Capstone Project - 2

BIKE SHARING DEMAND PREDICTION

BY

RUSHIKESH ARJUN MANE

(Cohort Seattle)

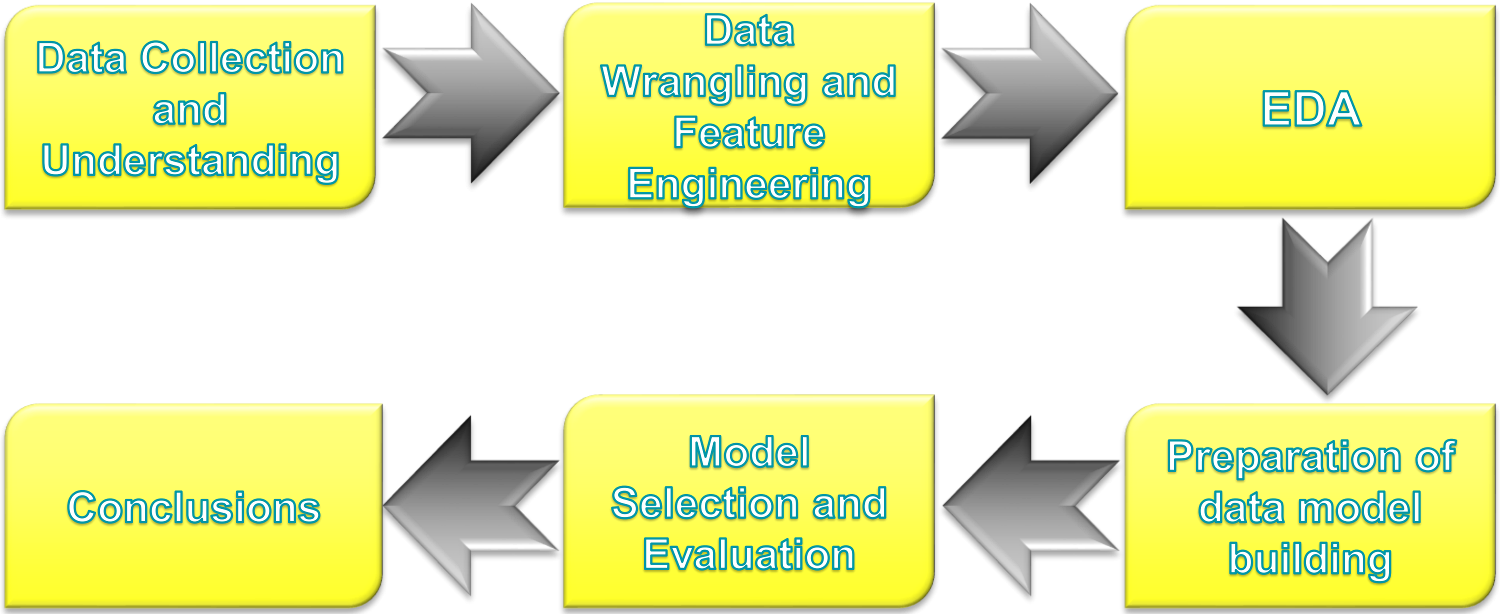
Contain

1. **Problem Statement**
2. **Work Flow**
3. **Data Collection and Understanding**
4. **Data Wrangling and Feature Engineering**
5. **EDA (Exploratory Data Analysis)**
6. **Preparation of Data for Model Building**
7. **Model Selection and Evaluation**
8. **Conclusion**

Problem Statement

- Currently Rental bikes are introduced in many urban cities for the enhancement of mobility comfort. The client is Seoul Bike, which participates in a bike share program in Seoul, South Korea. An accurate prediction of bike count is critical to the success of the Seoul bike share program. 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 final aim of this project is the prediction of bike count required at each hour for the stable supply of rental bikes

Work Flow


Data Collection and Understanding
Data Wrangling and Feature Engineering
EDA
Preparation of data model building
Model Selection and Evaluation
Conclusions

Data Collection and Understanding

- For analysis and model building we are have the Seoul Bike Data.
- The Dataset contain 8760 rows of observations and 14 attributes.
- **DATA DESCRIPTION** :

  Date : year-month-day.

  Hour - Hour of he day.

  Temperature-Temperature in Celsius.

  Humidity - %.

  Wind speed - m/s.

  Visibility - m.

  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).
- **Rented Bike count** - Count of bikes rented at each hour (Target Variable i.e Y variable)

- ❑ **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 .

Data Wrangling and Feature Engineering

Here we renamed columns name because they had units mentioned in it.

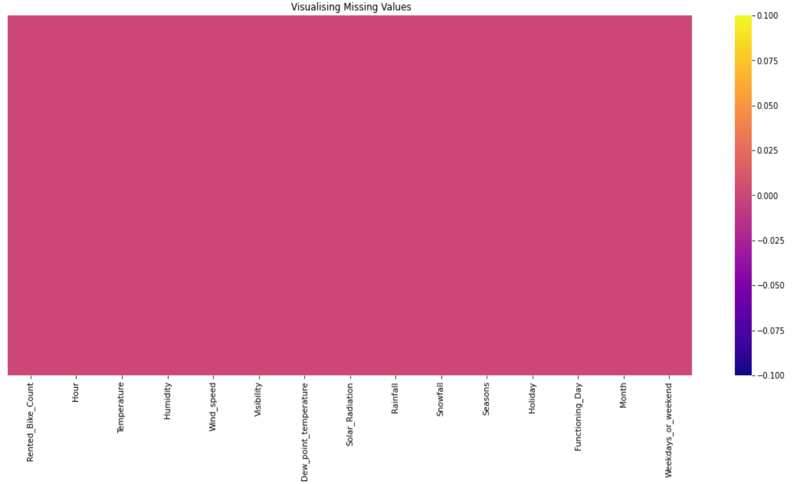
```
[12] ## Checking columns

     df_main.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')
```

```
[14] #Since the variables having units with name, renaming columns for better variable analysis.
     df_main.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'},inplace=True)
```

- In dataset we don’t have any missing or null value.
- We have zero duplicates.
- For feature engineering we changed data type of Date column form ‘object’ to ‘datetime64[ns]’.
- We creating new columns ‘Day’ and ‘Month’ form Data for further EDA.

```
## Visulaizing null or missing values using heatmap.
plt.figure(figsize=(20,8))
sns.heatmap(df_main.isnull(),cmap='plasma',annot=False,yticklabels=False)
plt.title(" Visualising Missing Values")
```

```
Text(0.5, 1.0, ' Visualising Missing Values')
```




```
[20] ## Checking Null or Missing values form our dataset.
     df_main.isnull().sum()
```

```
Rented_Bike_Count          0
Hour                       0
Temperature                0
Humidity                   0
Wind_speed                 0
Visibility                 0
Dew_point_temperature      0
Solar_Radiation            0
Rainfall                   0
Snowfall                   0
Seasons                    0
Holiday                    0
Functioning_Day            0
Month                      0
Weekdays_or_weekend        0
dtype: int64
```

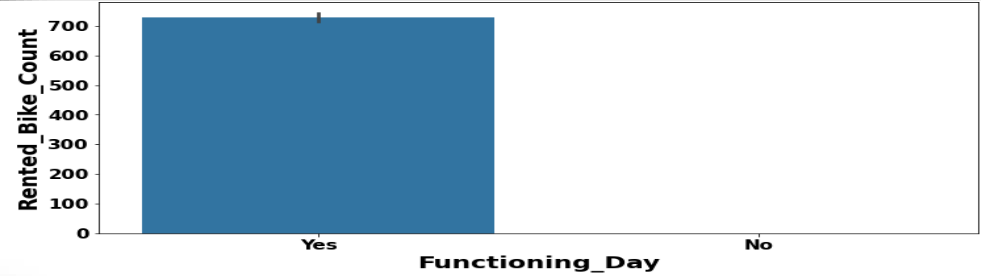
```
## Checking Duplicate rows in our Dataset.
duplicates=df_main.duplicated().sum()
print(f"We are haveing {duplicates} rows in our Dataframe.")
```

```
We are haveing 0 rows in our Dataframe.
```

