SUPERVISED ML REGRESSION CAPSTONE PROJECT

BIKE SHARING DEMAND PREDICTION

Contents

Introduction

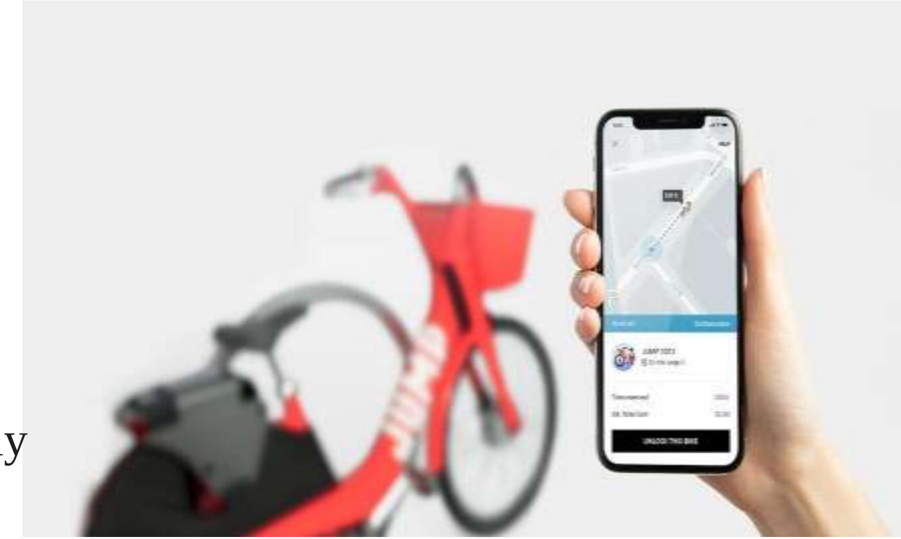
A bike rental or bike hire business rents out motorcycles for short periods of time, Usually for a few hours. Most rentals are provided by bike shops
as a sideline to their main businesses of sales and service, but some shops specialize in rentals.

As with car rental, bicycle rental shops primarily serve people who do not have access to vehicles, typically travelers and particularly tourists.

Bike rental shops rent by the day or week as well as by the hour, and these provide an excellent opportunity for those who would like to avoid shipping their own bikes but would like to do a multi-day bike tour of a particular area.

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

Data Analysis Steps

Imported Libraries

In this part, we imported the required libraries NumPy, Pandas, matplotlib, and seaborn, to perform Exploratory Data Analysis and for prediction, we imported the Scikit learn library.

Descriptive Statistics

In this part, we start by looking at descriptive statistic parameters for the dataset. We will use describe() function to find out mean, median and standard deviation.

Missing Value Imputation

We will now check for missing values in our dataset. after checking non existed any missing values, In case there are any missing entries, we will impute them with appropriate values.

Encoded categorical data

Since machine learning models can only be trained with numeric data ,we used OneHot encoder and Label Encoder to change categorical data into numerical data.

Data Analysis Steps

Scaling Data

We have used MinMax scaler, StandardScaler and RobustScaler for our numeric data so that it becomes range bounded.

Spliting training and testing set

We split the dataset into a training and testing set. We have a randomly selected 20% subset of the data for testing. Also, we have used just the numeric and encoded columns.

Checked various models and applied hyperparamter tuning

We have used around 12 models and have applied hyperparamter tuning to get us the best accuracy with least error

Graphical Representation

We started with Univariate Analysis then bivariate Analysis and concluded with various prediction models driving the Demand for bikes

Attributes of each variable

Date: Date in year-month-day format

Rented Bike Count: Count of bikes rented at each hour

Hour: Hour of the Day

Temperature: Temperature in Celsius

Humidity: Humidity in %

Windspeed: Speed of wind in m/s

Visibility (10m): Visibility

Dew point temperature: Dew Point Temp (Celsius)

Solar radiation: Radiation in MJ/m2

Rainfall: Rainfall (mm)

Snowfall: Snowfall (cm)

