Motorcycle Sharing Demand Prediction Using Data Mining

Report submitted to the SASTRA Deemed to be
University as the requirement for the course

CSE 300 – MINI PROJECT

Submitted by

Garlapadu Harshith (Reg. No: 123003066, B.Tech, III YEAR CSE)

Bayyani Jashwanth (Reg. No: 123003034, B.Tech, III YEAR CSE)

Nikhil Reddy Gaddam (Reg. No: 123003163, B.Tech, III YEAR CSE)

June 2022

SCHOOL OF COMPUTING

THANJAVUR, TAMIL NADU, INDIA – 613 401

SASTRA
ENGINEERING · MANAGEMENT · LAW · SCIENCES · HUMANITIES · EDUCATION
DEEMED TO BE UNIVERSITY
(U/S 3 of the UGC Act, 1956)
THINK MERIT | THINK TRANSPARENCY | THINK SASTRA
THANJAVUR | KUMBAKONAM | CHENNAI

SCHOOL OF COMPUTING

THANJAVUR-613401

Bonafide Certificate

This is to certify that the report titled "**Motor cycle Sharing Demand Prediction using Data Mining**" submitted as a requirement for the course, CSE300: **MINI PROJECT** for B.Tech. is a bonafide record of the work done by **Mr. Garlapadu Harshith (Reg.No.123003066, BTech-CSE**), **Mr. Bayyani Jashwanth (Reg.No.123003034, BTech-CSE), Mr. Nikhil Reddy Gaddam (Reg.No.123003163, BTech-CSE)** during the academic year 2021-22, in the School of Computing, under my supervision.

Signature of Project Supervisor:

Name with Affiliation : Prof. Rajendiran P
Date :

Mini Project *Viva voce* held on ________________

K.R. [signature]

Examiner1 **Examiner2**

SASTRA

ENGINEERING · MANAGEMENT · LAW · SCIENCES · HUMANITIES · EDUCATION

DEEMED TO BE UNIVERSITY

(U/S 3 of the UGC Act,1956)

THINK MERIT | THINK TRANSPARENCY | THINK SASTRA

THANJAVUR | KUMBAKONAM | CHENNAI

SCHOOL OF COMPUTING

THANJAVUR – 613 401

Declaration

We declare that the report titled " **Motor cycle Sharing Demand Prediction using Data Mining** " submitted by us is an original work done by us under the guidance of **Prof.Rajendiran P, School of Computing, SASTRA Deemed to be University** during the even semester of the Pre final academic year 2021- 22, in the **School of Computing**. The work is original, and wherever we have used materials from other sources, we have given due credit and cited them in the text of the report. This report has not formed the basis for the award of any degree, diploma, associate-ship, fellowship or other similar title to any candidate of any University.

Acknowledgements

We expressour deep sense of gratitude to our Honorable Vice-Chancellor, **Dr S. VAIDHYASUBRAMANIAM, SASTRA DEEMED TO BE UNIVERSITY,** the Paramount of the institution, for his auspice and **Dr R.CHANDARAMOULI, The Registrar, SASTRA DEEMED TO BE UNIVERSITY** for providing us with a platform to carry on this project.

We express our sincere thanks and gratitude to **Dr A.UMAMAKESWARI, Dean, School of Computing, SASTRA DEEMED TO BE UNIVERSITY,** for her support in accomplishing this project work.

We are incredibly grateful to our project guide **Dr Siva Rama Krishnan, School of Computing, SASTRA DEEMED TO BE UNIVERSITY,** for her recurrent guidance which assisted me in fulfilling this project work.

We want to record our sincere thanks and gratitude to Project coordinators and Review panel members for their valuable suggestions and comments to improve the quality of our project work.

Table of Contents

List of Figures

Abbreviation

LR	Linear Regression
GBM	Gradient Boosting Machine
SVM	Support Vector Machine
XGB	Extreme Gradient boosting
MAE	Mean Absolute Error
RMSE	Root Mean Squared Error

ABSTRACT

Currently Rental bikes have been offered in numerous cities to increase mobility comfort. It is essential to have the rental bike available and ready for use at the appropriate time to minimize the time that consumers must wait. It eventually becomes very difficult to maintain a consistent supply of rental bikes for the city. Temperature, humidity, and wind speed data. Dewpoint and Visibility, Solar radiation. Weather data (such as snowfall and rain), and dates are all taken into consideration. The study looks into a feature filtering technique to eliminate non-predictive traits and ranks features according to how well they predict. Using a testing set, the performance of five statistical regression models was evaluated after they had been trained using their best hyperparameters: (a) Linear Regression (b) Support Vector Machine (c) Gradient Boosting Machine (d) Boosted Trees (e) Extreme Gradient Boosting Trees.

KEYWORDS: Linear Regression, Support Vector Machine, Extreme Gradient Boosting Trees, Predictive analytics.

CHAPTER - 1

Summary of the Base Paper

1.1. Base Paper Details

Title	Using data mining techniques for bike sharing demand prediction in metropolitan city
Authors	Sathish Kumar V E, Jang woo Park, Yongyun Cho
Journal Name	The International Journal for the Computer and Telecommunications Industry
Year of Publishing	2020
Publisher	ScienceDirect
Indexing	Scopus Indexed

Table 1.1. Base Paper Details

1.2. Introduction

Currently For the purpose of improving transportation comfort, rental bikes have been introduced in several urban areas. It is crucial to make the rental bikes accessible and available to the general public at the appropriate time since it reduces waiting. Eventually, maintaining a steady supply of rental bikes for the city emerges as a top priority. By allowing users to borrow vehicles from any station and return them to any other station, it improves mobility and benefits a larger group of users than renting.

1.3. Proposed Architecture and Methodology

- A dataset comprising of 8760 entries is taken in which the rented bike count was calculated for each hour of each day (365x24)

Parameters/Features	Abbreviation	Type	Measurement
Date	Date	year-month-day	–
Rented Bike count	Count	Continuous	0, 1, 2, …, 3556
Hour	Hour	Continuous	0, 1, 2, …, 23
Temperature	Temp	Continuous	°C
Humidity	Hum	Continuous	%
Windspeed	Wind	Continuous	m/s
Visibility	Visb	Continuous	10 m
Dew point temperature	Dew	Continuous	°C
Solar radiation	Solar	Continuous	MJ/m2
Rainfall	Rain	Continuous	Mm
Snowfall	Snow	Continuous	cm
Seasons	Seasons	Categorical	Autumn, Spring, Summer, Winter
Holiday	Holiday	Categorical	Holiday, Workday
Functional Day	Fday	Categorical	NoFunc, Func

Fig 1.1 Dataset Description

- The data was prepared for Exploratory Data Analysis (EDA) by Checking null values and renaming the attributes.
- The training and testing data was split in the ratio of 4:1 and EDA is carried out for training data
- Now the Exploratory Data Analysis is carried out by visualizing the Categorical data and Numerical data separately by plotting respective graphs for them.
- Heat map has been plotted among the attributes for determining the correlation among the attributes.
- Boruta- A feature selection algorithm has bee been implemented on the training dataset to rank the features and to feed it to the regression algorithms.
- The Following regression algorithms are carried out to analyze the performance based different parameters.

1.Linear Regression

2.XG Boost

3.Gradient Boosting Machine

4.Ada Boost

5.Support Vector Machine

1.4. Evaluation Metrics

- After training the algorithm with training data set the performance of the algorithms has been analyzed using the following :

  1.R2 Score

  2.Root Mean Square Error

  3.Mean Absolute Error

After analyzing the evaluation scores, the best performed algorithm has been selected and we further proceed to the feature elimination.