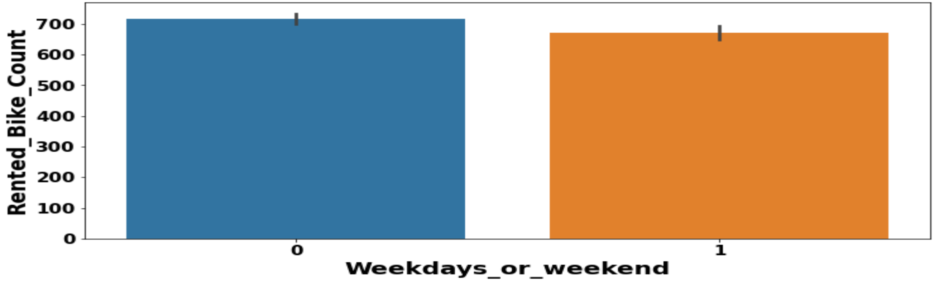
```
## For further analysis of bike rentals with months,year and day. Changeing the datatype of Date column to extract 'Month' ,'Day', "Year".
df_main['Date']=df_main['Date'].astype('datetime64[ns]')
```

EDA

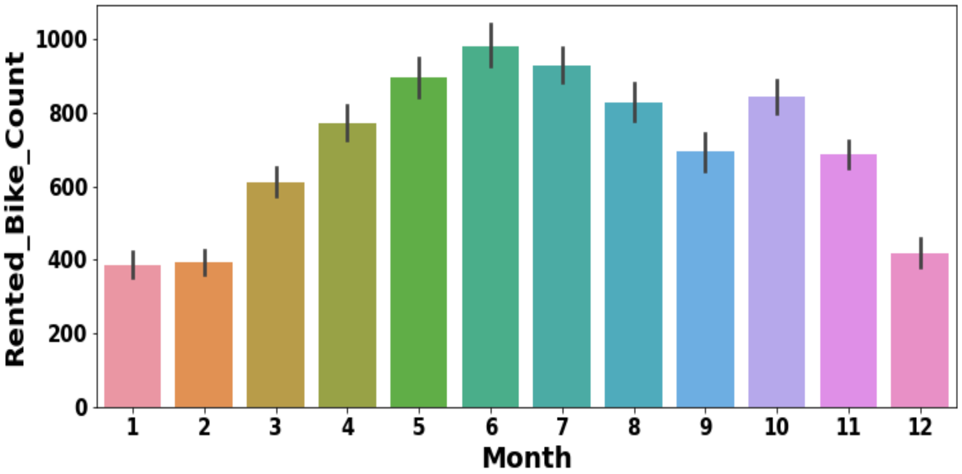
- The bikers rented high numbers of bike on No Holidays. Which is 700 bikes.



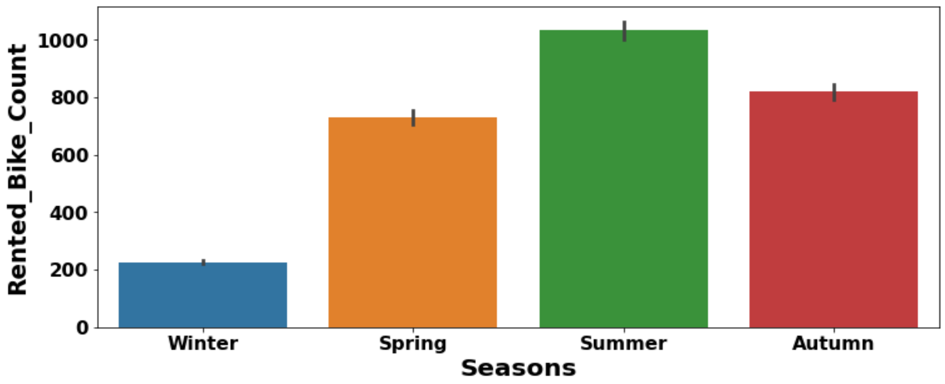
- There are no bike rented on no functioning day and more bikes are rented on functioning day i.e. 700 bikes.



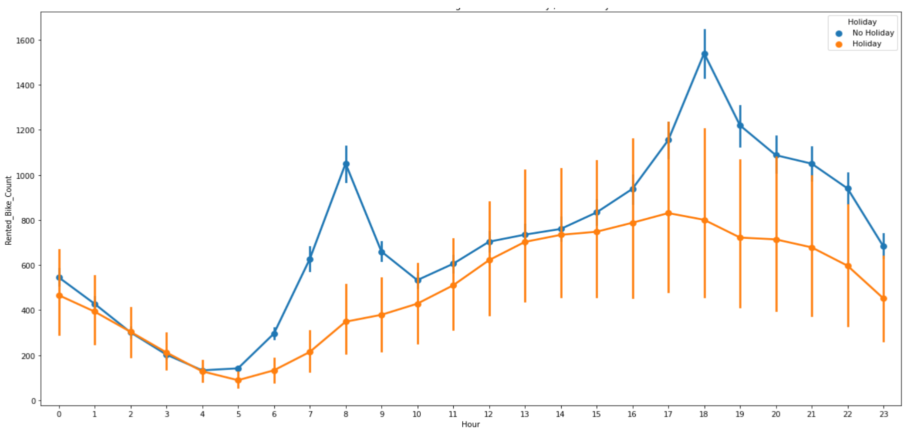
- On weekdays 650 bikes were rented and more than 700 bikes were rented on weekdays.



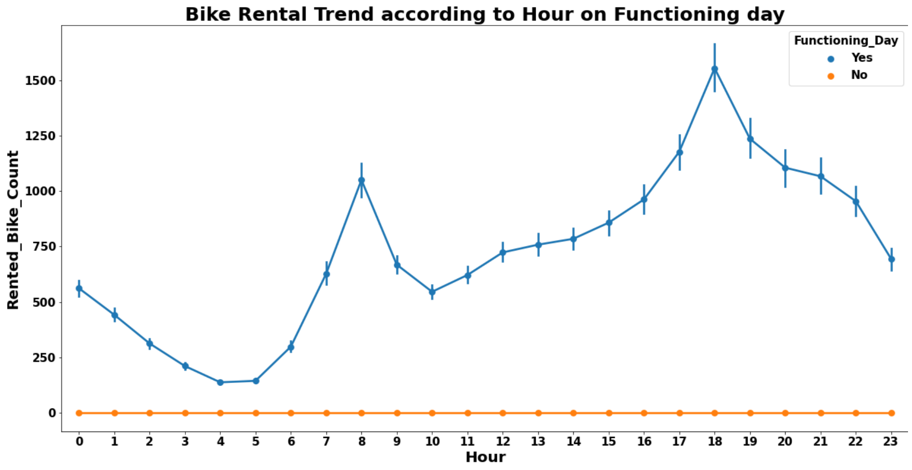
From graph we can see bike rent count starts increasing form March till June.



