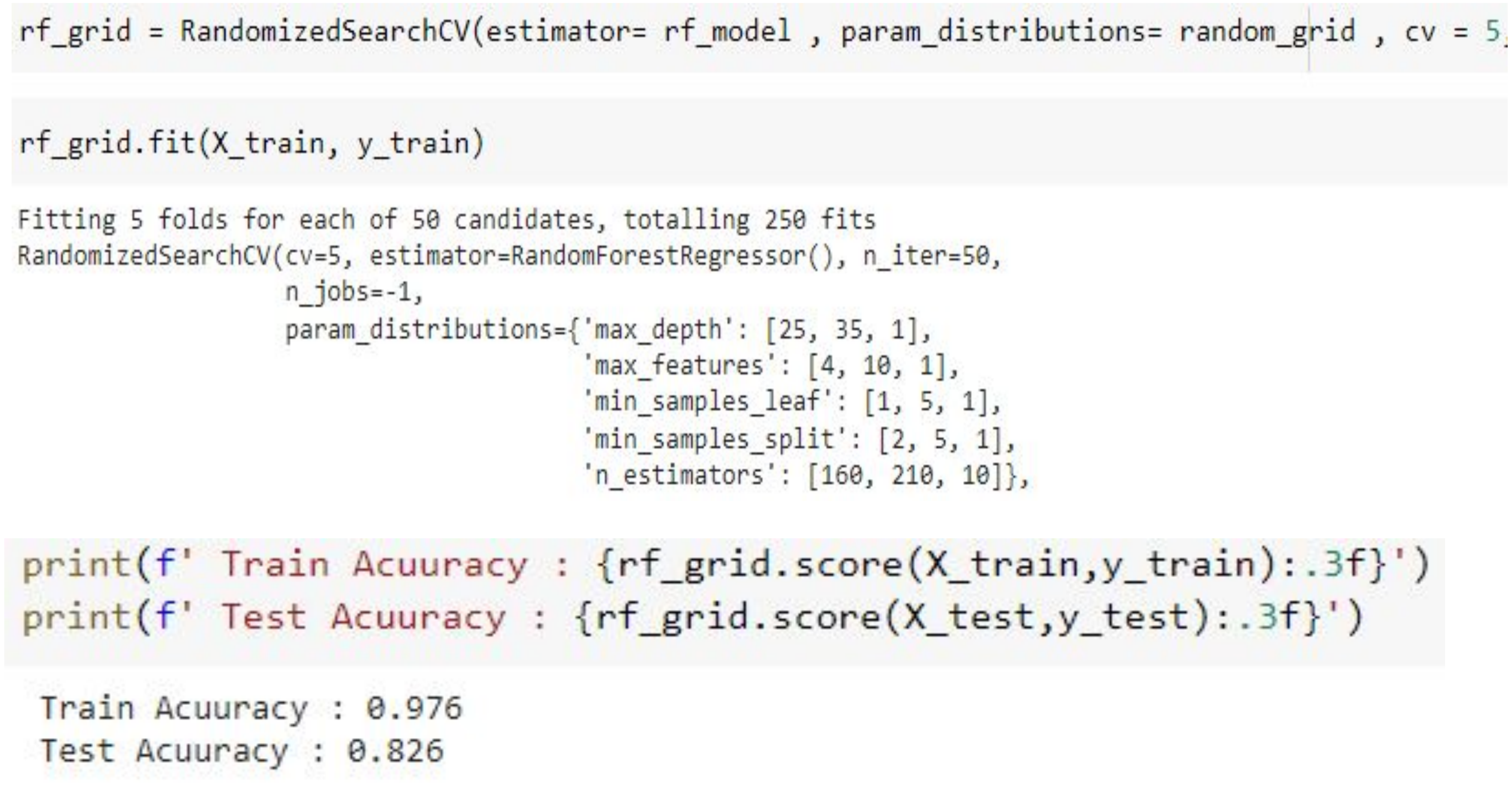
Random forest Regressor (Hyperparameter Tuning)

- **Max_depth : The max_depth of a tree in Random Forest is defined as the longest path between the leaf root node and the left node.**
- **Min_samples_leaf : This hyperparameter specifies the minimum number of samples that should be in the leaf node after splitting a node.**
- **Min_samples_split : A parameter that tells the decision tree in a random forest the minimum required number of observation in given node.**
- **N_estimator : We know that a Random forest algorithm is a nothing but number of trees.**
- **Max_features : This resembles the number of maximum features provided to each tree in a Random Forest.**



```
n_estimators= [160,210,10]
max_depth = [25,35,1]
min_samples_split = [2,5,1]
min_samples_leaf = [1,5,1]
max_features= [4,10,1]
```

Cross validation on Random forest Regressor



```
rf_grid = RandomizedSearchCV(estimator= rf_model , param_distributions= random_grid , cv = 5
```

```
rf_grid.fit(X_train, y_train)
```

```
Fitting 5 folds for each of 50 candidates, totalling 250 fits
RandomizedSearchCV(cv=5, estimator=RandomForestRegressor(), n_iter=50,
                   n_jobs=-1,
                   param_distributions={'max_depth': [25, 35, 1],
                                        'max_features': [4, 10, 1],
                                        'min_samples_leaf': [1, 5, 1],
                                        'min_samples_split': [2, 5, 1],
                                        'n_estimators': [160, 210, 10]},
```

```
print(f' Train Acuuracy : {rf_grid.score(X_train,y_train):.3f}')
print(f' Test Acuuracy : {rf_grid.score(X_test,y_test):.3f}')
```

```
 Train Acuuracy : 0.976
 Test Acuuracy : 0.826
```

Conclusion

- ➢ We calculate the numbers of bikes rented on Function day, holidays, monthly, day wise, season wise and yearly.
- ➢ Value counts of these rented bikes between 5pm to 8pm is pretty high.
- ➢ It shows very less counts on holidays and high counts during working days.
- ➢ In the months of May, June, July and October we have a high number of rented bikes counts.
- ➢ Use of these bikes in the first 10 days of every month is much higher than other days.
- ➢ We assumed that 7 in the day of week as the Sunday and stores are mostly closed during Sundays.
- ➢ In the rainy season and snow season use of these bikes was very less.
- ➢ In 2017, the count of rented bikes was only 8.5% (744) but within one year use of these bikes increased to 91.5% (8016).
- ➢ Linear Regression, Lasso Regression and Random Forest Regression are used to train the model.
- ➢ It is better to implement the Random Forest Regression rather than going for Linear and Lasso Regression..

Thank you