1.5. Feature Elimination

- After selecting the best performed algorithm, using Boruta feature elimination, each feature is dropped and scores have been calculated;
- Feature importance map has been plotted.

CHAPTER – 2

2.1 Merits:

- As we implemented different regression algorithms, we will get to know which kind of algorithm is suitable for this type of problems by evaluation metrices.
- The Company can use this prediction and make available the supply of bikes on that day.]
- Box plot is plotted to visualize the spread of data.
- Correlation matrix will help us to know how strongly the attributes are correlated

2.2 Demerits:

- The Boruta algorithm eliminates some features having less rank but, when we feed this directly to the regression algorithm the efficiency of the algorithm decreased.
- As dataset was quite big enough which led more computation time.

CHAPTER – 3

Source Code :

3.1 Importing necessary packages and dataset

```
import datetime

import numpy as npy

import pandas as pds

import matplotlib.pyplot as plt

import seaborn as sns

import missingno as msno

from sklearn.model_selection import train_test_split

from boruta import BorutaPy

from sklearn.ensemble import RandomForestRegressor

from sklearn.feature_selection import RFE

from sklearn.feature_selection import f_classif

from boruta import BorutaPy

from sklearn.linear_model import LinearRegression

from sklearn.metrics import r2_score,mean_absolute_error,mean_squared_error

from sklearn.ensemble import GradientBoostingRegressor

from sklearn.svm import SVC

Dataset= pds.read_csv(r'C:\Users\bhara\Downloads\SeoulBikeData.csv',encoding =
'unicode_escape')
```

3.2 Data preparation

```
#First look of dataset

Dataset.head()

Dataset.info()

#renaming Attributes

Dataset.rename({"Temperature(°C)": "Temperature",
```

```
            "Functioning Day":"Functioning_Day",
            "Humidity(%)": "Humidity",
            "Wind speed (m/s)": "Wind_speed",
            "Visibility (10m)": "Visibility",
            "Dew point temperature(°C)": "Dew_point_temperature",
            "Solar Radiation (MJ/m2)": "Solar_Radiation",
            "Snowfall (cm)": "Snowfall",
            "Rainfall(mm)": "Rainfall",
            "Rented Bike Count": "Rented_Bike_Count"},
           axis = "columns", inplace = True)
Dataset.info()
#Checking the nullvalues in Dataset
Dataset.isnull().sum()
#Checking the duplicate values in Dataset
print("Number of Duplicate values:",Dataset.duplicated().sum())
msno.bar(Dataset)
Dataset['Rented_Bike_Count'].plot(kind='hist');
 sns.boxplot(x=Dataset['Rented_Bike_Count'])
print ("the median is",Dataset.Rented_Bike_Count.median())
```

3.3 Exploratory data analysis

```
#splitting dataset into training and testing dataset
training_data , testing_data = train_test_split(Dataset,test_size=0.25,random_state=25)

print ("Number of training dataset entries:",training_data.shape[0])
print ("Number of testing dataset entries:",testing_data.shape[0])
#Splitting the Date column into date month and year separately
```

training_data['Day'] = pds.DatetimeIndex(training_data['Date'], dayfirst=True).day_name()

training_data['month'] = pds.DatetimeIndex(training_data['Date'], dayfirst=True).month_name()

training_data['year'] = pds.DatetimeIndex(training_data['Date'], dayfirst=True).year

3.3.1 Visualizing Catgeroical Data

```
#holiday,seasons-FREQUENCY

Holiday =
pds.DataFrame(training_data.groupby('Holiday').agg({'Rented_Bike_Count':'count'}))

Season=
pds.DataFrame(training_data.groupby('Seasons').agg({'Rented_Bike_Count':'count'}))

fig,size = plt.subplots(1,1,figsize=(15,10))

ax1=plt.subplot(2, 2,1)

sns.barplot(x=Holiday.index, y = Holiday['Rented_Bike_Count'])

ax1.set_ylabel("Frequency")

ax1=plt.subplot(2, 2,2)

sns.barplot(x=Season.index, y = Season['Rented_Bike_Count'])

ax1.set_ylabel("Frequency")

#RENTED BIKE COUNT-SEASONS

plt.figure(figsize=(9,6))

training_data.groupby('Seasons')['Rented_Bike_Count'].sum().plot.bar(color='blue')

plt.ticklabel_format(style='plain', axis='y')

plt.ylabel('Rented Bike Count')

#RENTED BIKE COUNT-MONTH

plt.figure(figsize=(9,6))

training_data.groupby('month')['Rented_Bike_Count'].sum().plot.bar(color='blue')

plt.ticklabel_format(style='plain', axis='y')

plt.ylabel('Rented Bike Count')
```

```
#RENTED BIKE COUNT-DAY
plt.figure(figsize=(9,6))
training_data.groupby('Day')['Rented_Bike_Count'].sum().plot.bar(color='orange')
plt.ticklabel_format(style='plain', axis='y')
plt.ylabel('Rented Bike Count')

#RENTED BIKE COUNT-HOLIDAY
plt.figure(figsize=(9,6))
training_data.groupby('Holiday')['Rented_Bike_Count'].sum().plot.bar(color='blue')
plt.ticklabel_format(style='plain', axis='y')
plt.ylabel('Rented Bike Count')
```

3.3.2 Visualizing Numerical Data

```
#RENTED BIKE COUNT-HOUR
plt.figure(figsize=(9,6))
training_data.groupby('Hour')['Rented_Bike_Count'].sum().plot.bar(color='blue')
plt.ticklabel_format(style='plain', axis='y')
plt.ylabel('Rented Bike Count')
#RENTED BIKE COUNT-TEMPERATURE
plt.figure(figsize=(9,6))
training_data.groupby('Temperature')['Rented_Bike_Count'].sum().plot()
plt.ticklabel_format(style='plain', axis='y')
plt.ylabel('Rented Bike Count')
#RENTED BIKE COUNT-WINDSPEED
plt.figure(figsize=(9,6))
training_data.groupby('Wind_speed')['Rented_Bike_Count'].sum().plot()
```

```
plt.ticklabel_format(style='plain', axis='y')
plt.ylabel('Rented Bike Count')
#RENTED BIKE COUNT-HUMIDITY
plt.figure(figsize=(9,6))
training_data.groupby('Humidity')['Rented_Bike_Count'].sum().plot()
plt.ticklabel_format(style='plain', axis='y')
plt.ylabel('Rented Bike Count')
#RENTED BIKE COUNT-RAINFALL
plt.figure(figsize=(9,6))
training_data.groupby('Rainfall')['Rented_Bike_Count'].mean().plot()
plt.ticklabel_format(style='plain', axis='y')
plt.ylabel('Rented Bike Count')
#RENTED BIKE COUNT-VISIBILITY
plt.figure(figsize=(9,6))
training_data.groupby('Visibility')['Rented_Bike_Count'].mean().plot()
plt.ticklabel_format(style='plain', axis='y')
plt.ylabel('Rented Bike Count')
#RENTED BIKE COUNT-SOLAR RADIATION
plt.figure(figsize=(9,6))
training_data.groupby('Solar_Radiation')['Rented_Bike_Count'].mean().plot()
plt.ticklabel_format(style='plain', axis='y')
plt.ylabel('Rented Bike Count')
```

3.4 Data Preprocessing

```
# Transforming the Holiday variable
Dataset['Holiday']=Dataset['Holiday'].apply(lambda x: 1 if x=='Holiday' else 0)
# Transforming the Functioning Day variable
```

```
Dataset['Functioning_Day']=Dataset['Functioning_Day'].apply(lambda x: 1 if x=='Yes' else 0)

# Transforming the Seasons variable

from sklearn.preprocessing import OneHotEncoder

one_hot_encoded_data = pds.get_dummies(Dataset['Seasons'])

Dataset=pds.concat([Dataset,one_hot_encoded_data],axis=1)

# Take a look of dataset after coverting categorical columns

Dataset.head()

Dataset.drop('Seasons',axis=1,inplace=True)

sns.heatmap(Dataset.corr(),annot=True)

sns.set(rc={'figure.figsize':(20,15)})
```