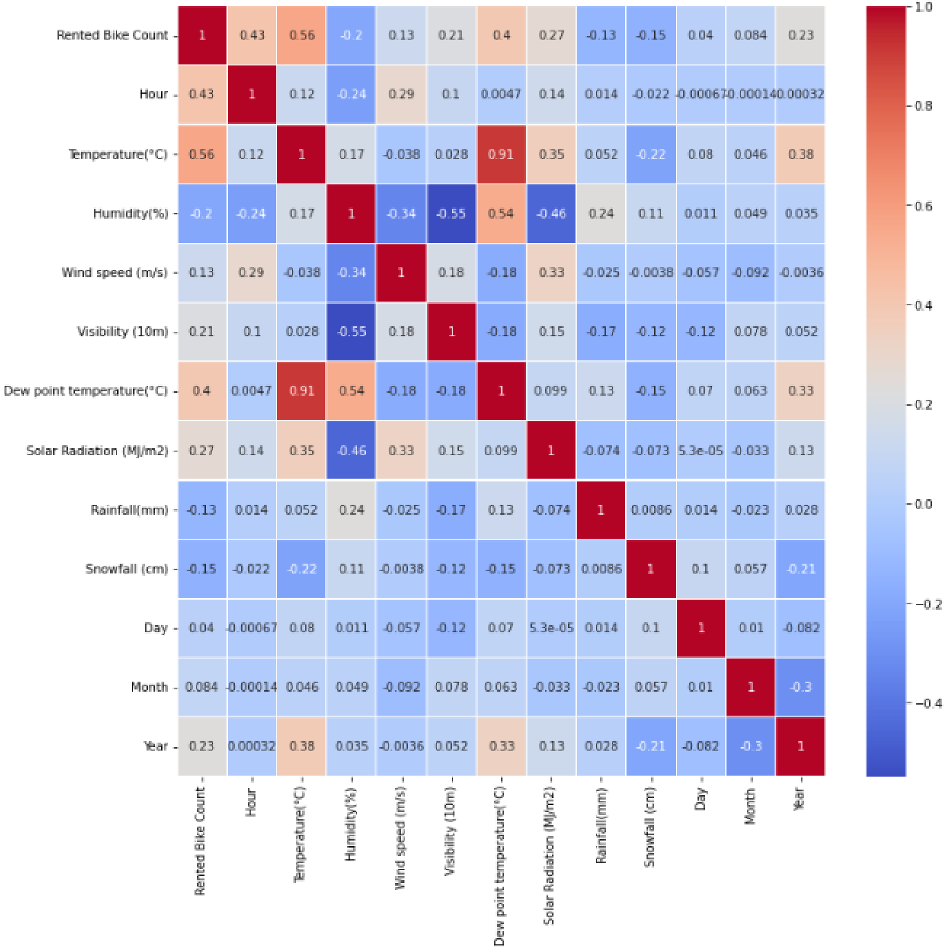
Form the graph we can see bikers rentals bike more in the summer season and the rentals are less in winters.



- We can see there is sudden peak between 4 AM to 7 AM for bike rentals in no holidays. But in on holidays the bike rentals are very less compare to no holidays.



- 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.



- **From the heatmap we can see that the temperature and dew point temperature are highly correlated. So we have to drop because that having very low correlation wfor the dew point temperature ith the target variable compare to the temperature.**

Preparation of data for model building

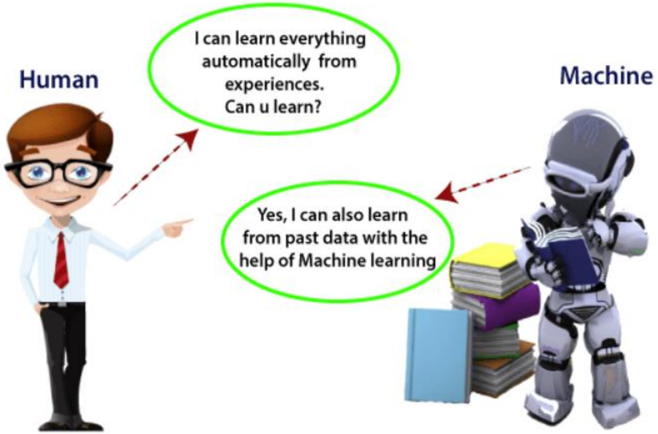
```
[ ] # Createing dummy variables
    df=pd.get_dummies(df,columns=['Seasons'],prefix='Seasons',drop_first=True)
```

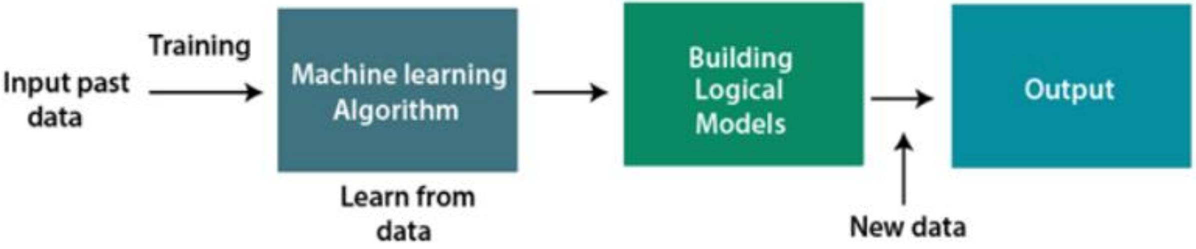
```
[ ] # Labeling for holiday=1 and no holiday=0
    df['Holiday']=df['Holiday'].map({'No Holiday':0, 'Holiday':1})
```

```
[ ] # Labeling for Yes=1 and no No=0
    df['Functioning_Day']=df['Functioning_Day'].map({'Yes':1, 'No':0})
```

- By using variation inflation factor we dropped ‘Visibility’ and ‘Humidity’ features as they had VIF value more than 5.
- For further modelling we created dummy variables for categorical Seasons columns and did mapping with 0 and 1.
- So we prepare our data for model building.

ML Algorithms





Model Selection and Evaluation

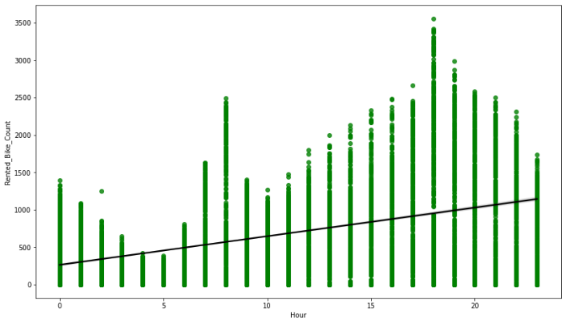
As this is the regression problem we are trying to predict continuous value. For this we used following regression models.

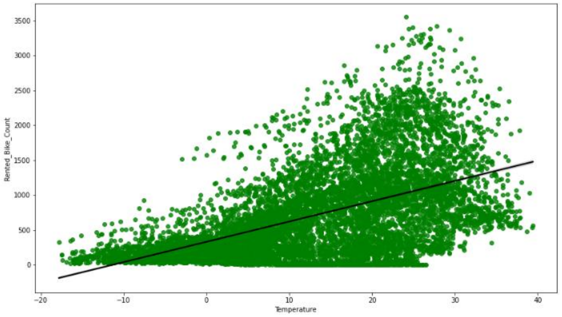
- Linear Regression
- Lasso Regression (regularized regression)
- Ridge Regression (regularized regression)
- Elastic Net regression
- Decision Tree Regression
- Random Forest Regression
- Gradient Boosting Regression

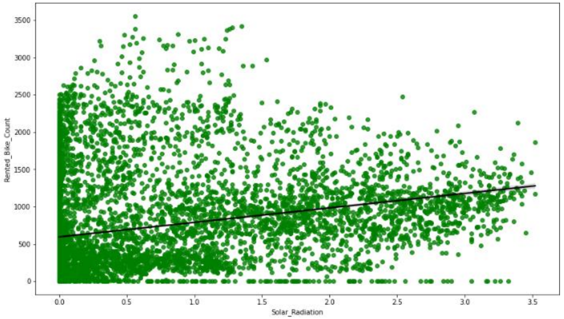
Assumptions of regression line:

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.

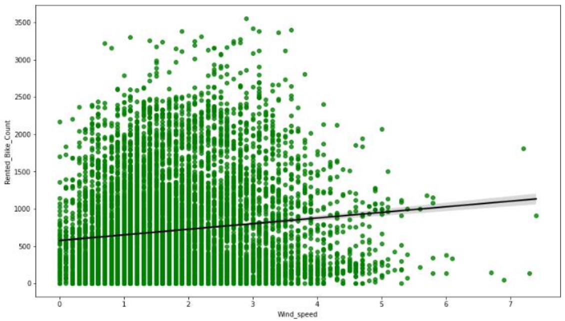
❖ Before and after applying these models we checked our regression assumption by distribution of residuals, scatter plot of actual and predicted values, removing multi-collinearity among independent variables.

