CAPSTONE PROJECT - 2

BIKE SHARING DEMAND PREDICTION

(Supervised Machine Learning Regression)

By

VIKASKUMAR SHARMA

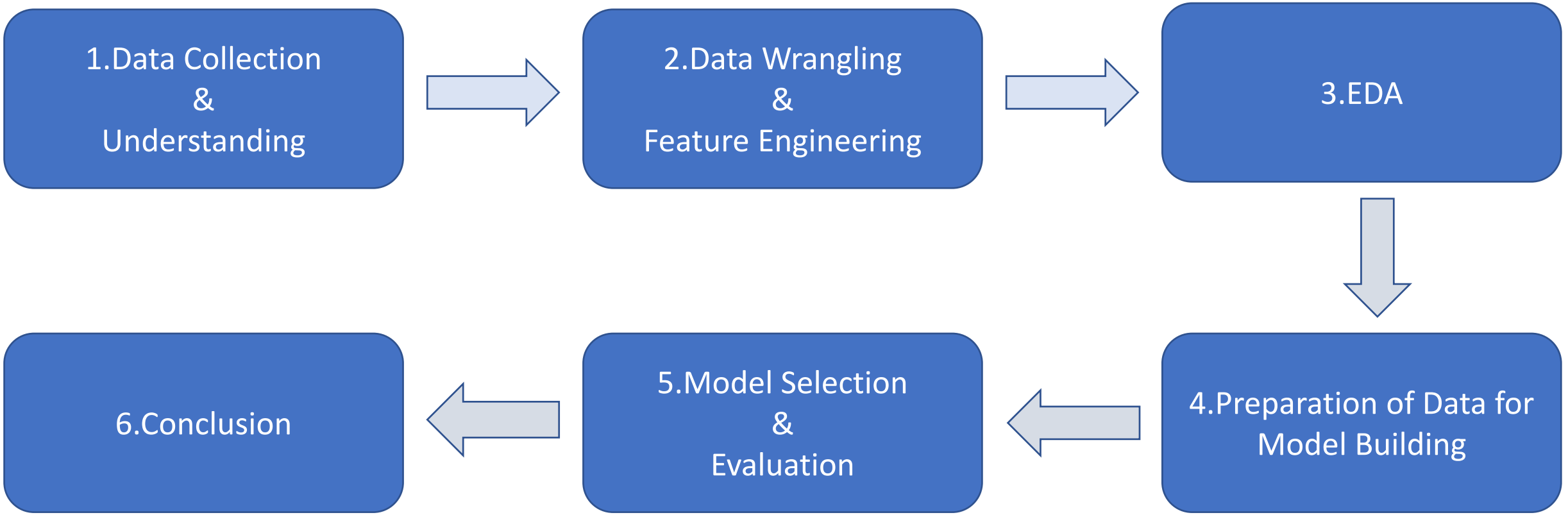
(Cohort Tosh)

❑ Problem Statement

- The Data is of Rental Bike sharing company based out of Seoul, South Korea which is into bike rental business. 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 main aim of this project is to predict the bike count required at each hour for the stable supply of rental bikes by trying different machine learning models and eventually selecting the best one whose accuracy is best .

❑ Work Flow

- We will divide our project into 6 major steps:-



❑ Data Description

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

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 - %
- 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)

❑ Data Wrangling & Feature Engineering

As we know we had 8760 observations and 14 features.

- **Categorical Features:** Seasons, Holiday and Functioning day.
- **Numerical Columns:** Date, Hour, Temperature, Humidity, Wind speed, Visibility, Dew point temperature, Solar radiation, Rainfall, Snowfall, Rented Bike count .
- **Rename Columns:** We renamed columns because they had units mentioned in brackets and was difficult to copy the feature name while working.

```
df.columns
```

```
Index(['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'],
      dtype='object')
```

```
[13] #Rename the complex columns name
     df=df.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_temperature',
                           'Solar Radiation (MJ/m2)':'Solar_Radiation',
                           'Rainfall(mm)':'Rainfall',
                           'Snowfall (cm)':'Snowfall',
                           'Functioning Day':'Functioning_Day'})
```

❑ Data Wrangling & Feature Engineering

- **We had zero null values in our dataset.**
- **Zero Duplicate entries found.**
- **We changed the data type of Date column from ‘object’ to ‘datetime64[ns]’. This was done for feature engineering.**
- **We Created four new columns with the help of Date column ‘Month’, ’Year’, ‘Day’ & ‘Weekdays_Weekend’. Which were further used for EDA. And later we dropped Date,Day & Year column.**

```
[10] #Check for count of missing values in each column.
     df.isnull().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
```

```
# Checking Duplicate Values
value=len(df[df.duplicated()])
print("The number of duplicate values in the data set is = ",value)
```

```
The number of duplicate values in the data set is =  0
```

```
[14] # Changing the "Date" column into three "year","month","day" column
     df['Date'] = df['Date'].apply(lambda x:
                                   dt.datetime.strptime(x,"%d/%m/%Y"))
```

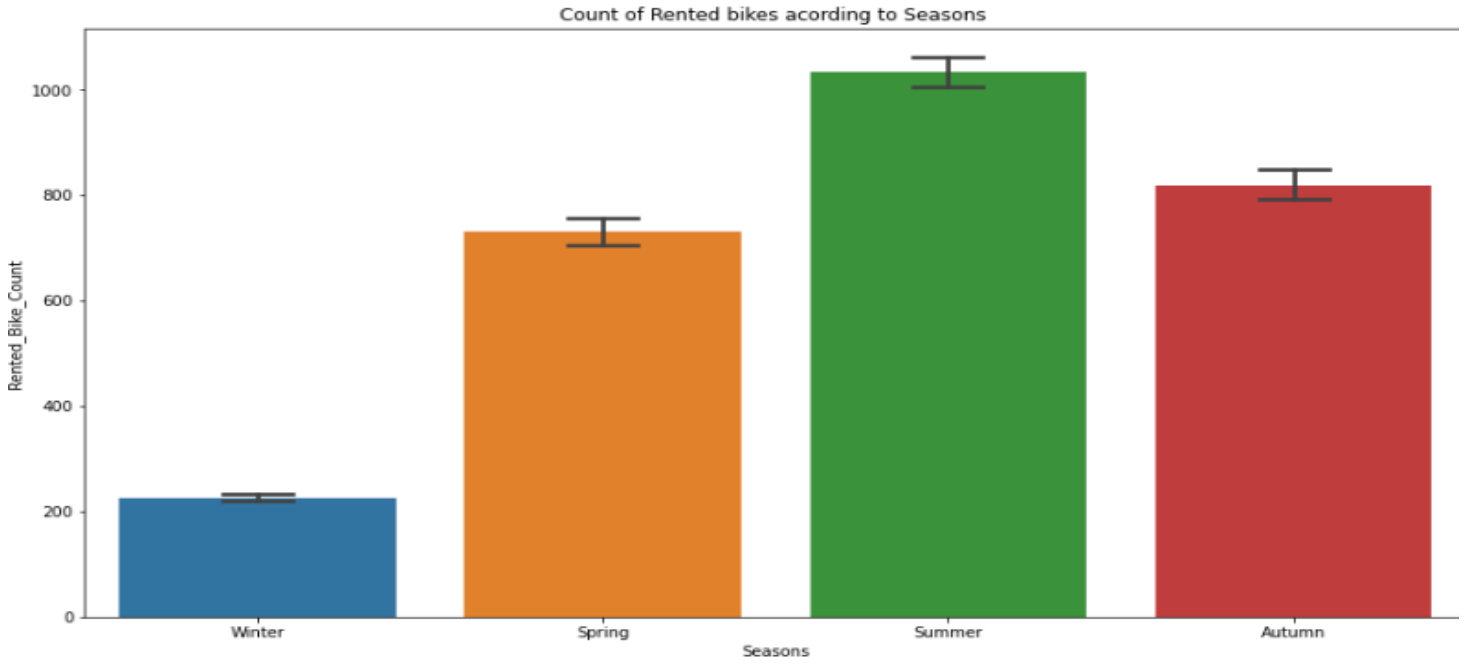
```
[15] df['year'] = df['Date'].dt.year
     df['month'] = df['Date'].dt.month
     df['day'] = df['Date'].dt.day_name()
```

```
#creating a new column of "weekdays_weekend" and drop the column "Date","day","year"
df['weekdays_weekend']=df['day'].apply(lambda x : 1 if x=='Saturday' or x=='Sunday' else 0 )
df=df.drop(columns=['Date','day','year'],axis=1)
```

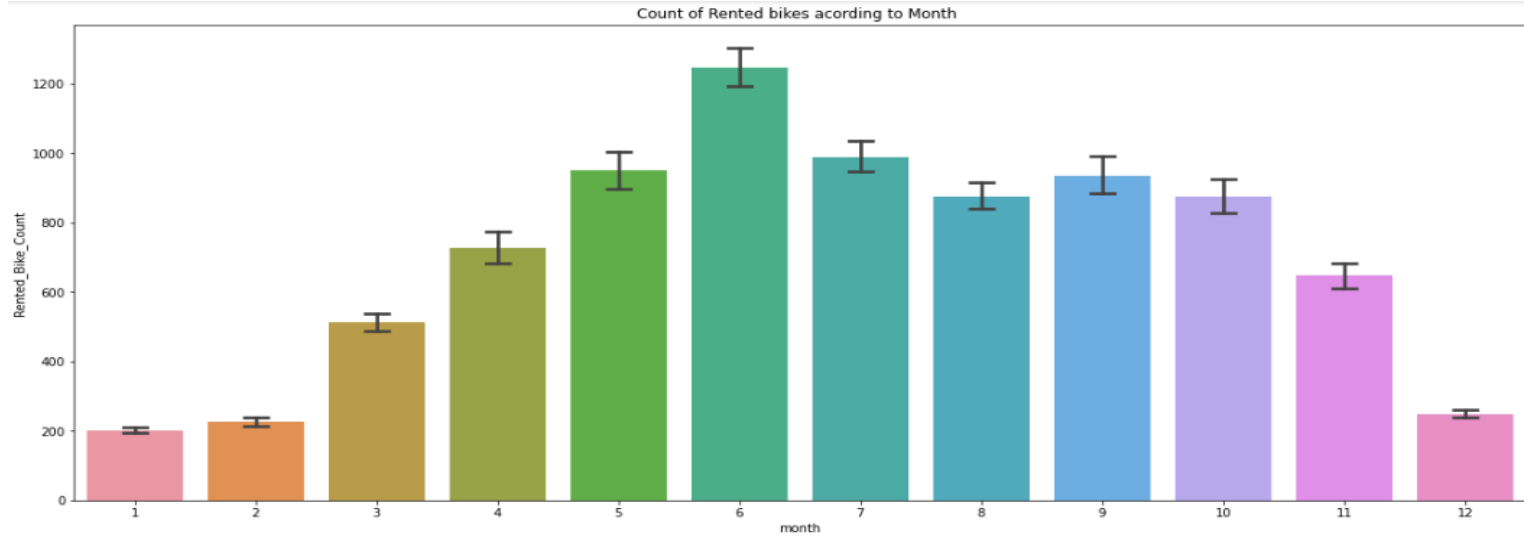
❑ Exploratory Data Analysis (EDA)

- **Relation of rented bike count with categorical features:**

Summer season had the highest Bike Rent Count. People are more likely to take rented bikes in summer. Bike rentals in winter is very less compared to other seasons.