Seasons: Winter, Spring, Summer, Autumn

Holiday: Holiday/No holiday

Functioning Day: if the day is neither weekend, holiday than 1 else 0

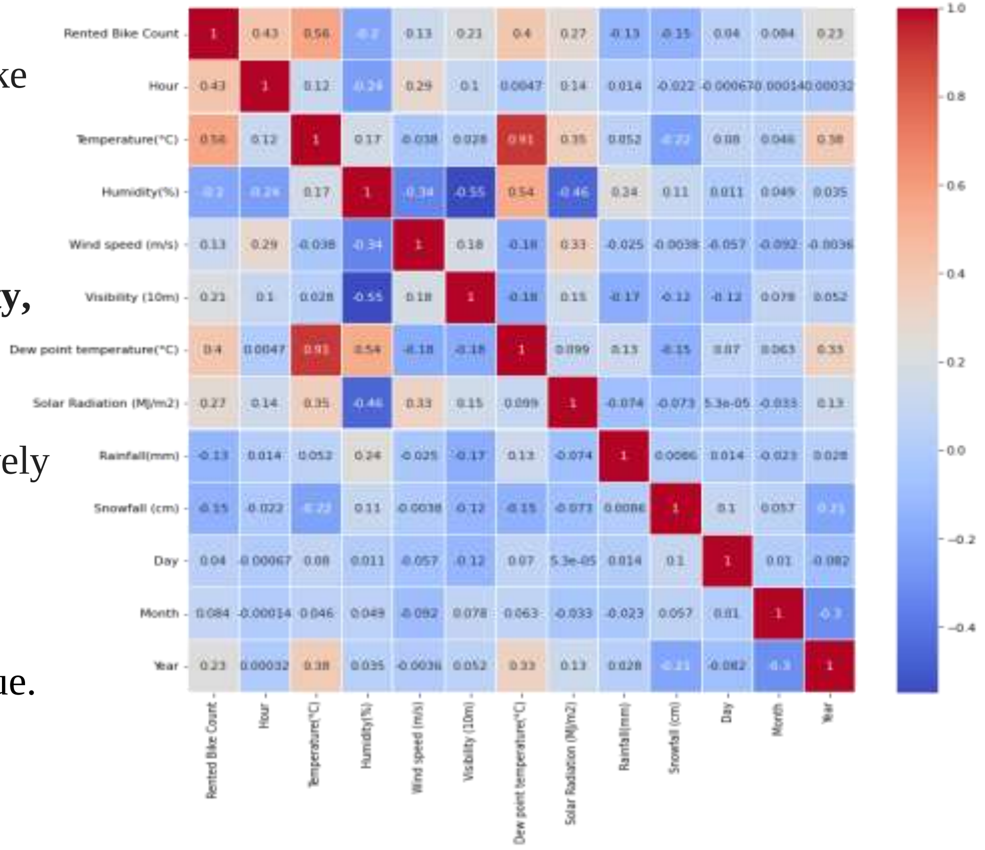
Correlation map

- Heat map shows slightly positive relation of Rented bike count with **Hour, Temperature, Dew point Temperature, Solar Radiation.**

- Bike sharing count is negatively co-related to **Humidity, Snowfall, Rainfall.**

- **Temperature** and **Dew point temperature** are positively co-related.

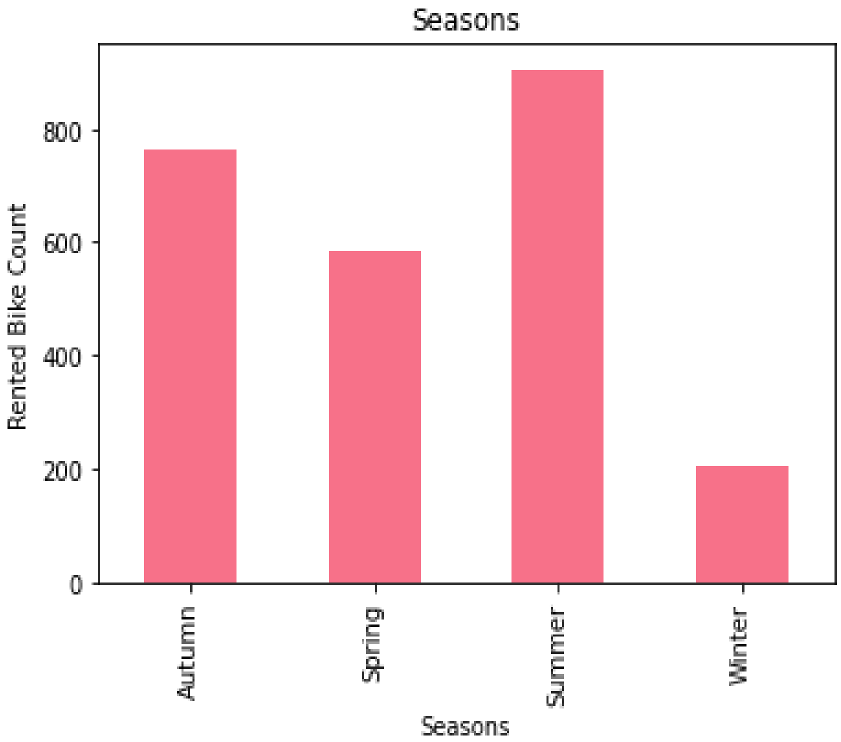
- **Temperature** and **Dew point temperature** are almost 0.91correlated, So it is generating multicollinearity issue. So we drop the Dew point temperature feature