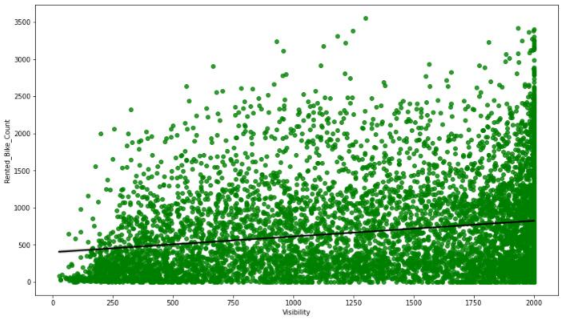


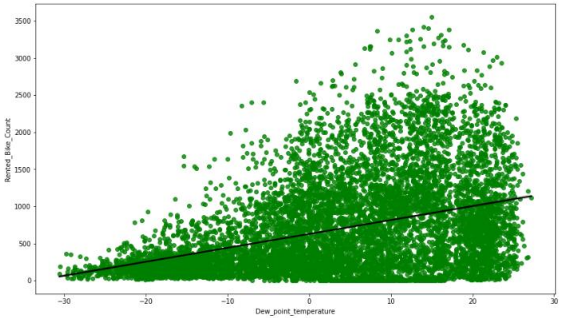


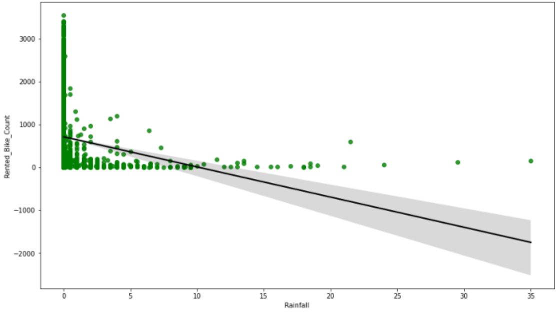


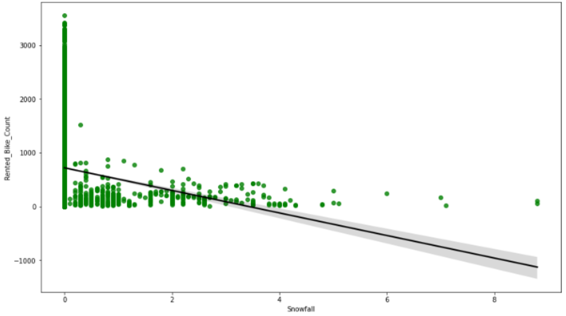
- **From the above regression plot of all numerical features we see that the columns 'Temperature', 'Wind_speed', 'Visibility', 'Dew_point_temperature', 'Solar_Radiation' are positively relation to the target variable,which means the rented bike count increases with increase of these features.**

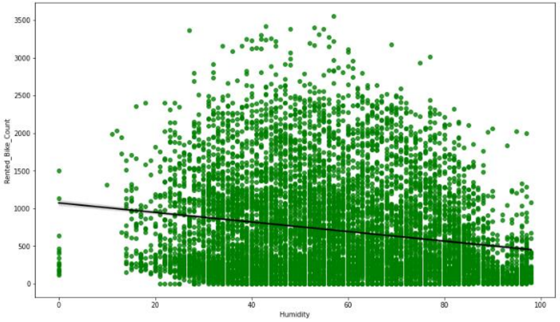










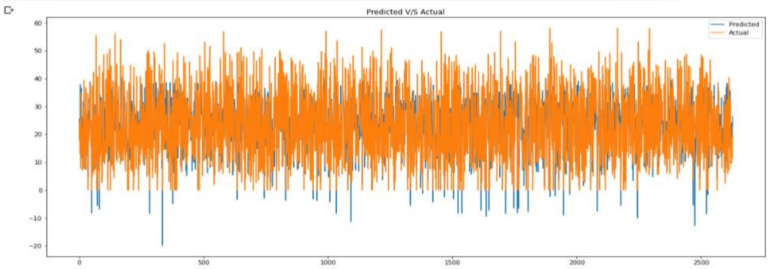


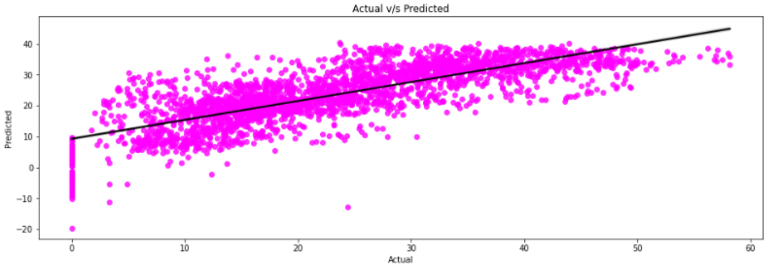
- 'Rainfall',‘ Snowfall’, 'Humidity' these features are negatively related with the target variable which means the rented bike count decreases when these features increase.

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.
```

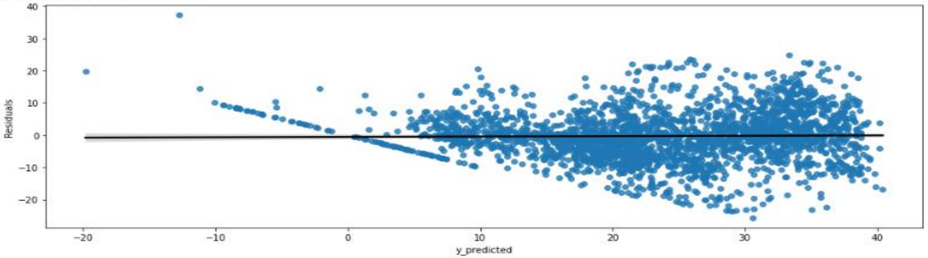


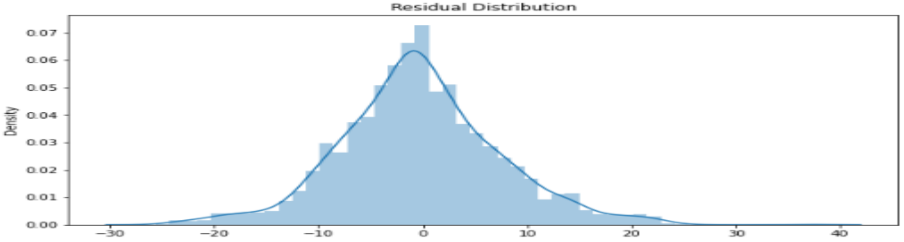


Score on Test set