- **From March Bike Rent Count started increasing and it was highest in June.**

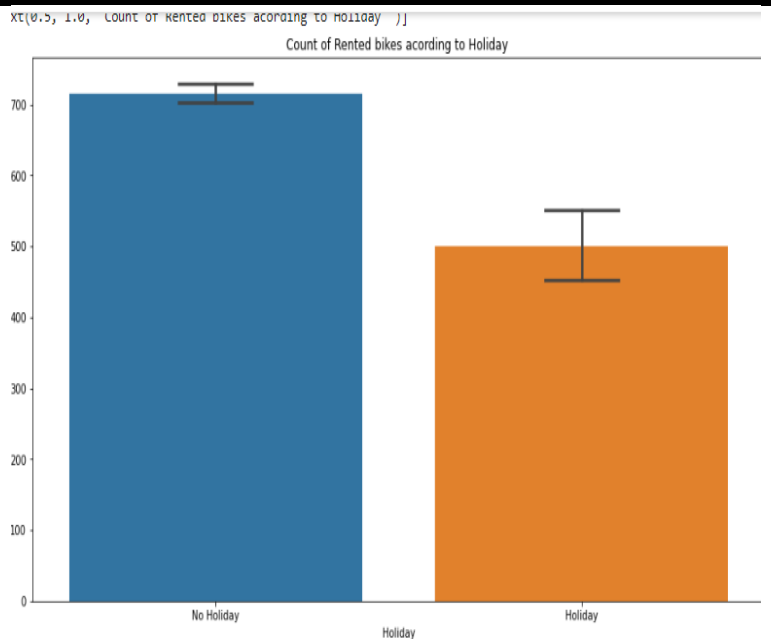


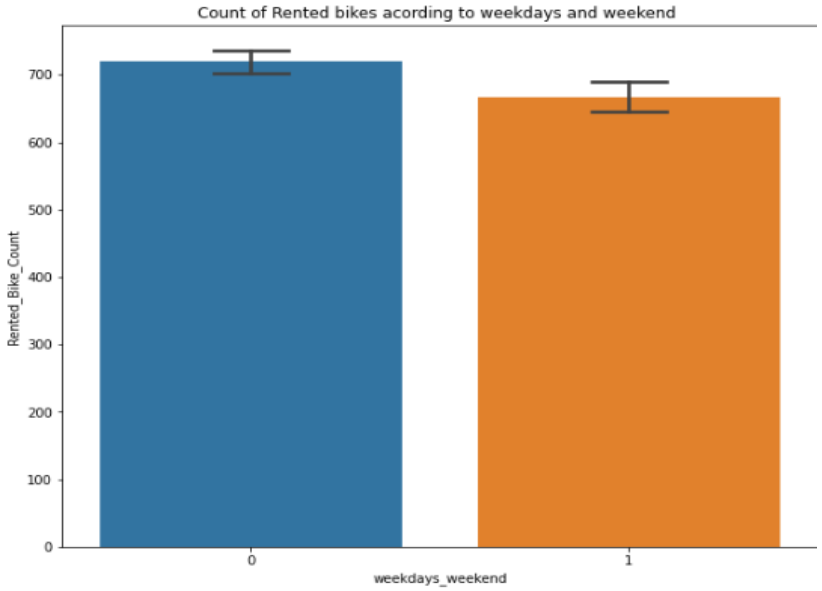
EDA

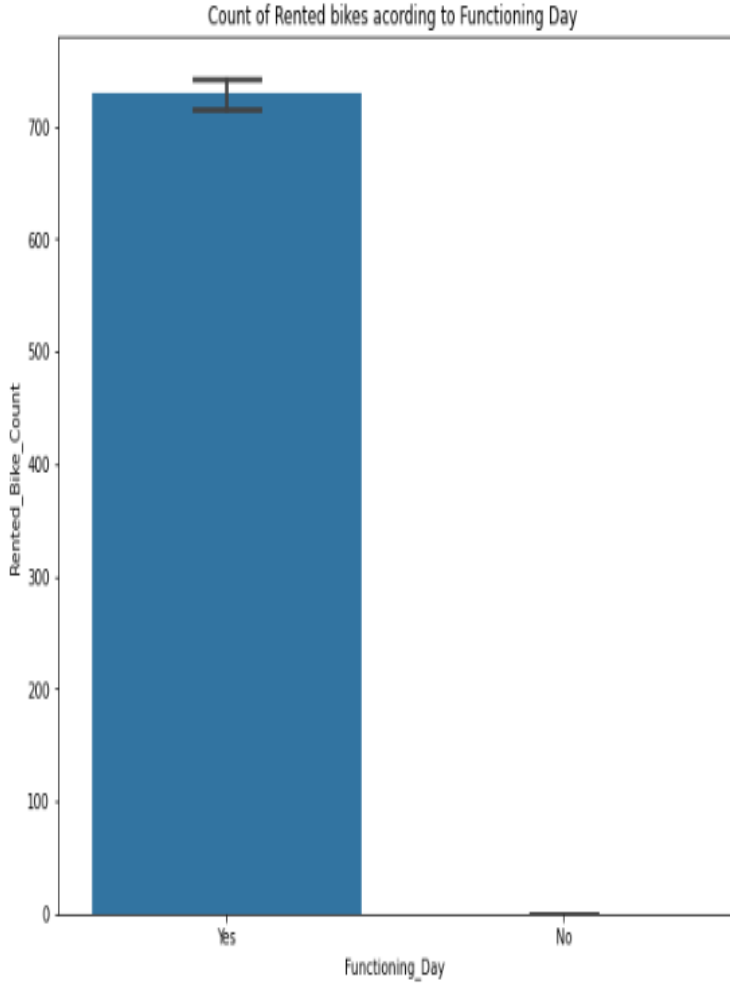
- **High number of bikes were rented on No Holidays. Which is almost 700 bikes.**

- **Zero Bikes were rented on no functioning day. More than 700 bikes rented on functioning day.**

- **More than 700 bikes were rented on weekdays. On weekdays, almost 650 bikes were rented.**







❑ EDA

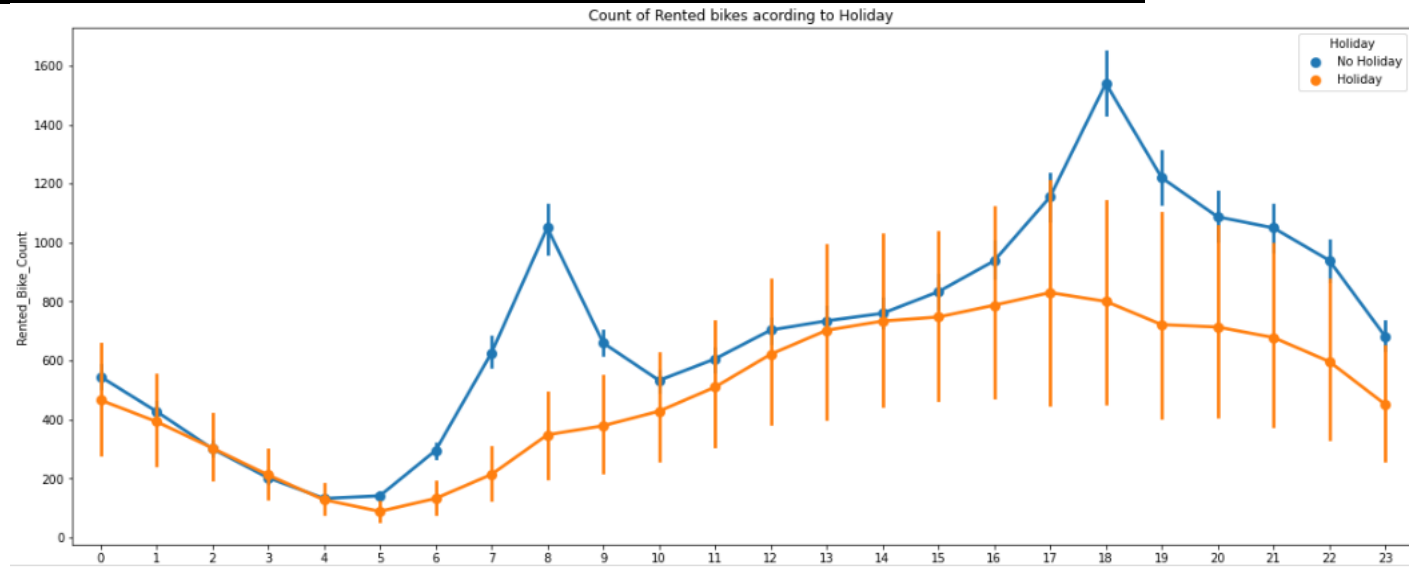
Observations:

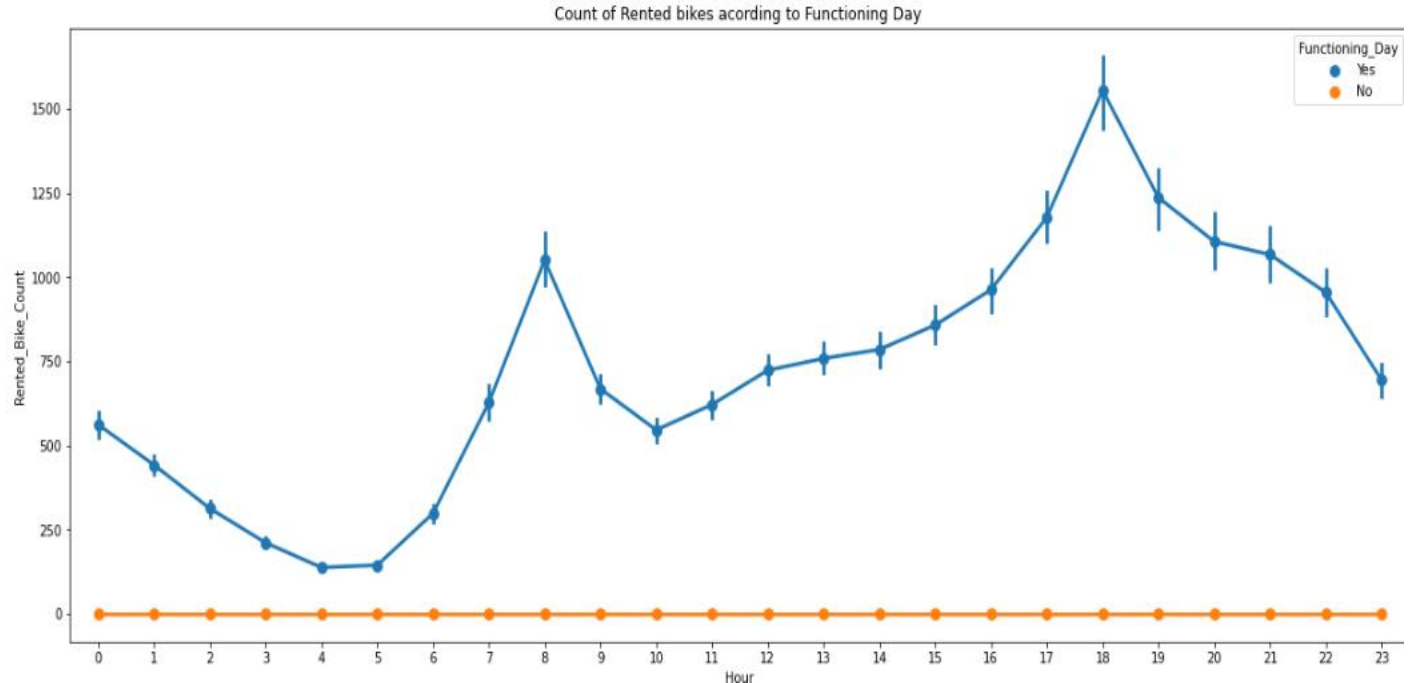
1)Here we observed that, Bike rental trend according to hours is almost similar in all scenarios.