3.5 Boruta Algorithm

```
newdata=Dataset.drop('Date',axis=1)

ind_col = [col for col in newdata.columns if col != 'Rented_Bike_Count']

dep_col = 'Rented_Bike_Count'

X = newdata[ind_col]

y = newdata[dep_col]

# splitting data into training and test set

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.50,random_state = 42)

rf_1 = RandomForestRegressor(n_jobs=-1, oob_score= True)

feat_selector = BorutaPy(rf_1, n_estimators = 'auto', max_iter= 50)

feat_selector.fit(X_train, y_train)

feature_names=npy.array(X_train.columns)

print(feat_selector.support_)

print(feat_selector.ranking_) #Rank 1 is the best
```

```
feature_ranks = list(zip(feature_names,
                         feat_selector.ranking_,
                         feat_selector.support_))
for feat in feature_ranks:
    print('Feature: {:<30} Rank: {},  Keep: {}'.format(feat[0], feat[1], feat[2]))
```

3.6 Methods

3.6.1 Linear Regression

```
regressor = LinearRegression()
regressor.fit(X_train,y_train)
y_pred_train=regressor.predict(X_train)
y_pred=regressor.predict(X_test)
importance = regressor.coef_
print("R2 SCORE:",r2_score(y_test, y_pred))
MSE  = mean_squared_error(y_test, y_pred)
RMSE = npy.sqrt(MSE)
print("RMSE :" ,RMSE)
MAE  = mean_absolute_error(y_test, y_pred)
print("MAE :" , MAE)
# summarize feature importance
for i,v in enumerate(importance):
print('Feature: %0d, Score: %.5f' % (i,v))
# plot feature importance
plt.bar([x for x in range(len(importance))], importance)
plt.show()
```

3.6.2 Gradient Boosting Machine

```
def plot_feature_importance(importance,names,model_type):
    #Create arrays from feature importance and feature names
```

```
    feature_importance = npy.array(importance)
    feature_names = npy.array(names)

#Create a DataFrame using a Dictionary
    data={'feature_names':feature_names,'feature_importance':feature_importance}
    fi_df = pds.DataFrame(data)

#Sort the DataFrame in order decreasing feature importance
    fi_df.sort_values(by=['feature_importance'], ascending=False,inplace=True)

#Define size of bar plot
    plt.figure(figsize=(10,8))
#Plot Searborn bar chart
    sns.barplot(x=fi_df['feature_importance'], y=fi_df['feature_names'])
#Add chart labels
    #plt.title(model_type + 'FEATURE IMPORTANCE')
    plt.xlabel('FEATURE IMPORTANCE')
    plt.ylabel('FEATURE NAMES')
ensemble = GradientBoostingRegressor()
ensemble.fit(X_train,y_train)
y_pred_train=ensemble.predict(X_train)
y_pred=ensemble.predict(X_test)
print("R2 SCORE:",r2_score(y_test, y_pred))
MSE  = mean_squared_error(y_test, y_pred)
RMSE = npy.sqrt(MSE)
print("RMSE :" ,RMSE)
MAE  = mean_absolute_error(y_test, y_pred)
```

```
print("MAE :" , MAE)
plot_feature_importance(ensemble.feature_importances_,X_train.columns,'GBM')
```

3.6.3 XG Boost

```
from xgboost import XGBRegressor
model_xgb = XGBRegressor(colsample_bytree=0.3,
            gamma=0,
            learning_rate=0.02, eval_metric ='mae',
            max_depth=2,
            min_child_weight=2,
            n_estimators=10000,
            reg_alpha=0.9,
            reg_lambda=0.2,
            subsample=0.5,
            seed=42)
model_xgb.fit(X_train, y_train)
y_pred_train=model_xgb.predict(X_train)
y_pred=model_xgb.predict(X_test)

r2_score(y_test, y_pred)
print(r2)
MSE  = mean_squared_error(y_test, y_pred)
RMSE = npy.sqrt(MSE)
print("RMSE :" ,RMSE)
MAE  = mean_absolute_error(y_test, y_pred)
print("MAE :" , MAE)
print(model_xgb.feature_importances_)
```

```
plt.bar(range(len(model_xgb.feature_importances_)),
model_xgb.feature_importances_)

plt.show()
```

3.6.4 Ada Boost

```
from sklearn.ensemble import AdaBoostRegressor

model = AdaBoostRegressor()

model.fit(X_train,y_train)

y_new=model.predict(X_train)

y_new2=model.predict(X_test)

r2=r2_score(y_test, y_new2)

print("R2 SCORE :",r2)

MSE  = mean_squared_error(y_test, y_new2)

RMSE = npy.sqrt(MSE)

print("RMSE :" ,RMSE)

MAE  = mean_absolute_error(y_test, y_new2)

print("MAE :" , MAE)

plot_feature_importance(model.feature_importances_,X_train.columns,'ADABOOST')
```

3.6.5 Support Vector Machine

```
from sklearn.svm import SVR

# most important SVR parameter is Kernel type. It can be #linear,polynomial or
gaussian SVR. We have a non-linear condition #so we can select polynomial or
gaussian but here we select RBF(a #gaussian type) kernel.

regressor = SVR(kernel='rbf')

regressor.fit(X,y)

#5 Predicting a new result

y_pred_train=model_xgb.predict(X_train)

y_pred=model_xgb.predict(X_test)
```

```
r2_score(y_test, y_pred)
print("R2 SCORE:",r2)
MSE  = mean_squared_error(y_test, y_pred)
RMSE = npy.sqrt(MSE)
print("RMSE :" ,RMSE)
MAE  = mean_absolute_error(y_test, y_pred)
print("MAE :" , MAE)
plot_feature_importance(model.feature_importances_,X_train.columns,'RANDOM FOREST')
```