```
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 0.618326967365199.
```

```
Text(0.5, 0, 'y_predicted')
```





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”.

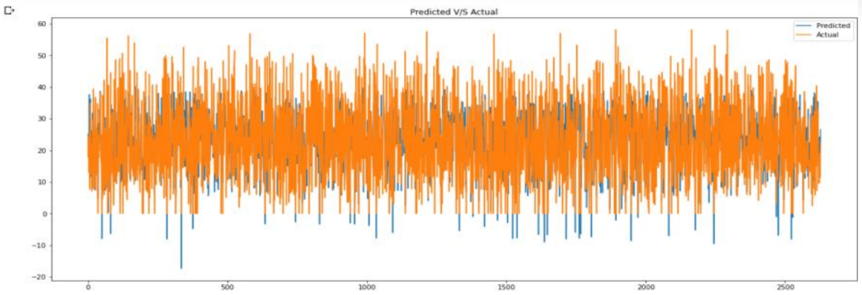
Lasso (Hyper-parameter tuned-alpha=0.1)

Scores on Train set

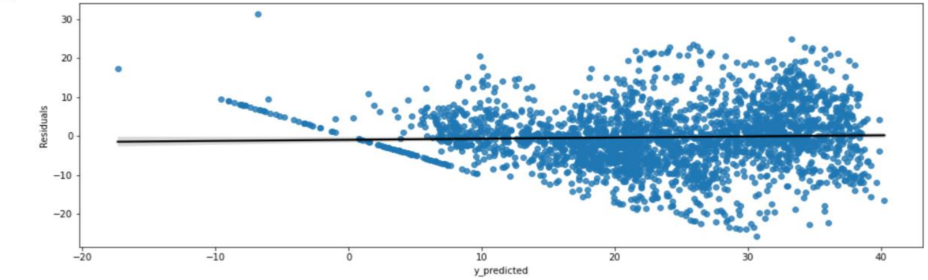
```
The Mean Absolute Error (MAE) is 5.869103531726283.
The Mean Squred Error(MSE) is 60.46402436494349.
The Root Mean Squared Error(RMSE) is 7.775861647749624.
The R2 Score is 0.6112950857219155.
```

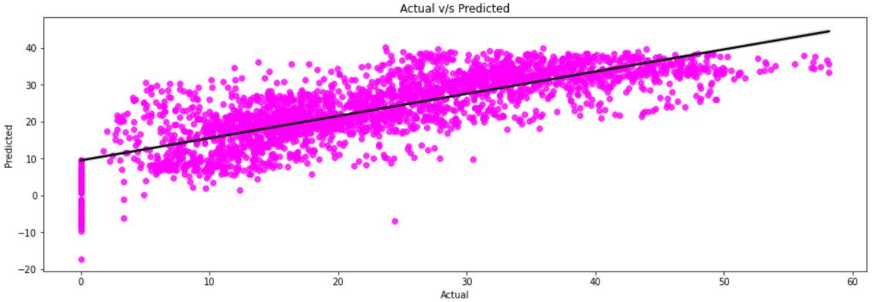
Score on Test set

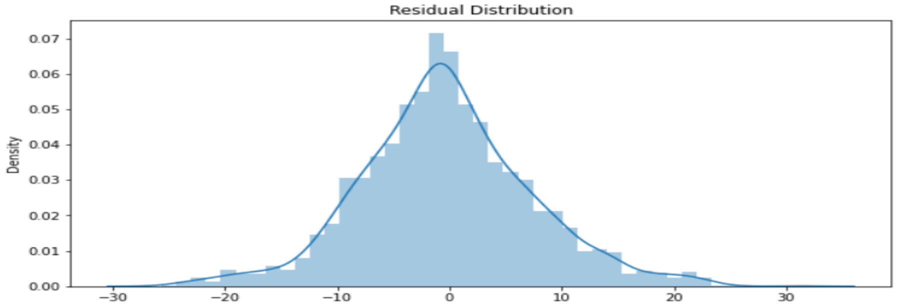
```
The Mean Absolute Error (MAE) is 5.850566426263689.
The Mean Squred Error(MSE) is 58.792684087499225.
The Root Mean Squared Error(RMSE) is 7.667638755673042.
The R2 Score is 0.61723035952942.
```




```
Text(0.5, 0, 'y_predicted')
```







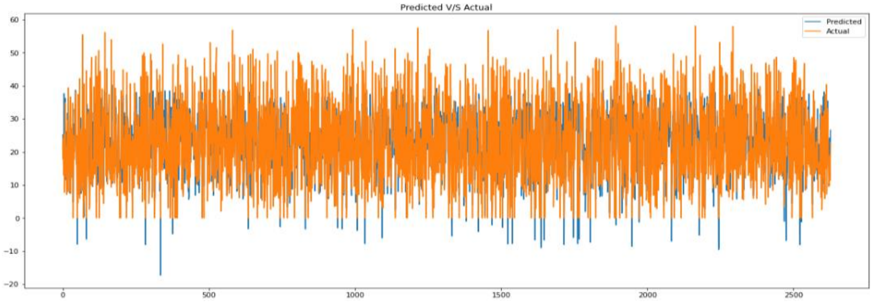
⌘Ridge (Hyper-parameter tuned-alpha=0.1)

Scores on Train set

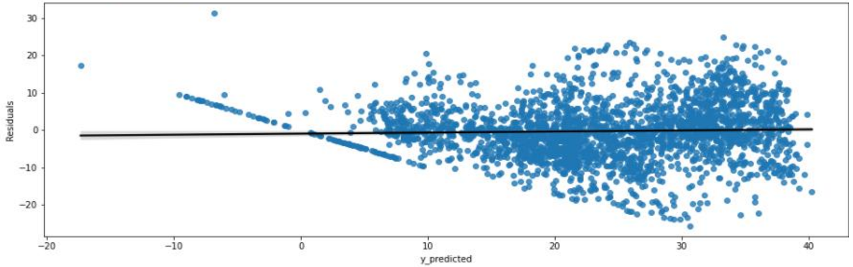
```
The Mean Absolute Error (MAE) is 5.869103531726283.
The Mean Squred Error(MSE) is 60.46402436494349.
The Root Mean Squared Error(RMSE) is 7.775861647749624.
The R2 Score is 0.6112950857219155.
```

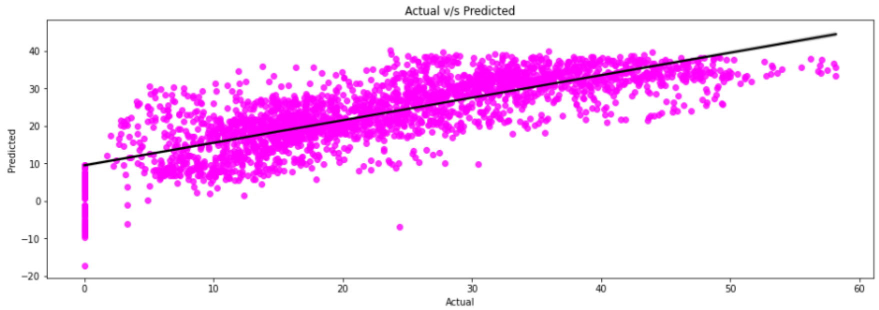
Score on Test set

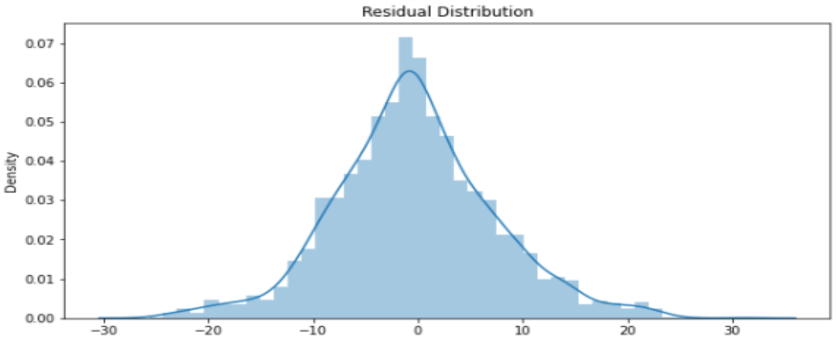
```
The Mean Absolute Error (MAE) is 5.850566426263689.
The Mean Squred Error(MSE) is 58.792684087499225.
The Root Mean Squared Error(RMSE) is 7.667638755673042.
The R2 Score is 0.61723035952942.
```