2)There is sudden peak between 6/7AM to 10 AM. Office /College going time could be the reason for this sudden peak on NO Holiday. But on Holiday the case is different, very less bike rentals happened.

3) Again there is peak between 4PM to 7 PM. may be its office leaving time for the above people.(NO Holiday).

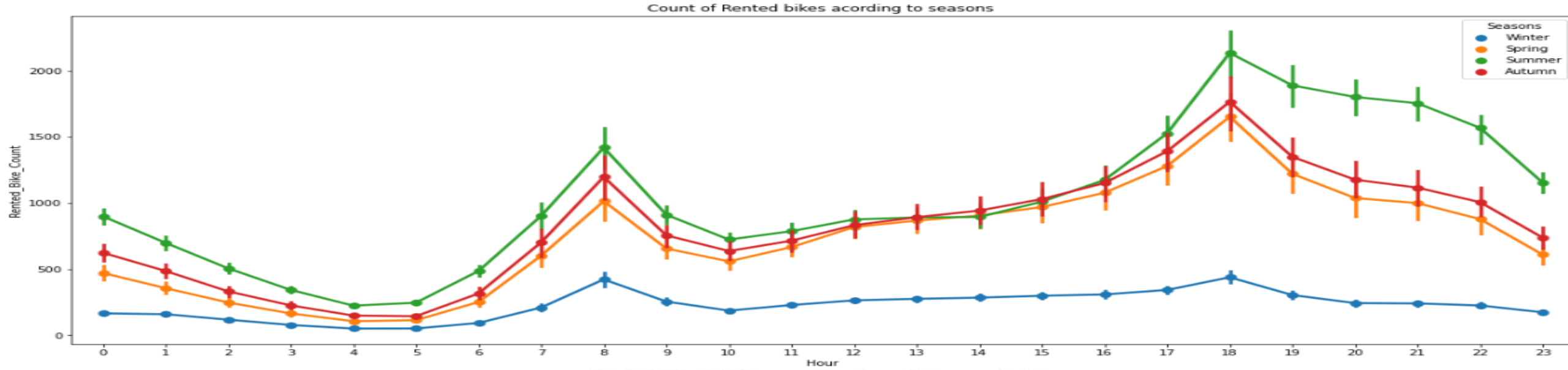
4)Here the trend for functioning day is same as of No holiday. Only the difference is on No functioning day there were zero bike rentals.

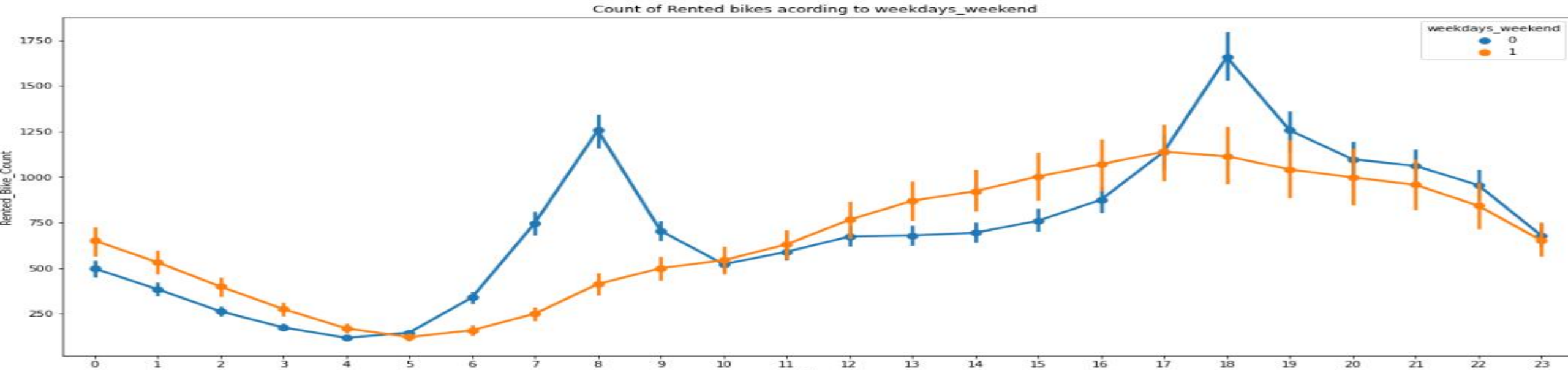




EDA

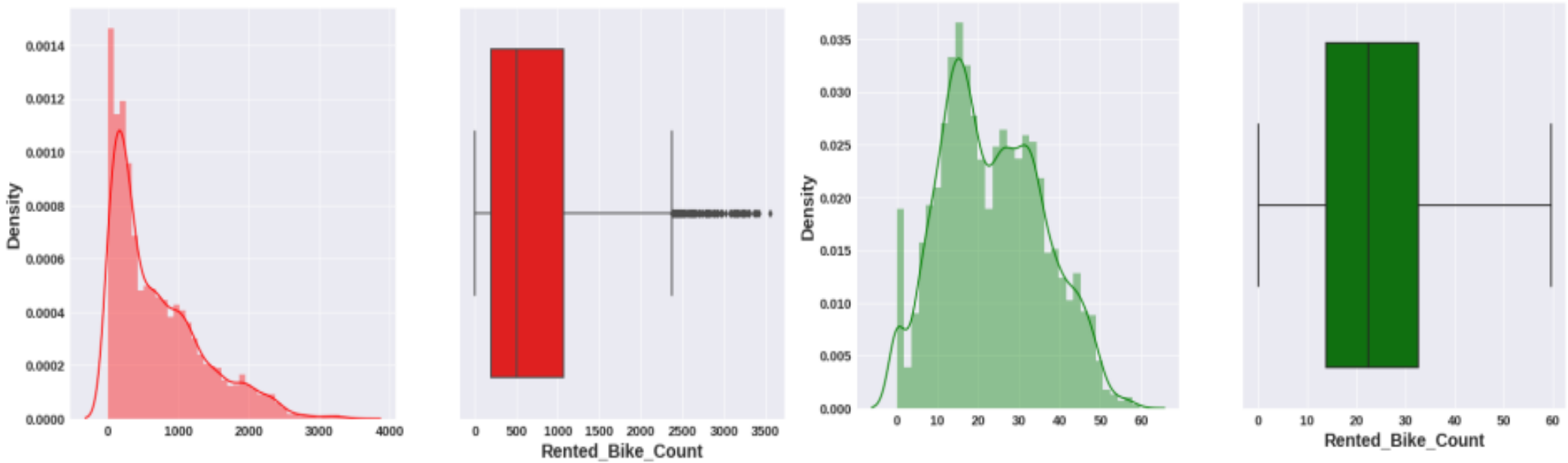
- **Bike Rent Trend according to hour in different scenarios.**





EDA

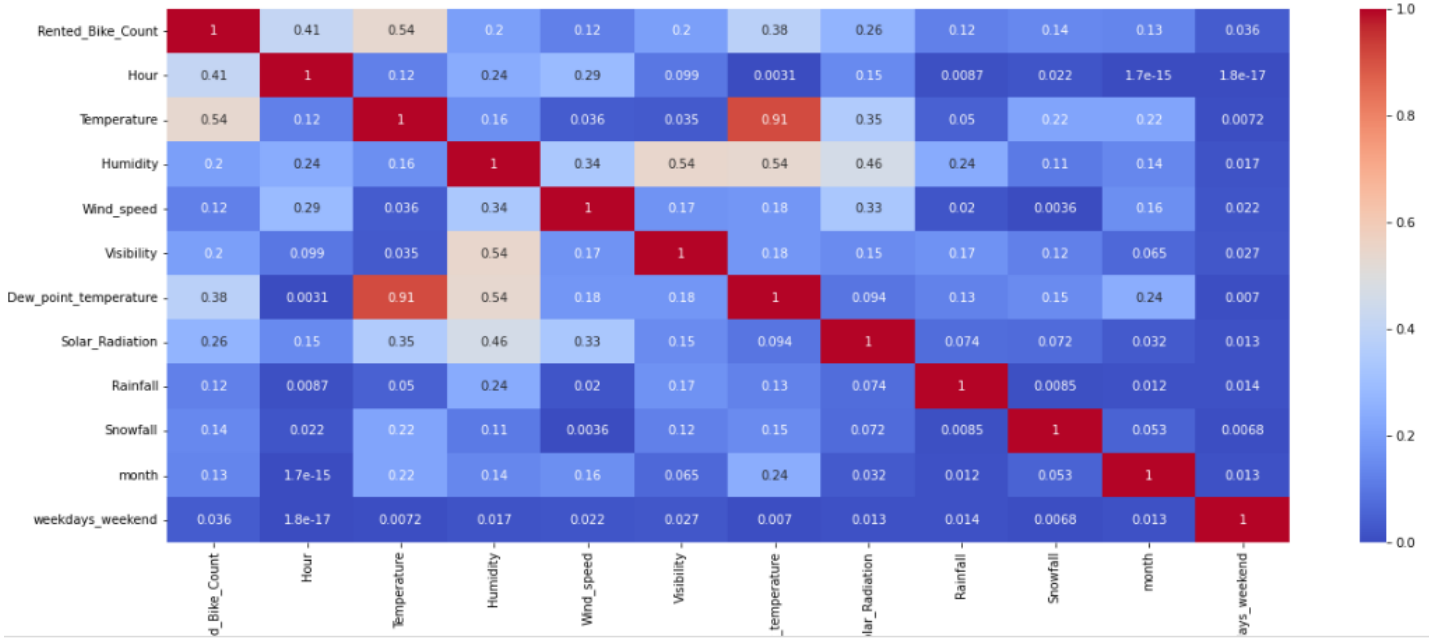
Distribution of target variable- Bike Rent Count



- Distribution is rightly skewed and some outliers are observed.
- To normalize the distribution we applied square root method. After normalization no outliers were found.