3.7 Selecting Features

```
#GBM-NO temp
new_train=X_train.drop('Temperature',axis=1)
new_test=X_test.drop('Temperature',axis=1)
ensemble = GradientBoostingRegressor()
ensemble.fit(new_train,y_train)
y_pred_train=ensemble.predict(new_train)
y_pred=ensemble.predict(new_test)
r2=r2_score(y_test, y_pred)
print("R2 SCORE:",r2)
MSE  = mean_squared_error(y_test, y_pred)
RMSE = npy.sqrt(MSE)
print("RMSE :" ,RMSE)
MAE  = mean_absolute_error(y_test, y_pred)
print("MAE :" , MAE)
plot_feature_importance(ensemble.feature_importances_,new_train.columns,'GBM')
#GBM-NO WEATHER ,Functioning_day
new_train=X_train.drop(['Summer','Winter','Autumn','Spring','Functioning_Day'],axis=1)
```

```
new_test=X_test.drop(['Summer','Winter','Autumn','Spring','Functioning_Day'],axis=1
)
ensemble = GradientBoostingRegressor()
ensemble.fit(new_train,y_train)
y_pred_train=ensemble.predict(new_train)
y_pred=ensemble.predict(new_test)
r2=r2_score(y_test, y_pred)
print("R2 SCORE:",r2)
MSE  = mean_squared_error(y_test, y_pred)
RMSE = npy.sqrt(MSE)
print("RMSE :" ,RMSE)
MAE  = mean_absolute_error(y_test, y_pred)
print("MAE :" , MAE)
plot_feature_importance(ensemble.feature_importances_,new_train.columns,'GBM'
#GBM- NO SNOWFALL
new_train=X_train.drop('Snowfall',axis=1)
new_test=X_test.drop('Snowfall',axis=1)
ensemble = GradientBoostingRegressor()
ensemble.fit(new_train,y_train)
y_pred_train=ensemble.predict(new_train)
y_pred=ensemble.predict(new_test)
r2=r2_score(y_test, y_pred)
print("R2 SCORE :",r2)
MSE  = mean_squared_error(y_test, y_pred)
RMSE = npy.sqrt(MSE)
print("RMSE :" ,RMSE)
MAE  = mean_absolute_error(y_test, y_pred)
```

```
print("MAE :" , MAE)
plot_feature_importance(ensemble.feature_importances_,new_train.columns,'GBM')
#GBM-No windspeed
new_train=X_train.drop('Wind_speed',axis=1)
new_test=X_test.drop('Wind_speed',axis=1)
ensemble = GradientBoostingRegressor()
ensemble.fit(new_train,y_train)
y_pred_train=ensemble.predict(new_train)
y_pred=ensemble.predict(new_test)
r2=r2_score(y_test, y_pred)
print("R2 SCORE :",r2)
MSE  = mean_squared_error(y_test, y_pred)
RMSE = npy.sqrt(MSE)
print("RMSE :" ,RMSE)
MAE  = mean_absolute_error(y_test, y_pred)
print("MAE :" , MAE)
plot_feature_importance(ensemble.feature_importances_,new_train.columns,'GBM')
#GBM- NO Visibility
new_train=X_train.drop('Visibility',axis=1)
new_test=X_test.drop('Visibility',axis=1)
ensemble = GradientBoostingRegressor()
ensemble.fit(new_train,y_train)
y_pred_train=ensemble.predict(new_train)
y_pred=ensemble.predict(new_test)
r2=r2_score(y_test, y_pred)
print(r2)
MSE  = mean_squared_error(y_test, y_pred)
```

```
RMSE = npy.sqrt(MSE)
print("RMSE :" ,RMSE)
MAE  = mean_absolute_error(y_test, y_pred)
print("MAE :" , MAE)
plot_feature_importance(ensemble.feature_importances_,new_train.columns,'GBM')
```

CHAPTER-4

Results :