```
Text(0.5, 0, 'y_predicted')
```



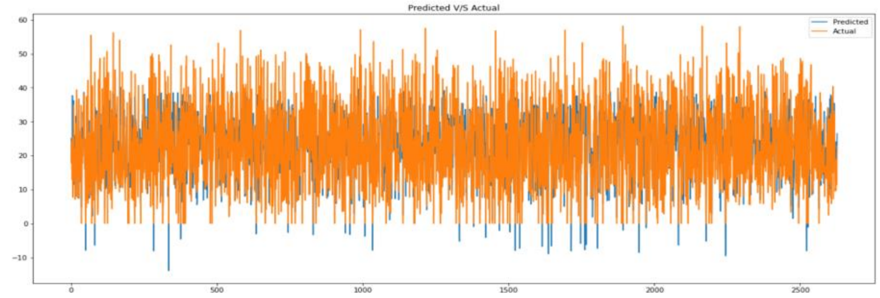


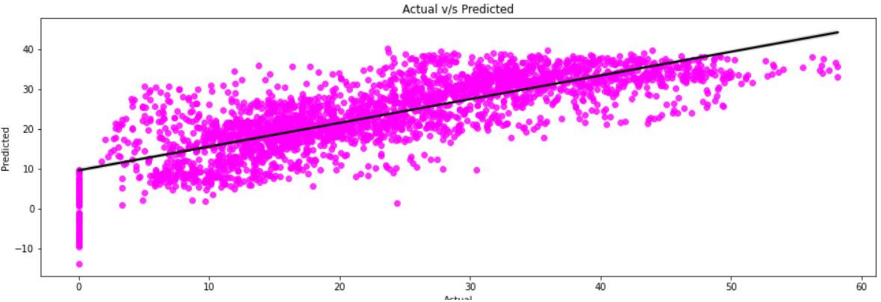


Elastic Net (Hyper-parameter tuned-alpha=0.01,l1_ratio=0.05)

Scores on Train set

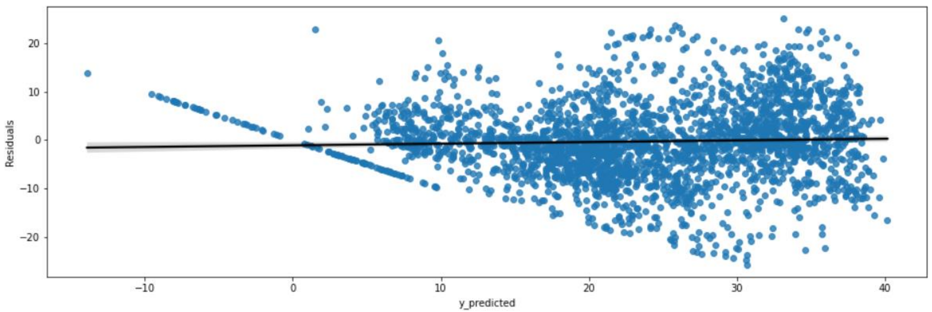
```
The Mean Absolute Error (MAE) is 5.8932275545714745.
The Mean Squred Error(MSE) is 60.90273656811195.
The Root Mean Squared Error(RMSE) is 7.804020538678249.
The R2 Score is 0.6084747377362095.
```

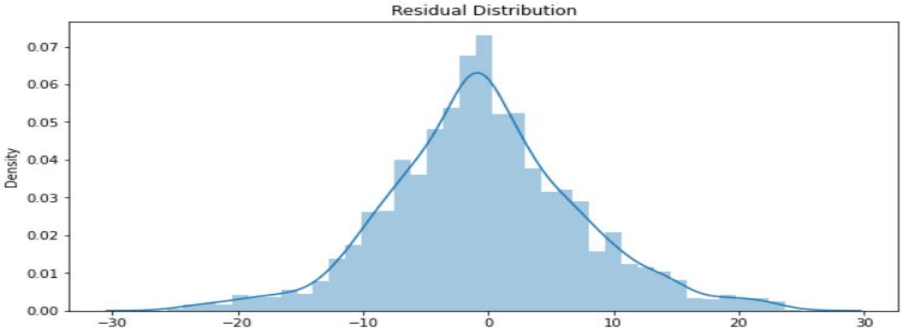




Score on Test set

```
The Mean Absolute Error (MAE) is 5.871068266349744.
The Mean Squred Error(MSE) is 59.2908889405223.
The Root Mean Squared Error(RMSE) is 7.700057723194178.
The R2 Score is 0.6139867979293316.
```





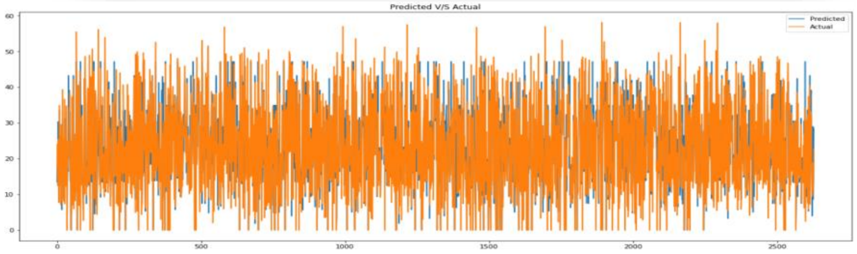
Decision Tree regression(Hyper-parameter tuned max_depth=9,max_features=‘auto’

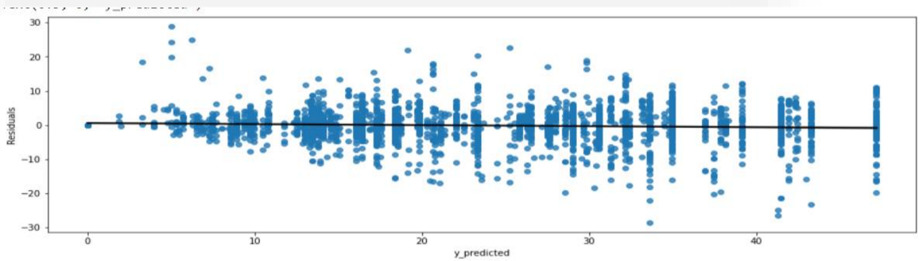
Scores on Train set

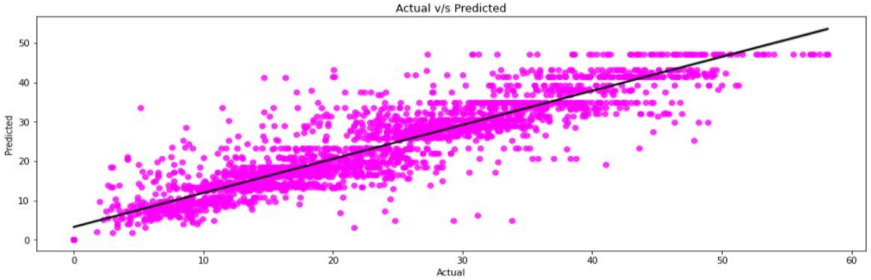
The Mean Absolute Error (MAE) is 2.88551652156907.
The Mean Squred Error(MSE) is 18.444625087726916.
The Root Mean Squared Error(RMSE) is 4.294720606480347.
The R2 Score is 0.8814250872495163.

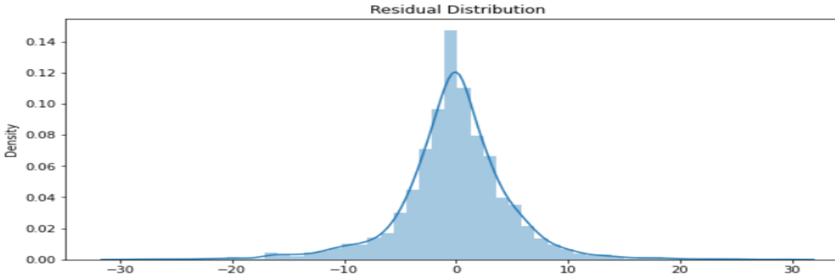
Score on Test set

The Mean Absolute Error (MAE) is 3.4014863276845166.
The Mean Squred Error(MSE) is 24.99357090624535.
The Root Mean Squared Error(RMSE) is 4.9993570492859725.
The R2 Score is 0.8372794115740394.









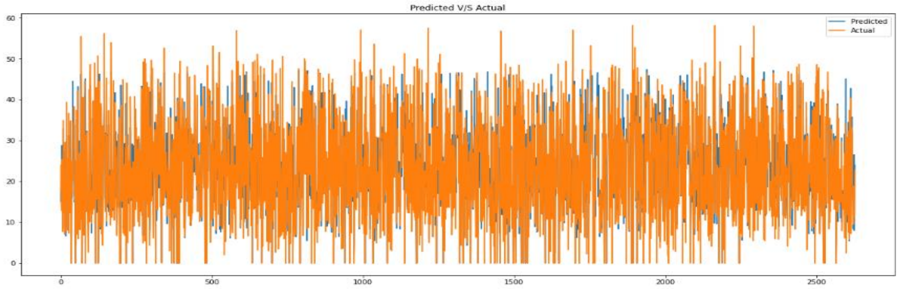
Random forest regression(Hyper-parameter tuned- 'max_depth': 9,'n_estimators':100')

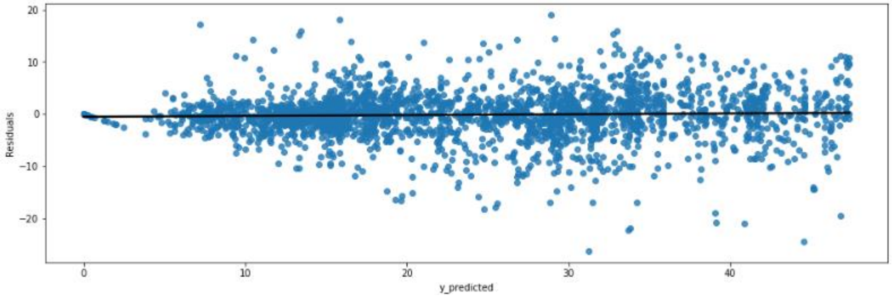
Scores on Train set

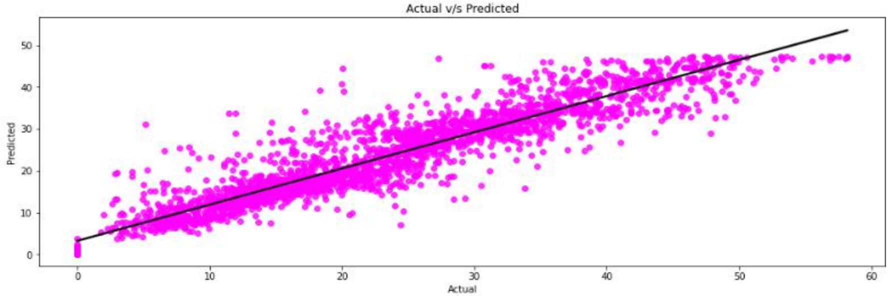
```
The Mean Absolute Error (MAE) is 2.622693501337842.
The Mean Squred Error(MSE) is 14.900275581749467.
The Root Mean Squared Error(RMSE) is 3.8600875095973493.
The R2 Score is 0.904210637588956.
```

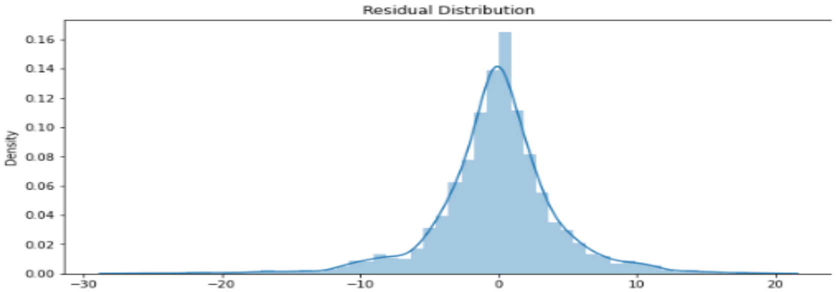
Score on Test set