❑ Preparation of Data for model Building

- With the heat map we dropped highly correlated variables. As we can see Temperature and Dew point temperature are 91 % correlated. So we dropped the Dew point temperature because it has very low correlation with our target variable as compared to temperature
- Later by using variation inflation factor we dropped ‘Visibility’ and ‘Humidity’ features as they had VIF value more than 5.
- Next we created dummy variables for categorical Seasons column and did mapping with 0 and 1 for holiday and functioning column.
- Thus we prepared our data for model building.




```
[49] #Assign all catagoriacla features to a variable
     categorical_features=list(df.select_dtypes(['object','category']).columns)
     categorical_features=pd.Index(categorical_features)
     categorical_features

     Index(['Seasons', 'Holiday', 'Functioning_Day'], dtype='object')
```

```
[50] #creat a copy
     bike_df_copy = df

     def one_hot_encoding(data, column):
         data = pd.concat([data, pd.get_dummies(data[column], prefix=column, drop_first=True)], axis=1)
         data = data.drop([column], axis=1)
         return data

     for col in categorical_features:
         bike_df_copy = one_hot_encoding(bike_df_copy, col)
     bike_df_copy.head()
```

❑ Model Selection & Evaluation

As this is the regression problem we are trying to predict continuous value. For this we used following regression models.

- Linear Regression
- Lasso regression (regularized regression)
- Ridge Regression(regularized regression)
- Elastic Net Regression
- Decision Tree regression.
- Random forest regression
- Gradient Boosting regression.
- Gradient Boosting GridSearchCV

Assumptions of regression line:

1.The relation between the dependent and independent variables should be almost linear.

2.Mean of residuals should be zero or close to 0 as much as possible. It is done to check whether our line is actually the line of “best fit”.

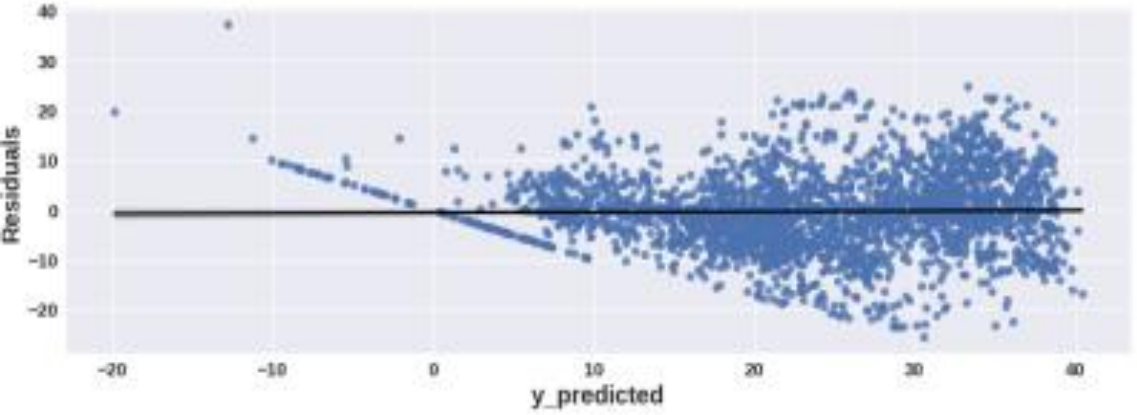
3.There should be homoscedasticity or equal variance in a regression model. This assumption means that the variance around the regression line is the same for all values of the predictor variable (X).

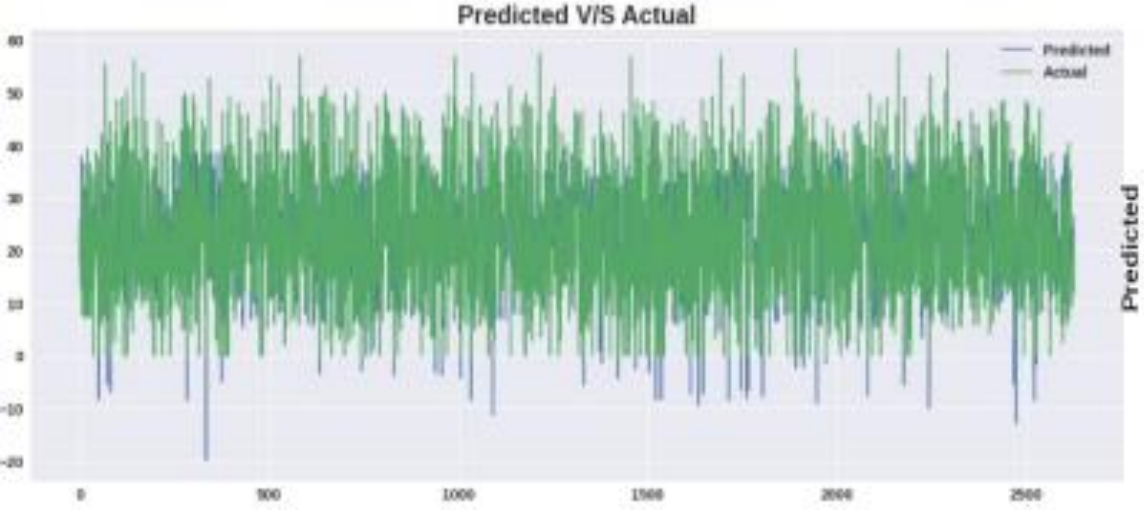
4.There should not be multicollinearity in regression model. Multicollinearity generally occurs when there are high correlations between two or more independent variables.

❑ Linear Regression

Scores on Train set

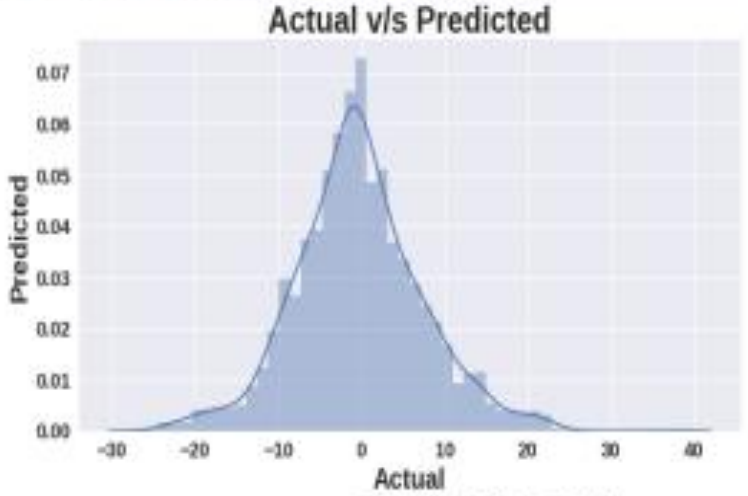
The Mean Absolute Error (MAE) is 5.8555397241788345.
The Mean Squred Error(MSE) is 60.29949292444555.
The Root Mean Squared Error(RMSE) is 7.765274813195316.
The R2 Score is 0.6123528085603556.

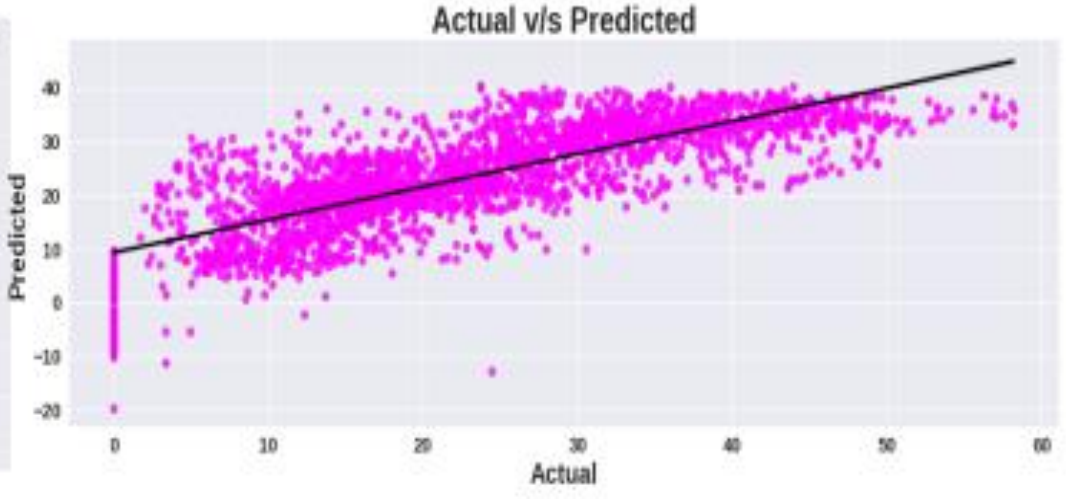




Scores on Test set

The Mean Absolute Error (MAE) is 5.834169822951748.
The Mean Squred Error(MSE) is 58.624247223024895.
The Root Mean Squared Error(RMSE) is 7.656647257319936.
The R2 Score is 0.6183269673651 99.





Mean of residuals should be zero or close to 0 as much as possible. It is done to check whether our line is actually the line of “best fit”

Lasso Regression

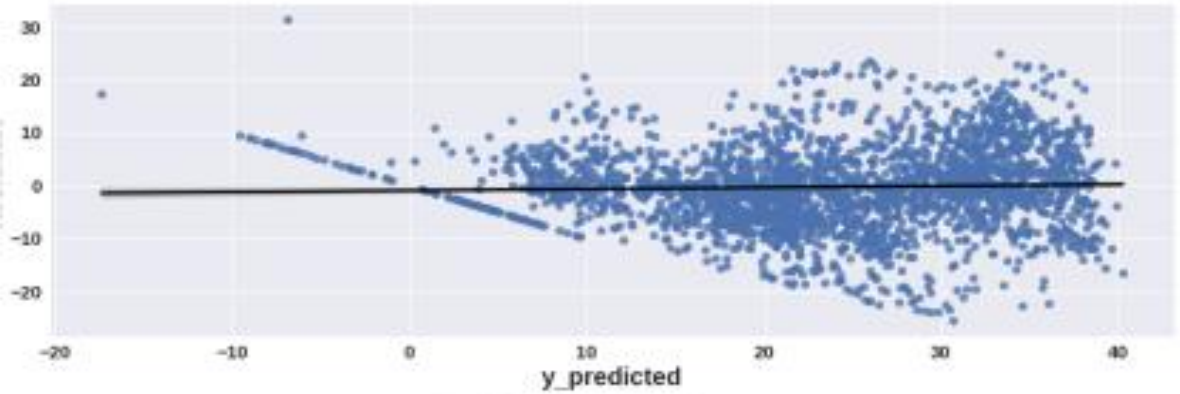
➢ **Lasso (Hyper-parameter tuned- alpha=0.01)**