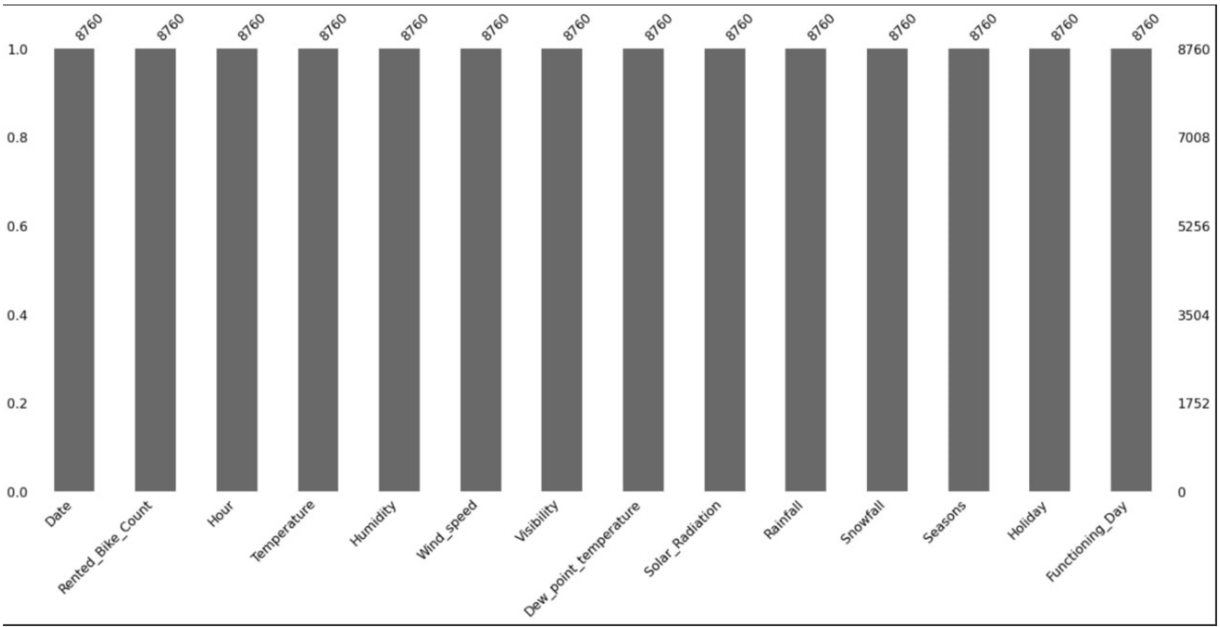


Figure 1: Visualizing Dataset

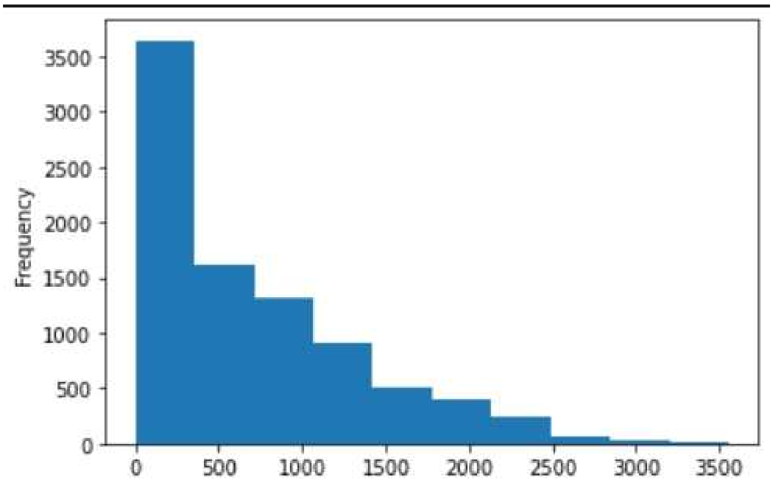


Figure 2: Histogram of Rented_Bike_Count

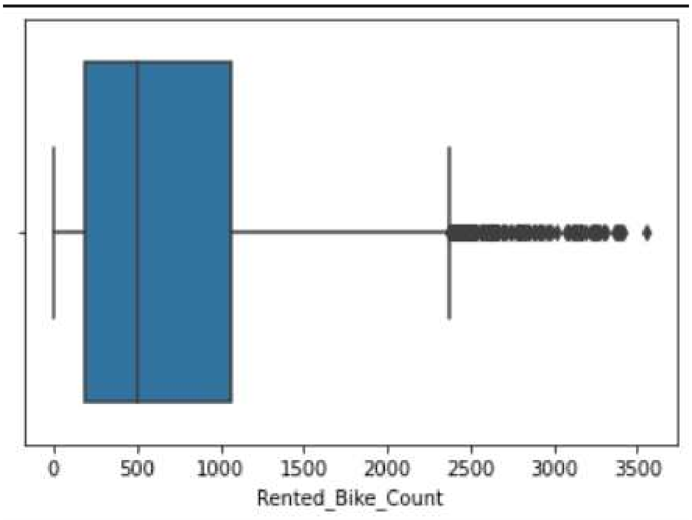


Figure 3: Box plot of rented bike count

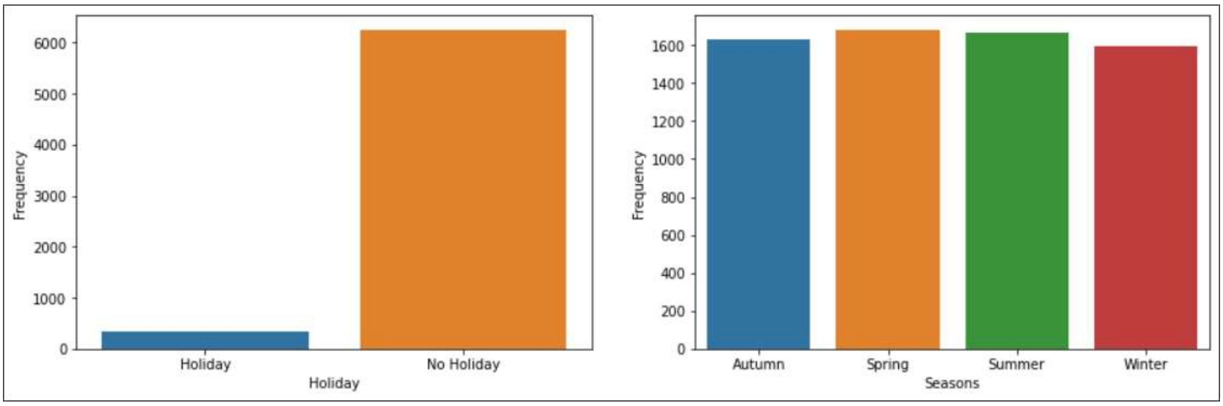
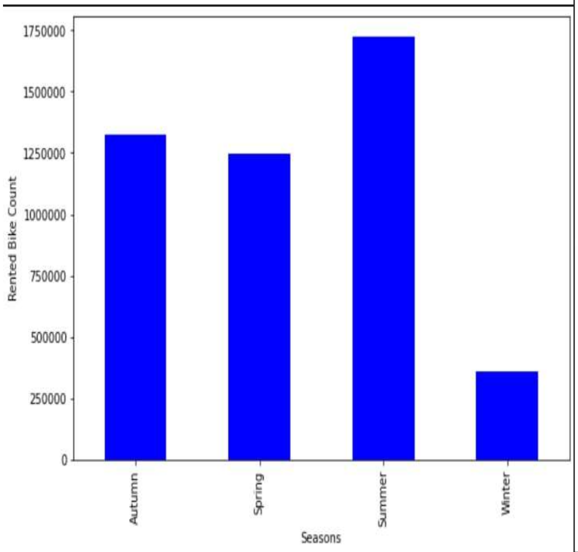