```
The Mean Absolute Error (MAE) is 2.949708958107071.
The Mean Squred Error(MSE) is 18.768067299650596.
The Root Mean Squared Error(RMSE) is 4.332212748659811.
The R2 Score is 0.8778105391153188.
```









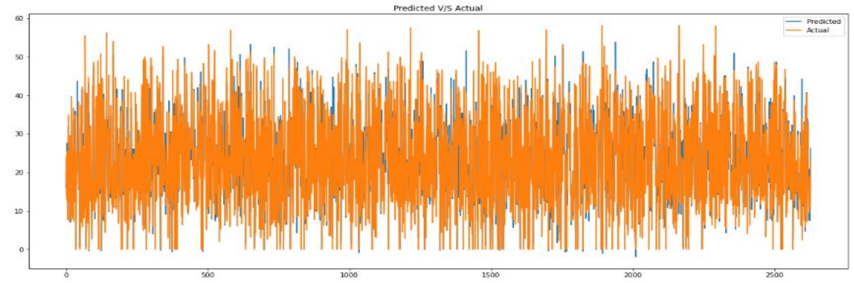
Gradient boosting regression(Hyper-parameter tuned- 'learning_rate': 0.04, 'max_depth': 8, 'n_estimators': 150, 'subsample': 0.9)

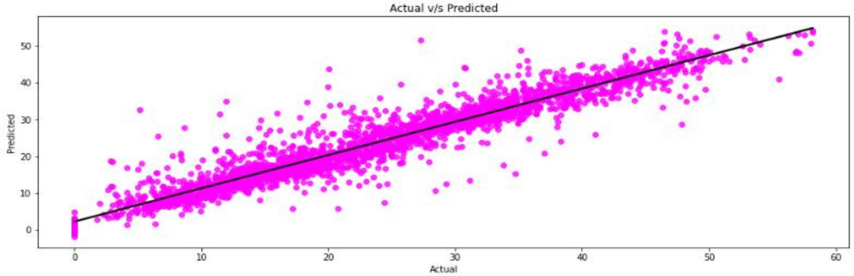
Scores on Train set

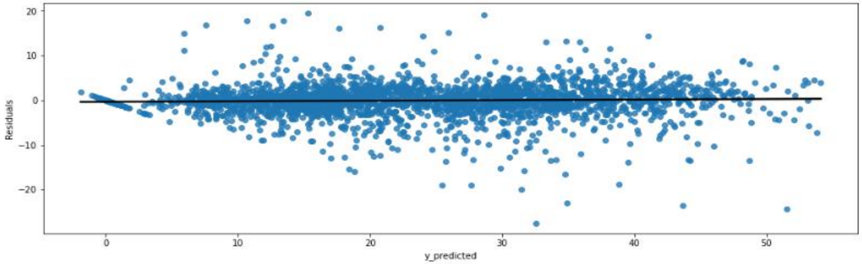
```
The Mean Absolute Error (MAE) is 1.5156506925891629.
The Mean Squred Error(MSE) is 4.783768452278599.
The Root Mean Squared Error(RMSE) is 2.1871827660894274.
The R2 Score is 0.9692466003429427.
```

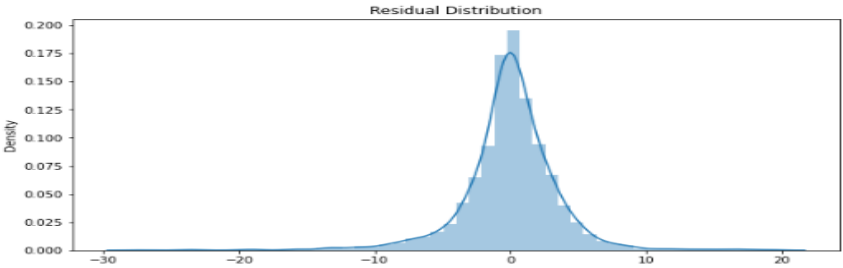
Score on Test set

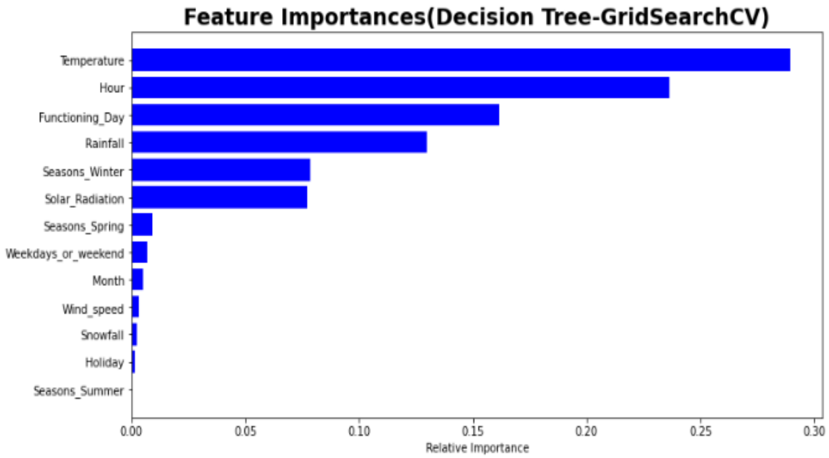
```
The Mean Absolute Error (MAE) is 2.369545110259515.
The Mean Squred Error(MSE) is 13.197128133406741.
The Root Mean Squared Error(RMSE) is 3.63278517578548.
The R2 Score is 0.914080126307036.
```

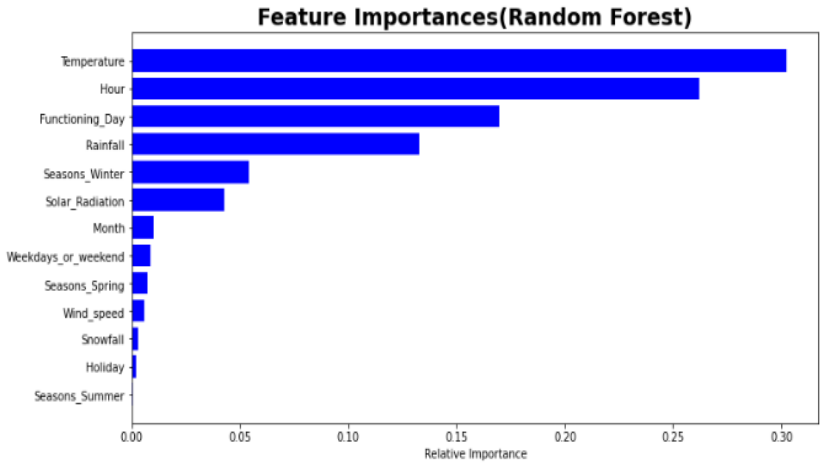


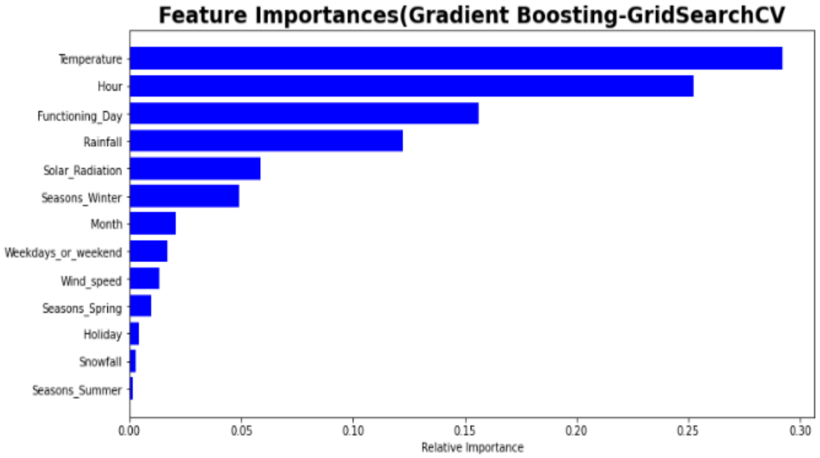







Feature Importances(Decision Tree-GridSearchCV)
Temperature
Hour
Functioning_Day
Rainfall
Seasons_Winter
Solar_Radiation
Seasons_Spring
Weekdays_or_weekend
Month
Wind_speed
Snowfall
Holiday
Seasons_Summer
0.00
0.05
0.10
0.15
0.20
0.25
0.30
Relative Importance


Feature Importances(Random Forest)
Temperature
Hour
Functioning_Day
Rainfall
Seasons_Winter
Solar_Radiation
Month
Weekdays_or_weekend
Seasons_Spring
Wind_speed
Snowfall
Holiday
Seasons_Summer
0.00
0.05
0.10
0.15
0.20
0.25
0.30
Relative Importance


Feature Importances(Gradient Boosting-GridSearchCV
Temperature
Hour
Functioning_Day
Rainfall
Solar_Radiation
Seasons_Winter
Month
Weekdays_or_weekend
Wind_speed
Seasons_Spring
Holiday
Snowfall
Seasons_Summer
0.00
0.05
0.10
0.15
0.20
0.25
0.30
Relative Importance

		Model	MAE	MSE	RMSE	R2_score
Training set	0	Linear Regression	5.8555	60.2995	7.7653	0.6124
	1	Lasoo	5.8691	60.4640	7.7759	0.6113
	2	Ridge GridSearchCV	5.8691	60.4640	7.7759	0.6113
	3	ElasticNet(GridSearchCV-Tunned)	5.8932	60.9027	7.8040	0.6085
	4	Decision Tree Regressor-GridSearchCV	2.8855	18.4446	4.2947	0.8814
	5	Random Forest	0.9375	2.1370	1.4618	0.9863
	6	Random Forest-GridSearchCv	2.6227	14.9003	3.8601	0.9042
	7	Gradient Boosting Regression(GridSearchCV)	1.5157	4.7838	2.1872	0.9692
Test set	0	Linear Regression	5.8342	58.6242	7.6566	0.6183
	1	Lasso	5.8506	58.7927	7.6676	0.6172
	2	Ridge(GridsearchCv Tunned)	5.8506	58.7927	7.6676	0.6172
	3	ElasticNet(GridSearchCV-Tunned)	5.8711	59.2909	7.7001	0.6140
	4	Decision Tree Regressor(GridsearchCV)	3.3998	25.0132	5.0013	0.8372
	5	Radom forest	2.4845	14.3597	3.7894	0.9065
	6	Random Forest-GridSearchCv	2.9497	18.7681	4.3322	0.8778