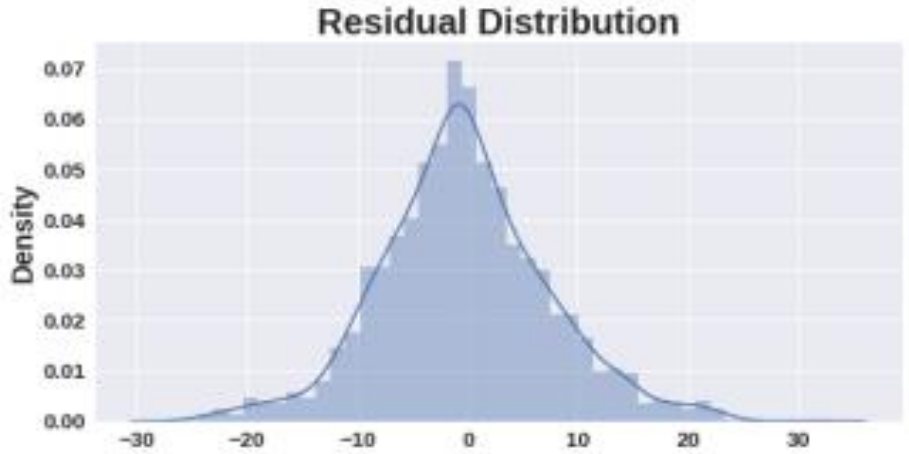
Scores on Train set

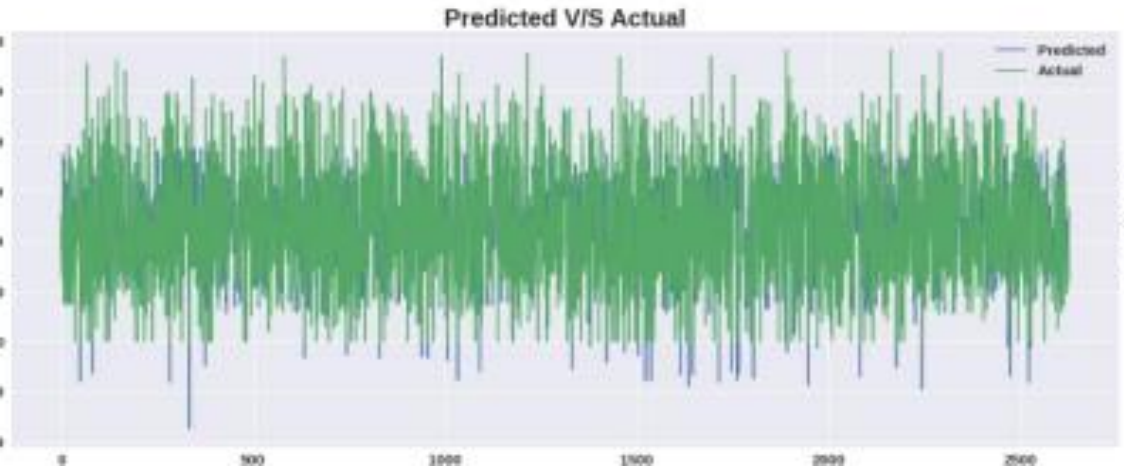
```
MSE  :  87.27547977434 75
RMSE  :  9.342134647624572
MAE  :  7.006477442680442
R2  :  0.4344993590897 9664
```

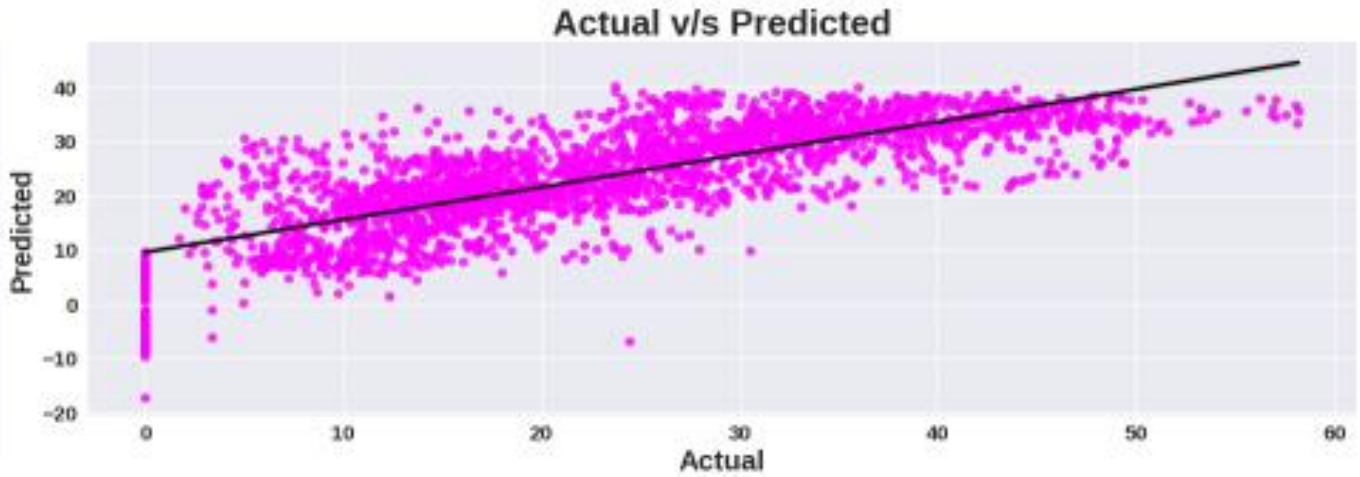
Scores on Test set

```
MSE  :  94.39530688333724
RMSE  :  9.715724722496889
MAE  :  7.298715427255486
R2  :  0.4006124180284252
```









❑ Ridge Regression

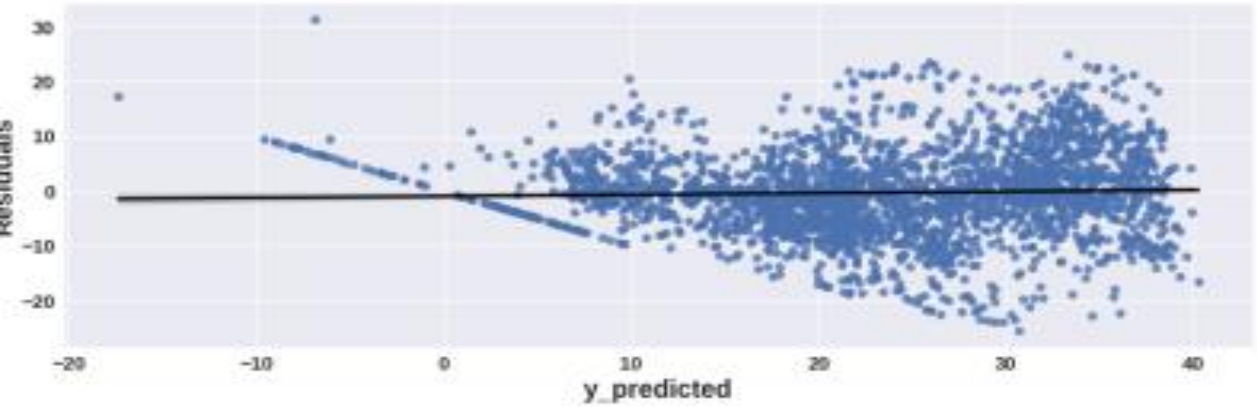
➢Ridge (Hyper-parameter tuned- alpha=0.1)

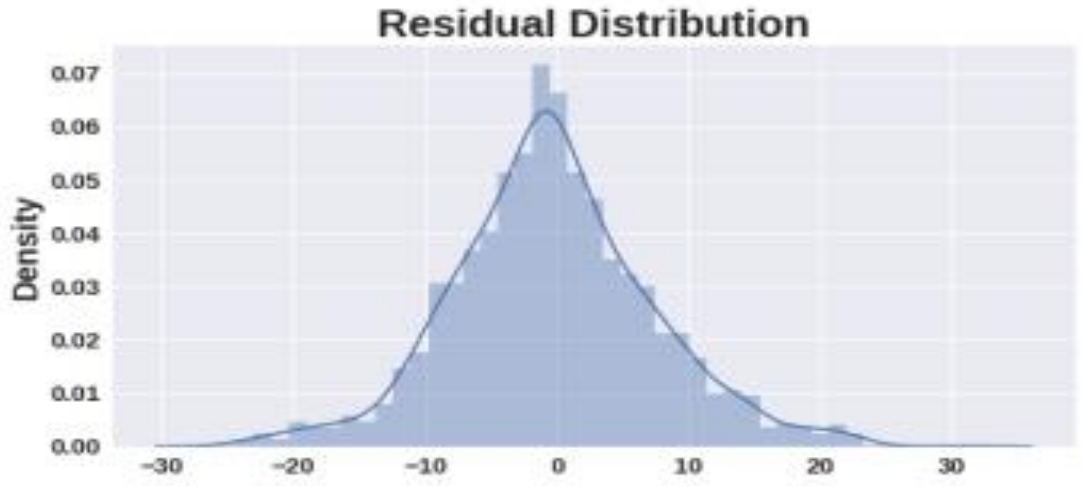
Scores on Train set

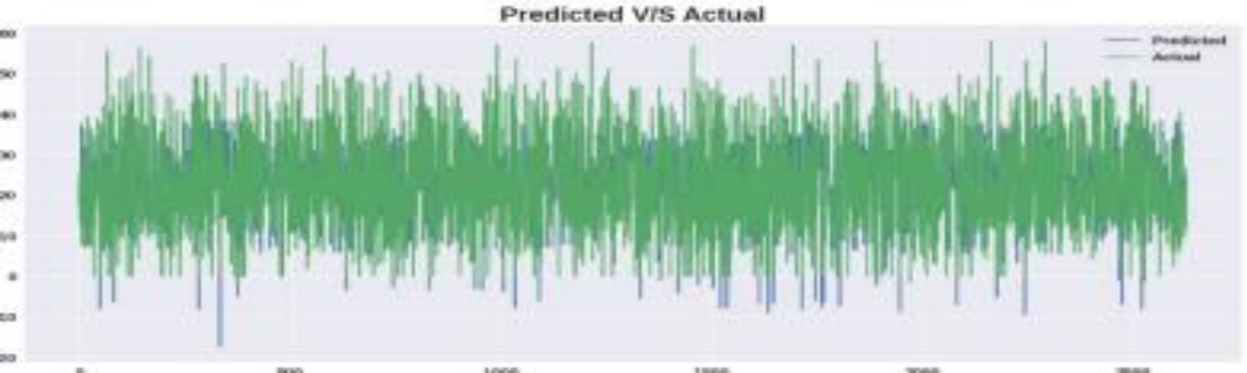
```
The Mean Absolute Error (MAE) is 5.869103531726283.
The Mean Squred Error(MSE) is 60.46402436494349.
The Root Mean Squared Error(RMSE) is 7.775861647749624.
The R2 Score is 0.6112950857219155.
```