Figure 4: Rented bike count vs categorical attributes



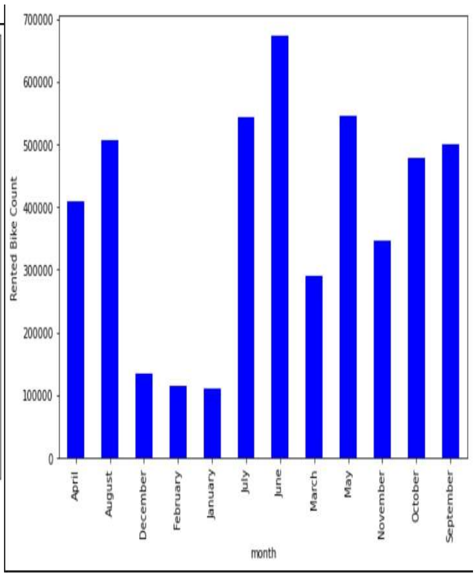


Figure 5: rented bike count vs seasons

Fig 6: month vs bike count

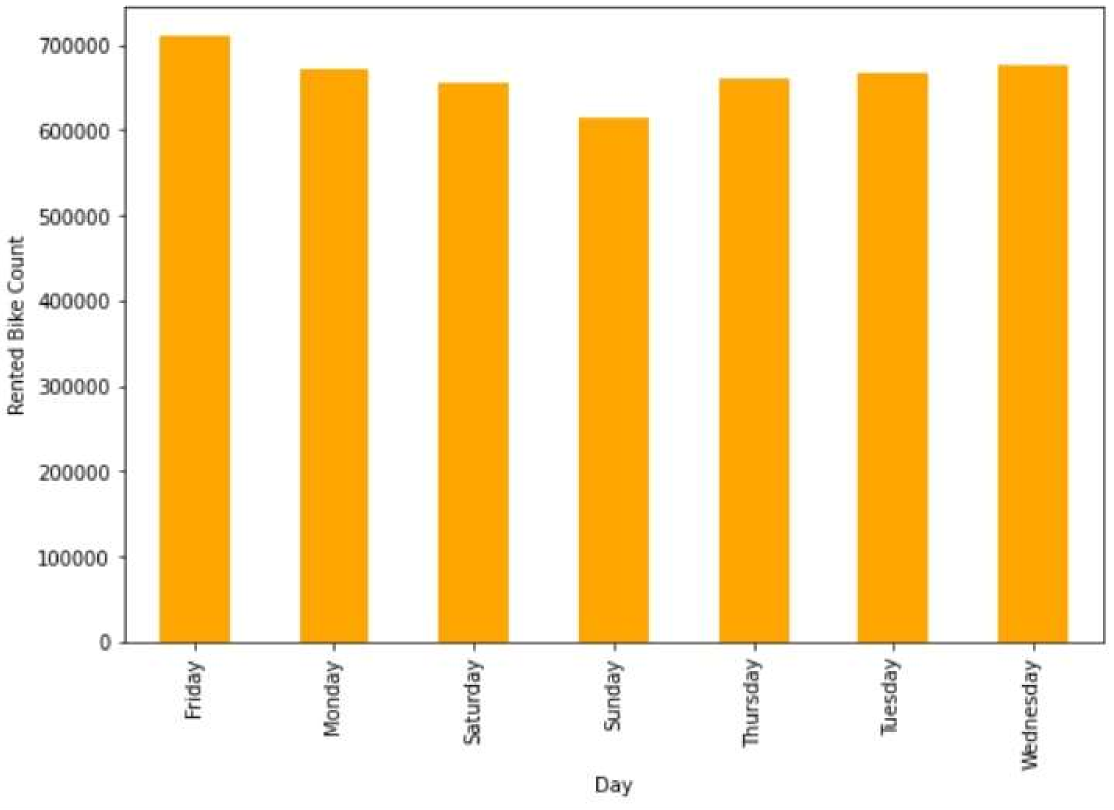


Figure 7: plot for number of bikes rented on each day of the week for one year

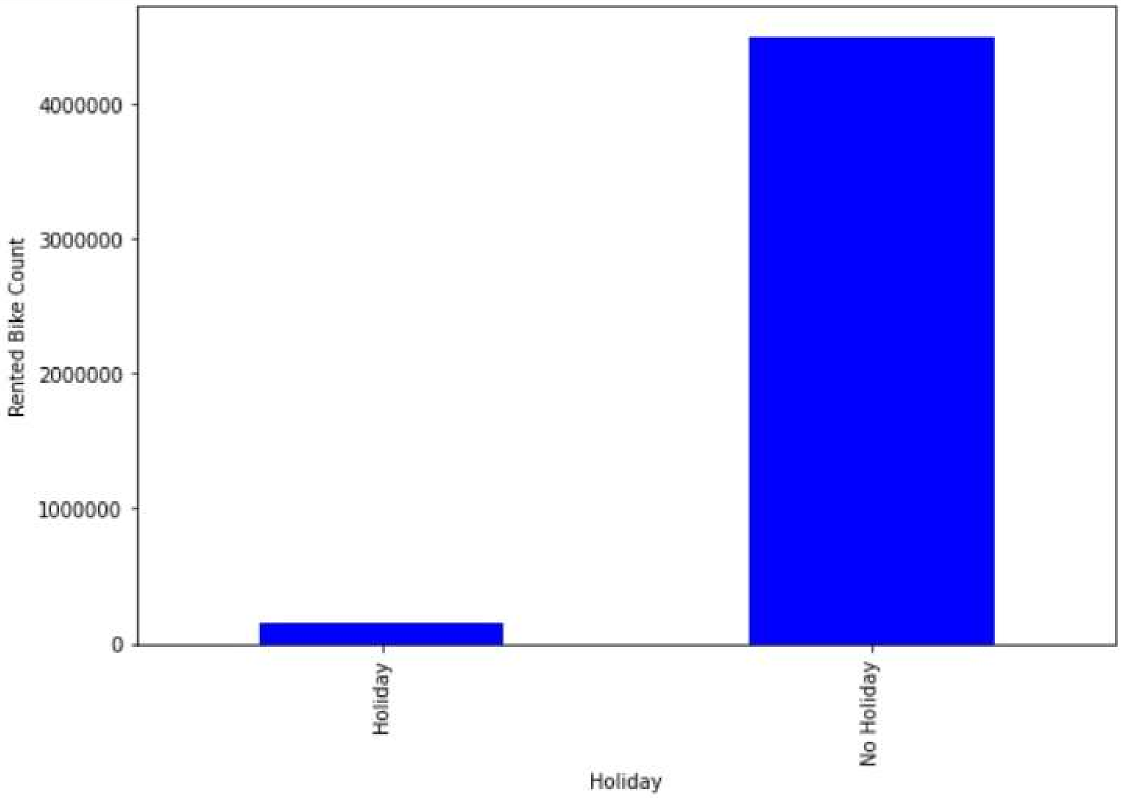


Figure 8: Bikes rented on holiday or non holiday

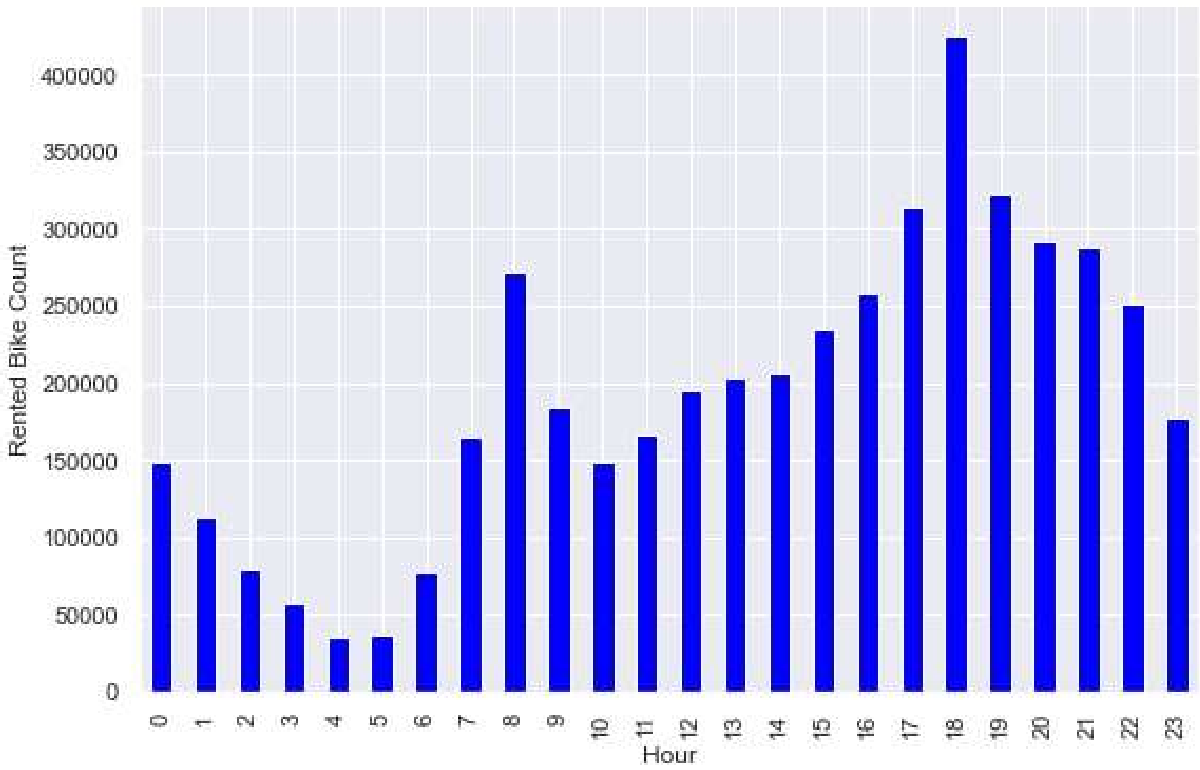
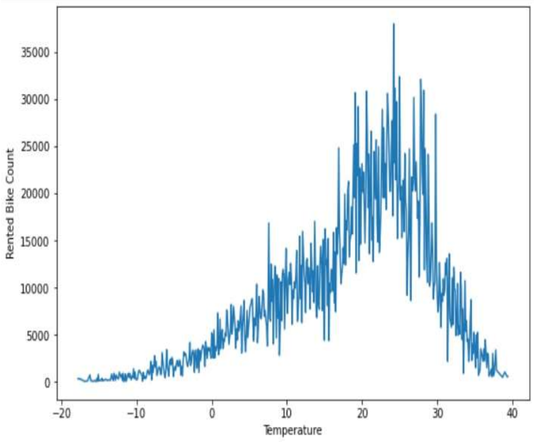


Figure 9: plot between rented bike count and each hour of the day



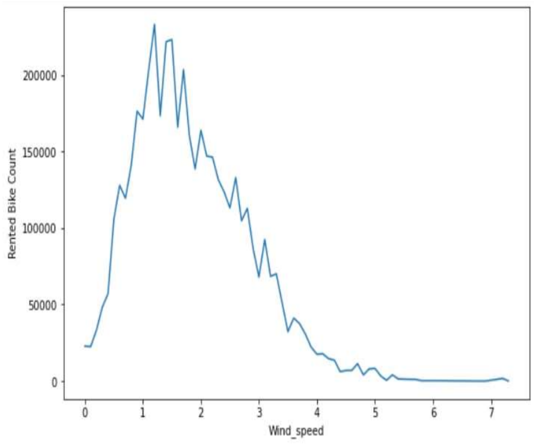
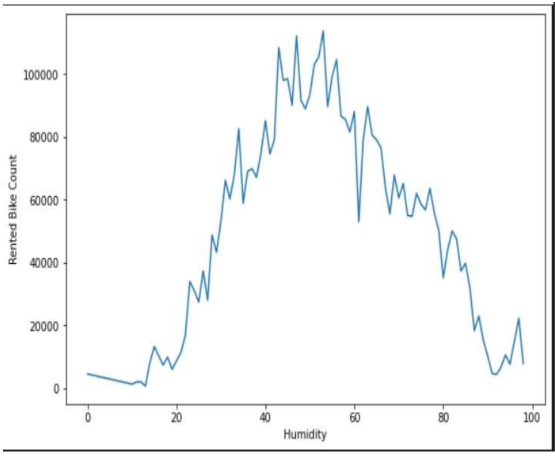