	7	Gradient Boosting Regression	3.2845	21.6837	4.6566	0.8588
	8	Gradient Boosting Regression(GridSearchCV)	2.3695	13.1971	3.6328	0.9141

Form our calculated MAE,MSE,RMSE and R^2 of each model. Based on R^2 we will decide our model performance.

Assumption: If the difference between Train data and Test data based on R^2 is more than 5% it will indicate the overfitting of model.

- **Linear, Lasso, Ridge and Elastic Net**

All this model have almost similar R^2 (i.e. 61%) for both train and test data. Even after using GridserachCV we have got similar results as of base models).

- **Decision Tree Regression:**

On Decision Tree Regression model we got 100% R^2 without hyper-parameter tuning on both train and test data. So we can see its over fitted.
After hyper-parameter tuning we got 88% on training data and 83% on test data of R^2 which is really good for us.

		Model	MAE	MSE	RMSE	R2_score
Training set	0	Linear Regression	5.8555	60.2995	7.7653	0.6124
	1	Lasoo	5.8691	60.4640	7.7759	0.6113
	2	Ridge GridSearchCV	5.8691	60.4640	7.7759	0.6113
	3	ElasticNet(GridSearchCV-Tunned)	5.8932	60.9027	7.8040	0.6085
	4	Decision Tree Regressor-GridSearchCV	2.8855	18.4446	4.2947	0.8814
	5	Random Forest	0.9375	2.1370	1.4618	0.9863
	6	Random Forest-GridSearchCv	2.6227	14.9003	3.8601	0.9042
	7	Gradient Boosting Regression(GridSearchCV)	1.5157	4.7838	2.1872	0.9692
Test set	0	Linear Regression	5.8342	58.6242	7.6566	0.6183
	1	Lasso	5.8506	58.7927	7.6676	0.6172
	2	Ridge(GridsearchCv Tunned)	5.8506	58.7927	7.6676	0.6172
	3	ElasticNet(GridSearchCV-Tunned)	5.8711	59.2909	7.7001	0.6140
	4	Decision Tree Regressor(GridsearchCV)	3.3998	25.0132	5.0013	0.8372
	5	Radom forest	2.4845	14.3597	3.7894	0.9065
	6	Random Forest-GridSearchCv	2.9497	18.7681	4.3322	0.8778
	7	Gradient Boosting Regression	3.2845	21.6837	4.6566	0.8588
	8	Gradient Boosting Regression(GridSearchCV)	2.3695	13.1971	3.6328	0.9141

- **Random Forest:**

For random forest regression model we got 98% on training data and 90% on testing data of R^2 without hyper-parametric tuning. So we can see its overfitted. After hyper -parameter tuning we got R^2 score as 90% on training data and 87% on test data which is very good for us.

- **Gradient Boosting Regression(Gradient Boosting Machine):**

On Random Forest regressor model, without hyper -parameter tuning we got R^2 score as 86% on training data and 85% on test data. Our model performed well without hyper -parameter tuning. After hyper -parameter tuning we got R^2 score as 96% on training data and 91% on test data, thus we improved the model performance by hyper -parameter tuning.

Conclusion

- THE GRADIENT BOOSTING REGRESSION(GIRDSEARCHCV) AND RANDOM FOREST(GRIDSEARCHCV) ARE BOTH FITTED MODELS WITH GOOD R^2 .


Thank You!