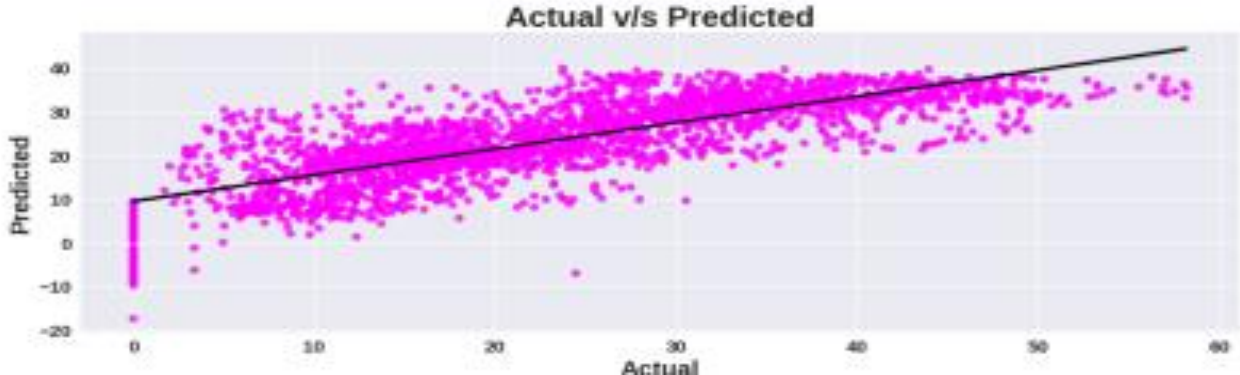
Scores on Test set

```
The Mean Absolute Error (MAE) is 5.850566426263689.
The Mean Squred Error(MSE) is 58.792684087499225.
The Root Mean Squared Error(RMSE) is 7.667638755673042.
The R2 Score is 0.61723035952942.
```









❑ Elastic Net Regression

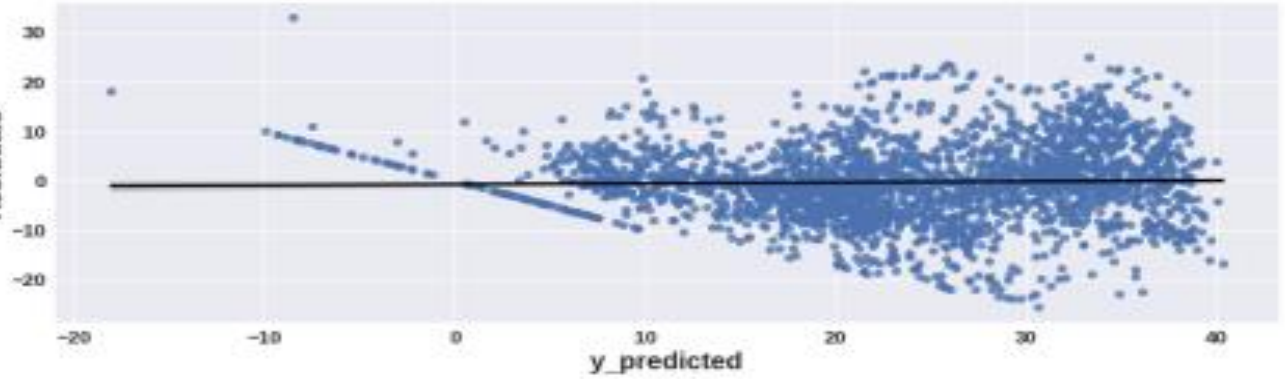
➢ **Elastic Net (Hyper-parameter tuned- alpha=0.001,l1_ratio=0.5)**

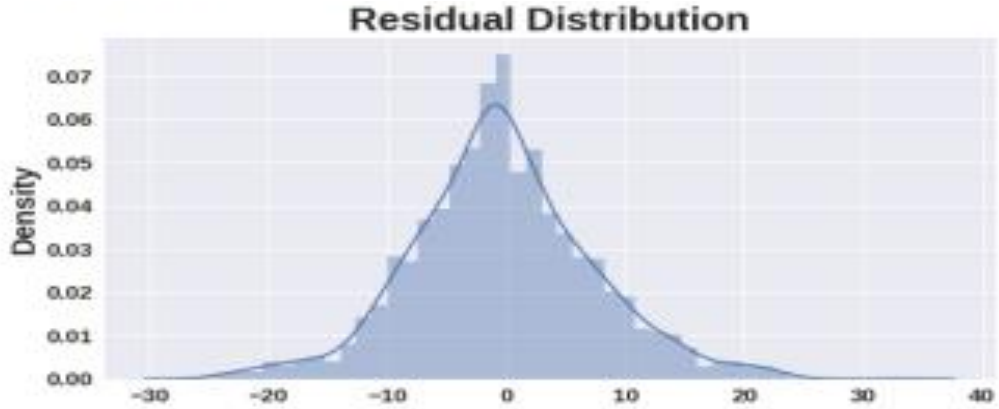
Scores on Train set

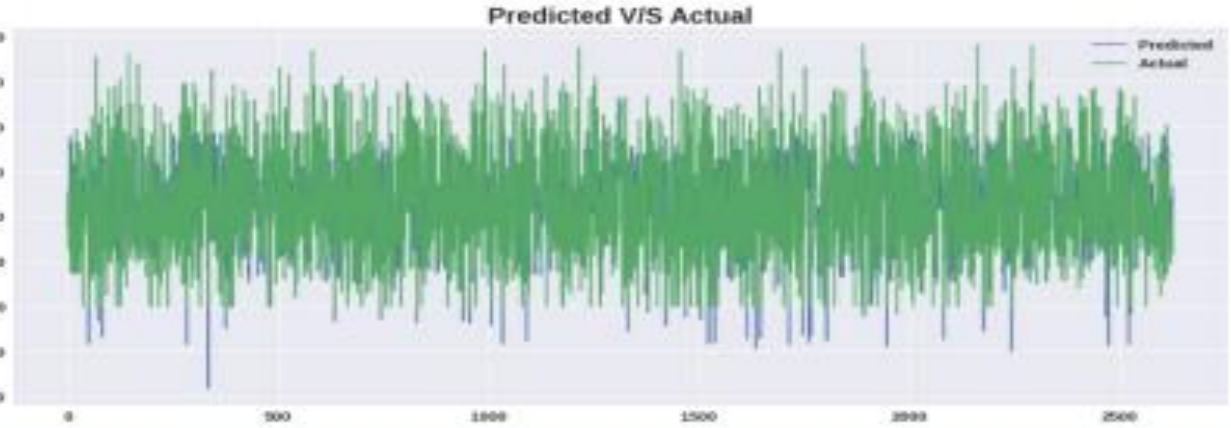
```
MSE  : 70.5576767228499 2
RMSE : 8.399861708555083
MAE  : 6.360933735604604
R2   : 0.5428222049186108
```

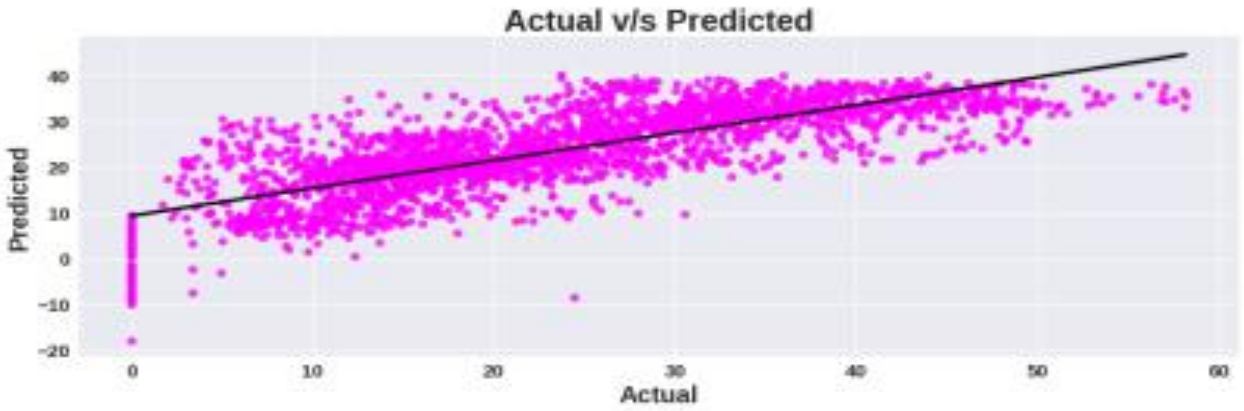
Scores on Test set

```
The Mean Absolute Error (MAE) is 6.5595008423322305.
The Mean Squred Error(MSE) is 74.2239434161807.
The Root Mean Squared Error(RMSE) is 8.615331880791402..
The R2 Score is 0.5286957430669388.
```









❑ Decision Tree

➢**Decision Tree regression(Hyper-parameter tuned- max_depth=9,max_features='auto')**

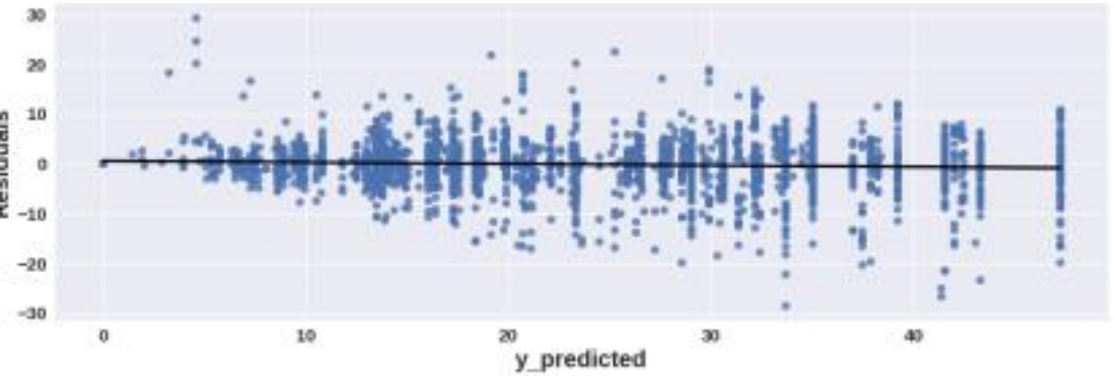
Scores on Train set

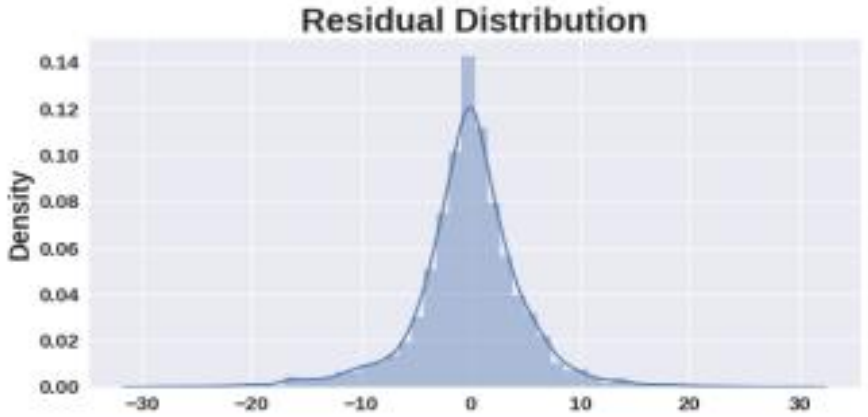
```
The Mean Absolute Error (MAE) is 2.8855165215690715.
The Mean Squred Error(MSE) is 18.444625087726916.
The Root Mean Squared Error(RMSE) is 4.294720606480347.
The R2 Score is 0.8814250872495163.
```