Figure 10: Temp vs Rented bike count

Fig 11: Windspeed vs bike count



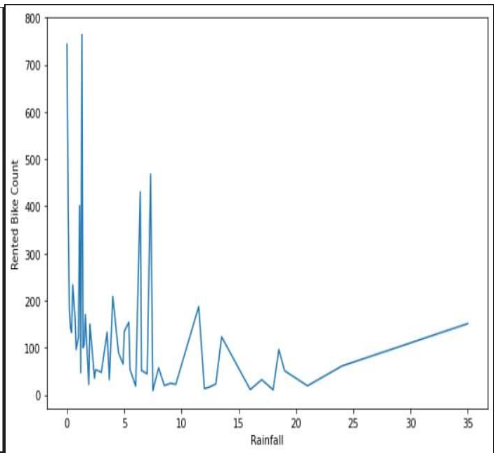
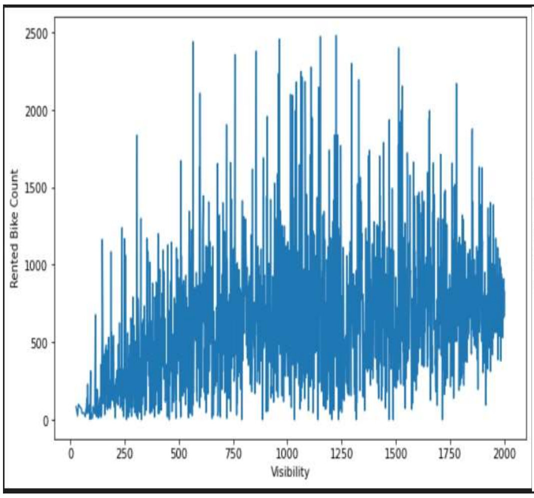


Fig 12: Humidity vs bike count **Fig 13:** Rainfall vs bike count



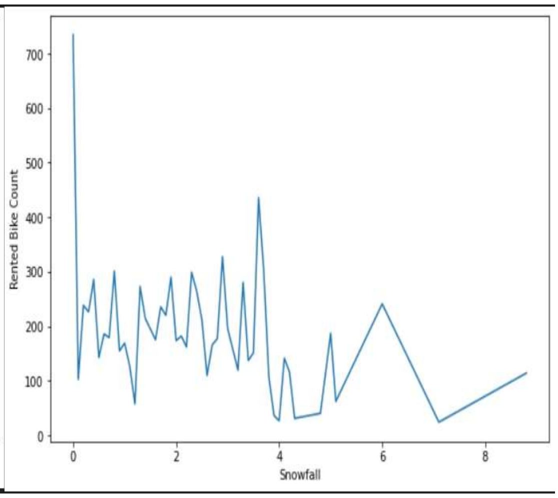
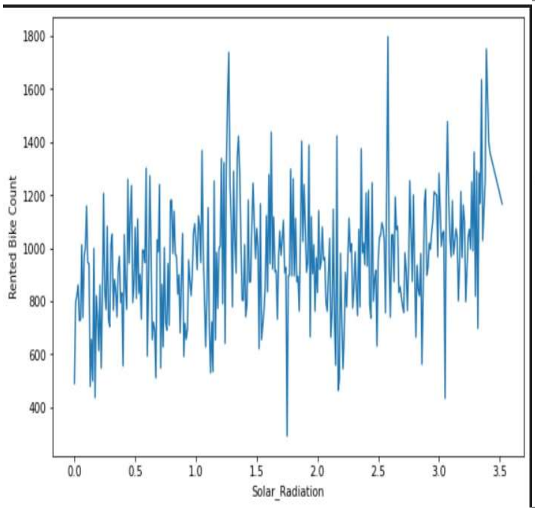


Fig 14: Visibility vs bike count **Fig 15:** Snow fall vs bike count



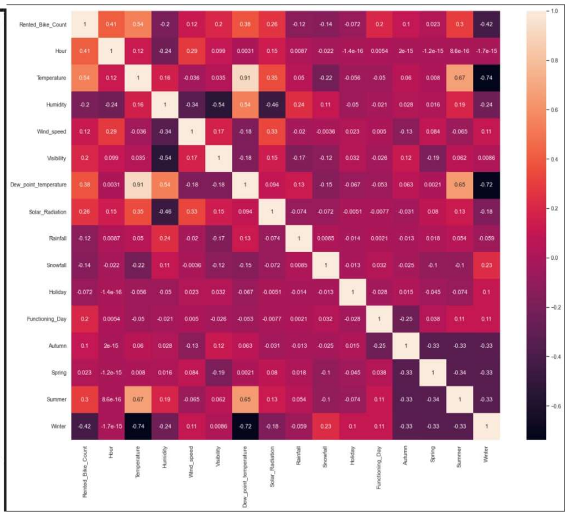
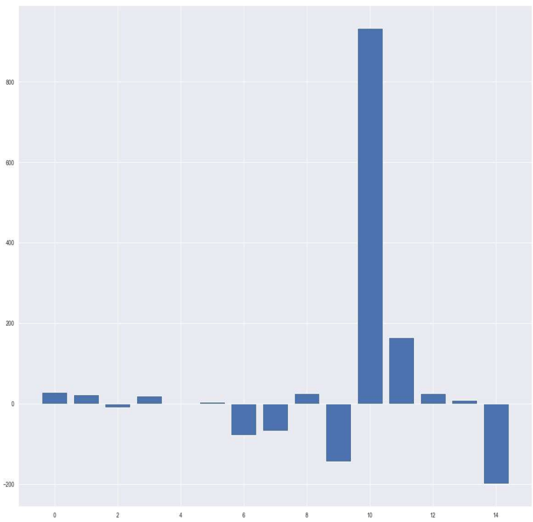