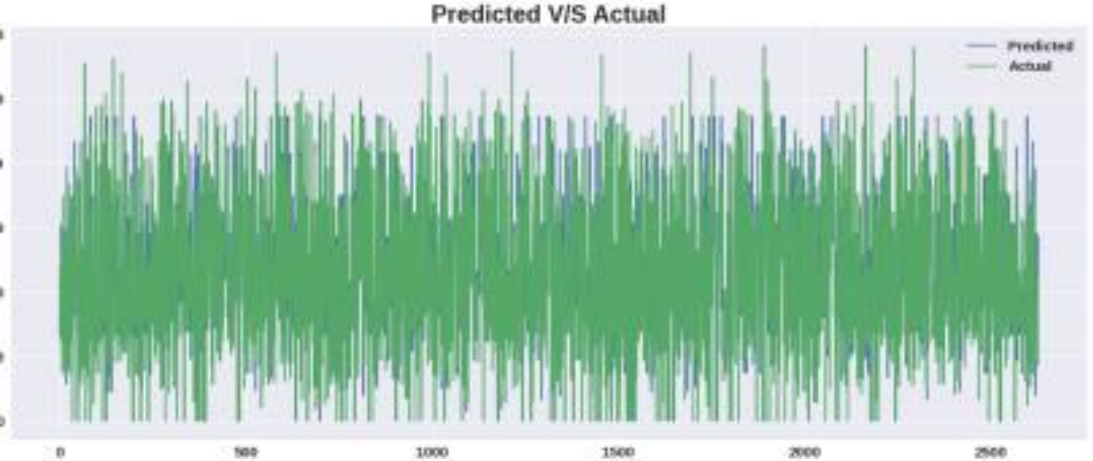
Scores on Test set

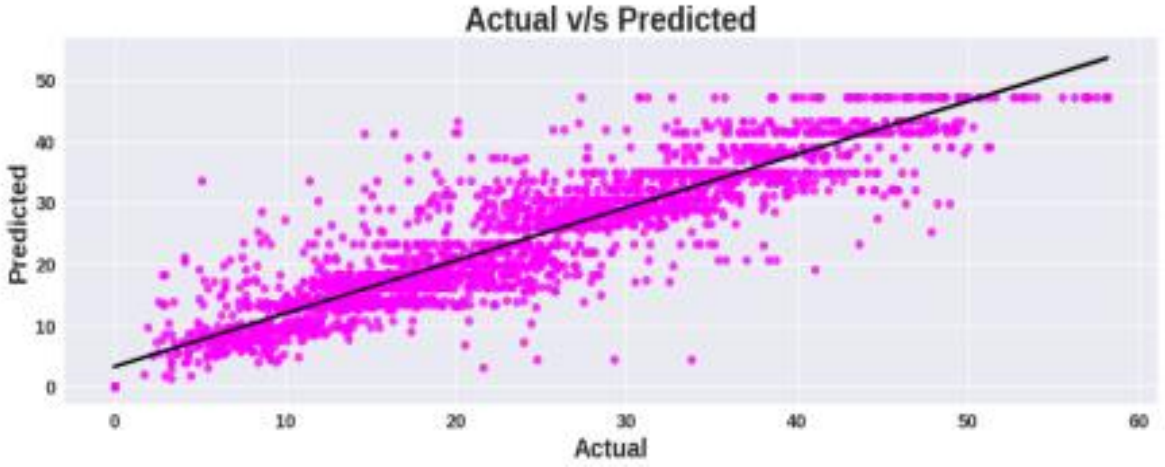
```
The Mean Absolute Error (MAE) is 3.3992026410244094.
The Mean Squred Error(MSE) is 24.910895604820194.
The Root Mean Squared Error(RMSE) is 4.9910816067081285.
The R2 Score is 0.8378176689421706.
```

The number of features to consider when looking for the best split









❑ Random Forest

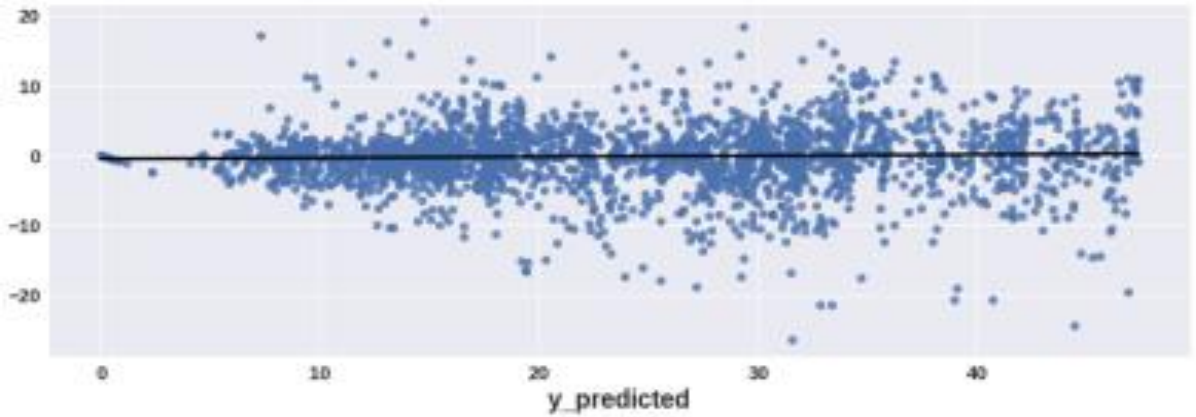
➢ **Random forest regression(Hyper-parameter tuned- 'max_depth': 9, 'n_estimators': 100')**

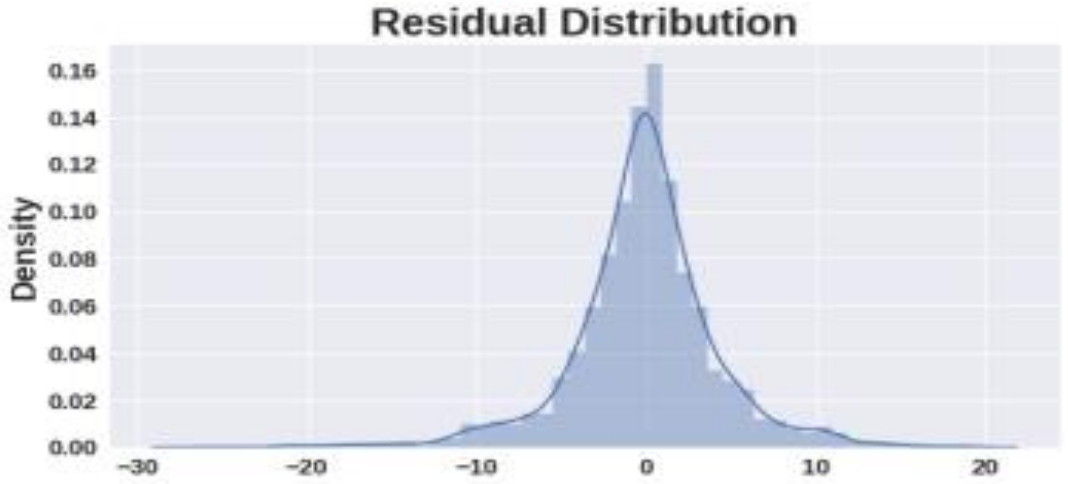
Scores on Train set

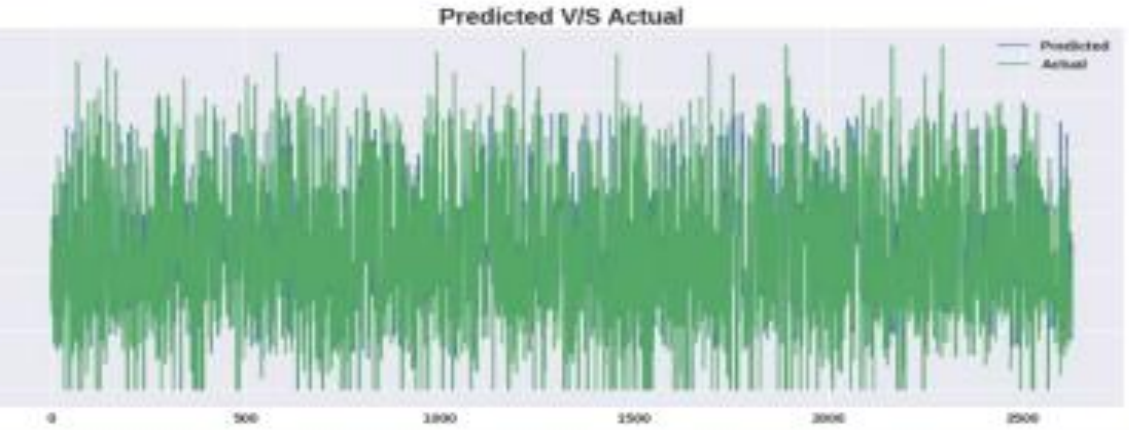
```
he Mean Absolute Error (MAE) is 2.6247545856850936.
he Mean Squred Error(MSE) is 14.905429807049964.
he Root Mean Squared Error(RMSE) is 3.8607550825000496.
he R2 Score is 0.904177502634334.
```

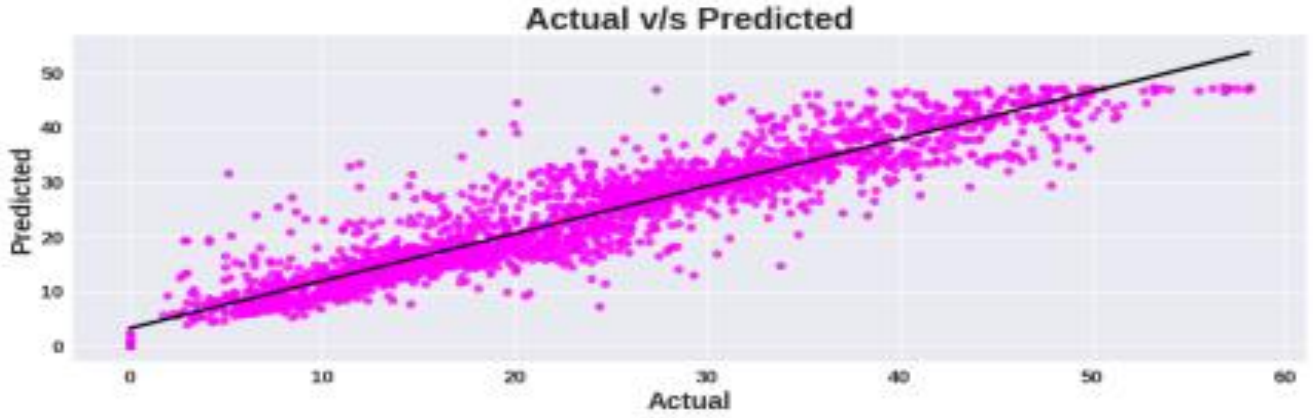
Scores on Test set

```
 The Mean Absolute Error (MAE) is 2.947737482949683.
 The Mean Squred Error(MSE) is 18.68756387544933.
The Root Mean Squared Error(RMSE) is 4.32291150446656.
The R2 Score is 0.8783346564815596.
```









❑ Gradient Boosting regression

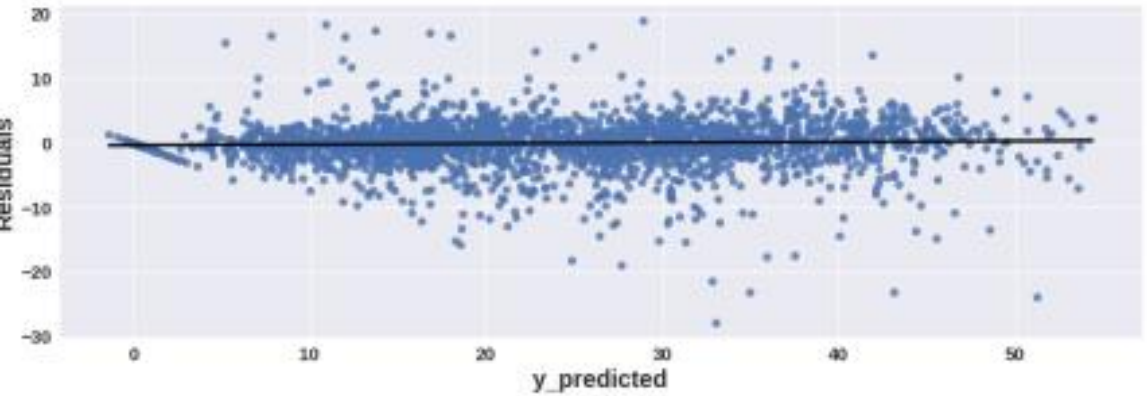
➢**Gradient boosting regression(Hyper-parameter tuned-** 'learning_rate': 0.04, 'max_depth': 8, 'n_estimators': 150, 'subsample': 0.9)

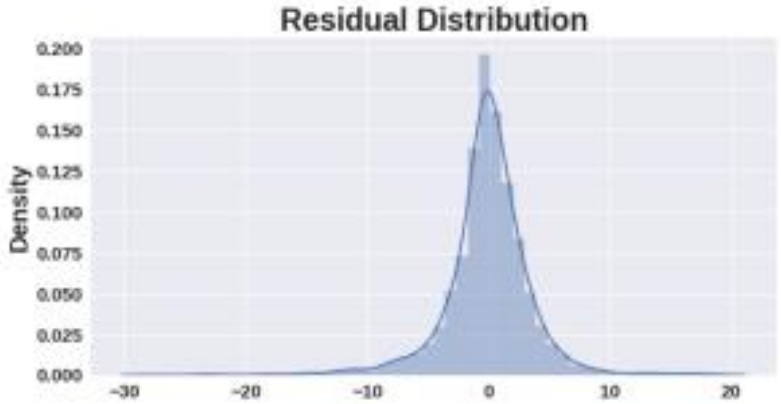
Scores on Train set

```
The Mean Absolute Error (MAE) is 1.5177256084968402.
The Mean Squred Error(MSE) is 4.794733133724516.
The Root Mean Squared Error(RMSE) is 2.1896879078363005.
The R2 Score is 0.9691761117241932.
```

Scores on Test set

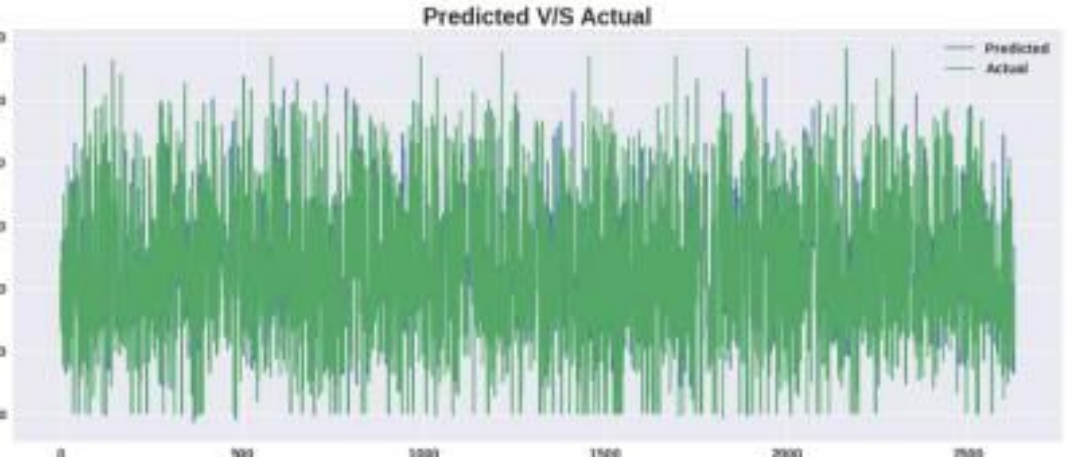
```
The Mean Absolute Error (MAE) is 2.3713827898940885.
The Mean Squred Error(MSE) is 13.217709880555015.
The Root Mean Squared Error(RMSE) is 3.635616850075791.
The R2 Score is 0.9139461288874848.
```

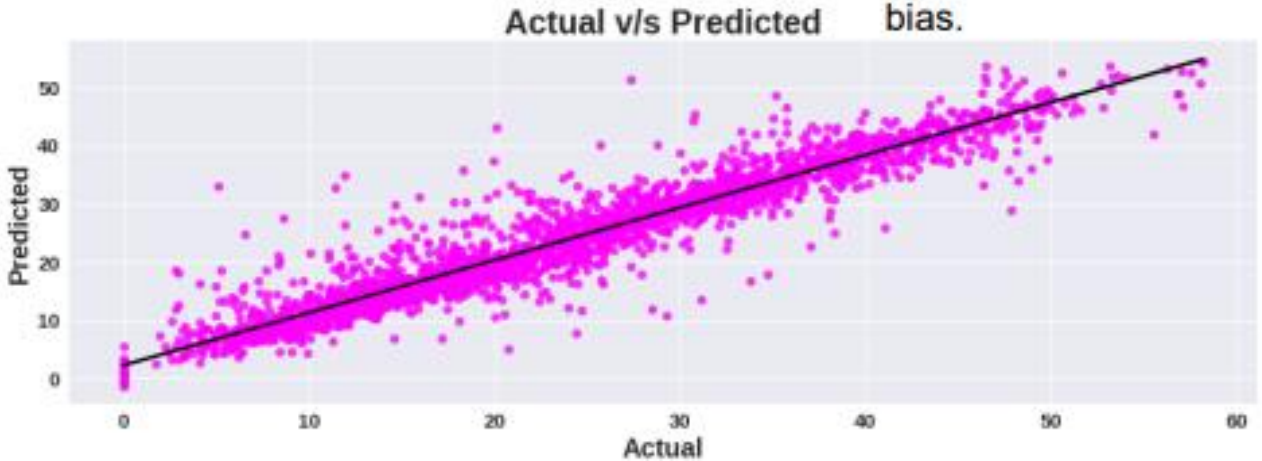




➢Learning rate shrinks the contribution of each tree by learning_rate. There is a trade-off between learning_rate and n_estimators.

➢Choosing subsample < 1.0 leads to a reduction of variance and an increase in bias.





Conclusion

		Model	MAE	MSE	RMSE	R2_score	Adjusted R2
Training set	0	Linear regression	5.821	59.387	7.706	0.615	0.61
	1	Lasso regression	7.006	87.275	9.342	0.434	0.43
	2	Ridge regression	5.823	59.361	7.705	0.615	0.61
	3	Elastic net regression Test	6.361	70.558	8.400	0.543	0.54
	4	Decision tree regression	3.333	23.604	4.858	0.847	0.85
	5	Random forest regression	0.778	1.658	1.288	0.989	0.99
	6	Gradient boosting regression	2.966	18.463	4.297	0.880	0.88
	7	Gradient Boosting gridsearchcv	1.765	7.818	2.796	0.949	0.95
Test set	0	Linear regression	5.868	59.345	7.704	0.623	0.62
	1	Lasso regression	7.299	94.395	9.716	0.401	0.40
	2	Ridge regression	5.883	59.581	7.719	0.622	0.62
	3	Elastic net regression Test	6.361	70.558	8.400	0.543	0.54
	4	Decision tree regression	3.487	25.871	5.086	0.836	0.83
	5	Random forest regression	2.094	11.248	3.354	0.929	0.93
	6	Gradient boosting regression	3.087	20.169	4.491	0.872	0.87
	7	Gradient Boosting gridsearchcv	2.138	11.397	3.376	0.928	0.93

As we have calculated MAE,MSE,RMSE and R2 score and Adjusted R2 for each model. Based on r2 score will decide our model performance.

Our assumption: if the difference of R2 score between Train data and Test is more than 10 % we will consider it as over fitting.

Linear, Lasso, Ridge and Elastic Net: Linear, Lasso, Ridge and Elastic regression models have below R2 scores(62%) on both training and test data.(Even after using GridsearchCV we have got similar results as of base models).

Decision Tree Regression: On Decision Tree Regressor model, without hyper -parameter tuning, we got r2 score as 100% on training data and on test data it was very less. Thus our model memorized the data. So it was a over fitted model. After hyper -parameter tuning we got r2 score as 85% on training data and 83% on test data which is quite good for us.

Random Forest: On Random Forest Regressor model, without hyper -parameter tuning we got r2 score as 98% on training data and 90% on test data. Thus our model memorized the data. So it was a over fitted model, as per our assumption After hyper -parameter tuning we got r2 score as 98% on training data and 92% on test data which is very good for us.

Gradient Boosting Regression(GridSearchCV): On Gradient Boosting GridSearchCV model, without hyper -parameter tuning we got r2 score as 88% on training data and 87% on test data. Our model performed well without hyper -parameter tuning. After hyper -parameter tuning we got r2 score as 95% on training data and 93% on test data, thus we improved the model performance by hyper -parameter tuning.

THANK YOU