Fig 16 : solar radiation vs bike count

Fig 17 : Heatmap between attributes



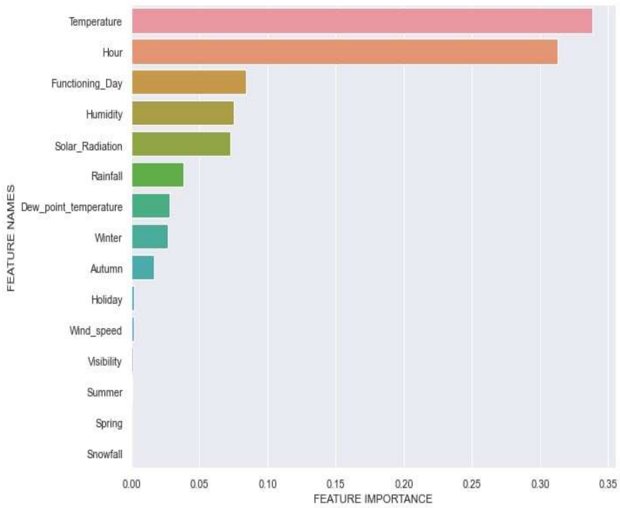


Fig 18: feature importance linear regression

Fig 19: feature importance GBM

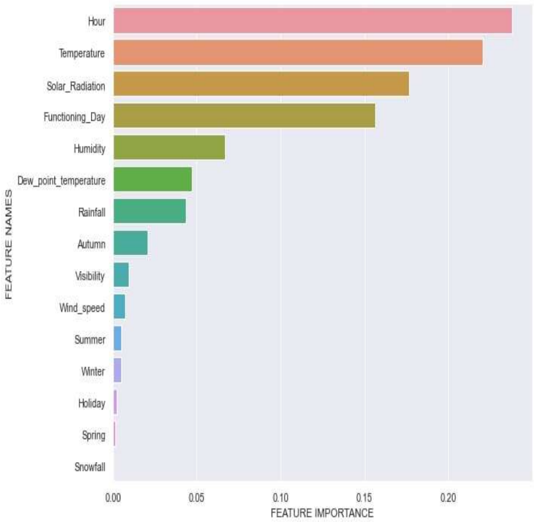


Fig 20: feature importance Xgboost

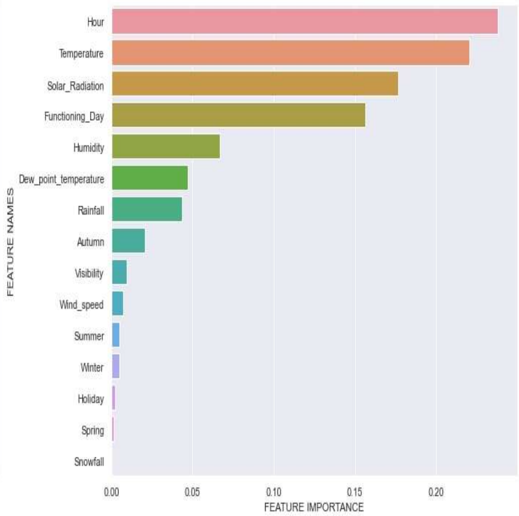


Fig 21: feature importance SVM

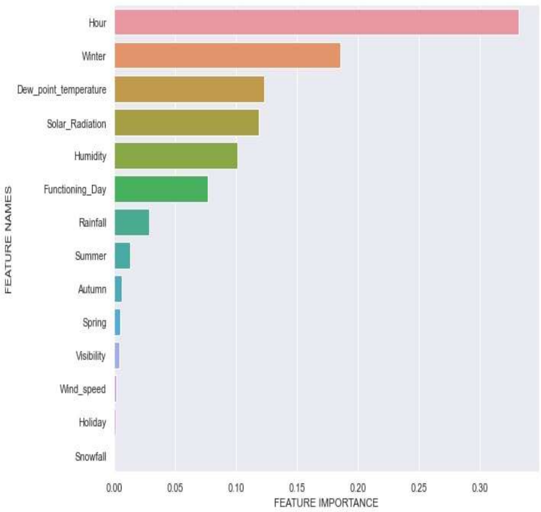


Fig 22: GBM drop（Temperature ）

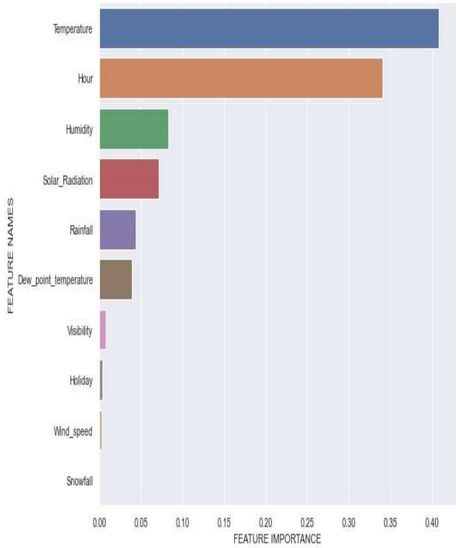


Fig 23: GBM drop(weather)

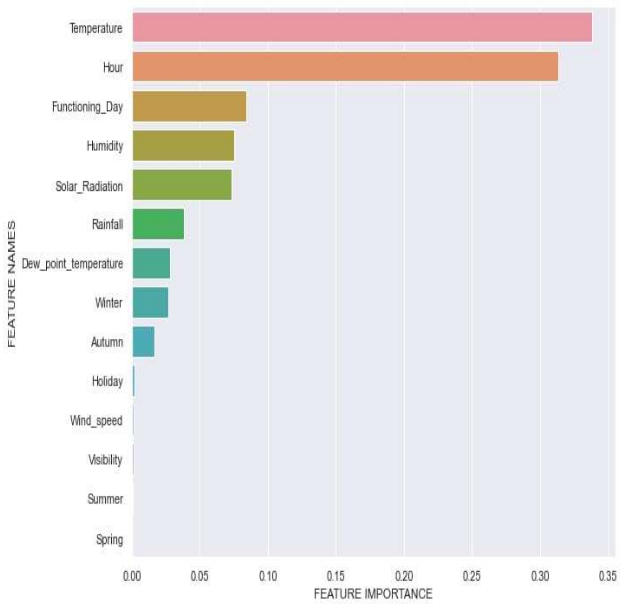


Fig 24: GBM drop (snowfall)

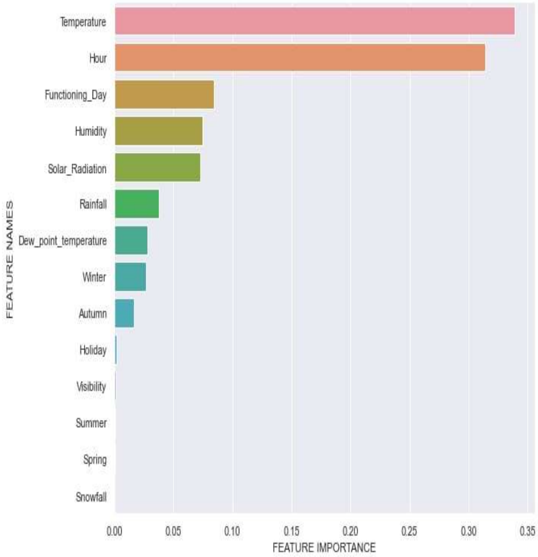


Fig 25: GBM drop (windspeed)

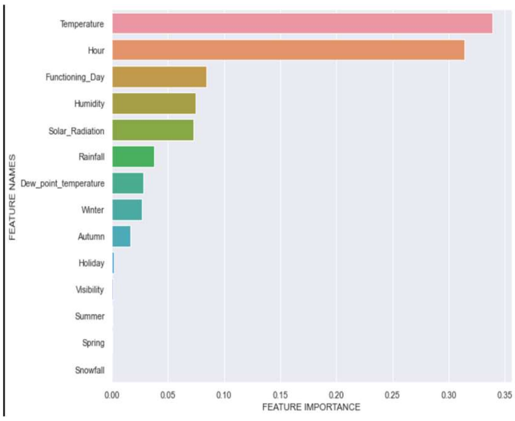


Fig 26: GBM drop(Visibility)

CHAPTER-5

Conclusion:

After performing the various models the Gradient Boosting and Xgboost found to be the best model that can be used for the Bike Sharing Demand Prediction since the performance metrics(RMSE, MAE)shows lower and(R2 score)shows higher value for Gradient Boosting model and Xgboost. R2 value for Xgboost and Gradient Boosting are 0.827 and 0.827 respectively. We can use either Random Forest or Gradient Boosting model for the bike rental stations. Out of all the features temperature is the most important feature this implies that temperature is directly impacting the Rented Bike Counts. By eliminating features one by one we need to eliminate them.

Future Extensions:

- Location based Motorcycle prediction this project was carried out for the Seoul Data set i.e., from Korea so in extension of this we can implement a geological approach of this project by implementing it for every city by collection the necessary data.

CHAPTER – 6

REFERENCES

John Pucher, Ralph Buehler, Why Canadians cycle more than Americans: a comparative analysis of bicycling trends and policies, Transp. Policy 13 (3) (2006) 265–279

SEOUL OPEN DATA. URL: http://data.seoul.go.kr/.

SOUTH KOREA PUBLIC HOLIDAYS. URL: publicholidays.go.kr

Maricica Nistor, André Dias, Bike distribution model for urban data applications, Int. J. Transp. Dev. Integr. 3 (1) (2019) 67–78.

Dimitrios Tomaras, Ioannis Boutsis, Vana Kalogeraki, Modeling and predicting bike demand in large city situations, in: 2018 IEEE International Conference on Pervasive Computing and Communications, PerCom, IEEE, 2018.

CHAPTER 7

Appendix of theBase Paper:

URL: https://www.sciencedirect.com/science/article/pii/S0140366419318997