Data Summary And EDA

Bikes Rented Per Season

- Highest number of bikes were rented in **Summer.**
- Second highest Bikes were rented in **Autumn** followed by **Spring**.
- **Winter** appears to be the least popular season for bike rentals.
- The **extreme temperatures** in Seoul in the **winter** might be a factor in the **low demand** for bikes in the winter

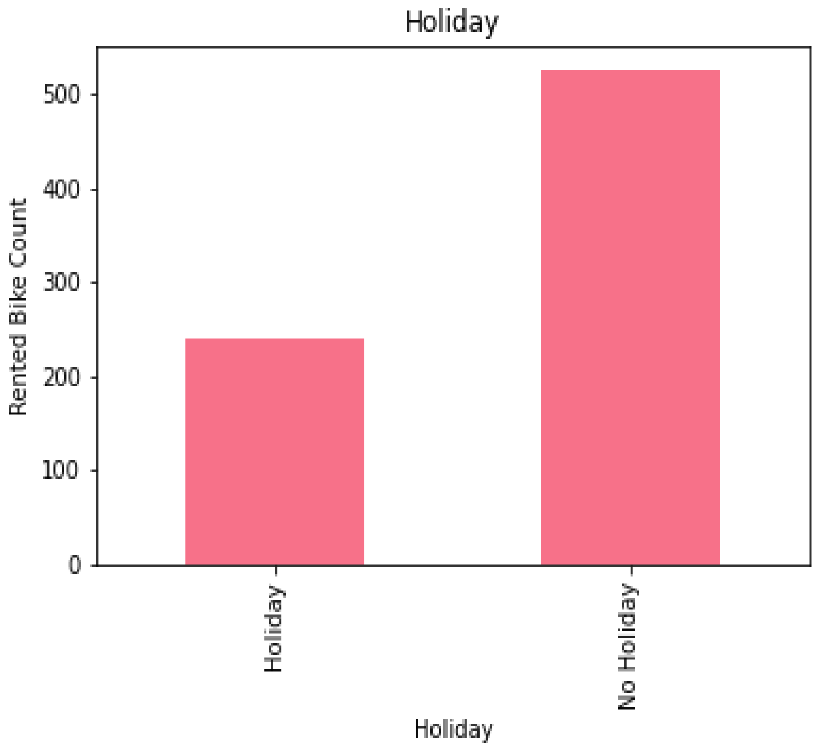


Bike Renting Trend on Holidays

- **People prefer** to use the bike on **Non-holiday more** compared to **Holidays**.

- **5.9 million** bikes are rented on **Non-holidays**, only a meager **2.15 million** bikes were rented on **holidays**.

- It's reasonable to conclude that the **majority of clients** in the **bike rental sector** are from **Seoul's working class**.

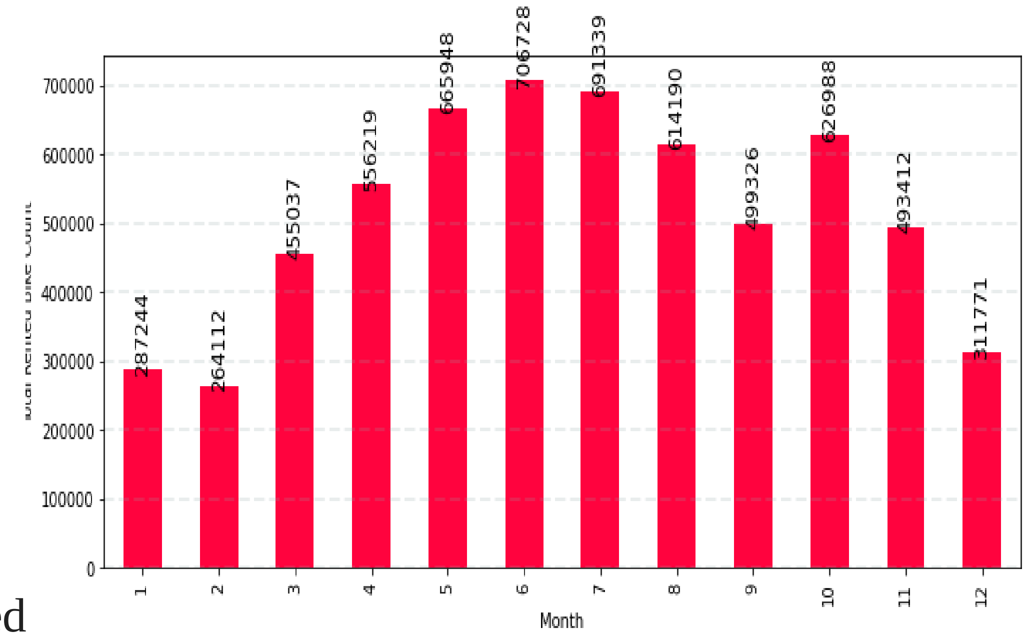


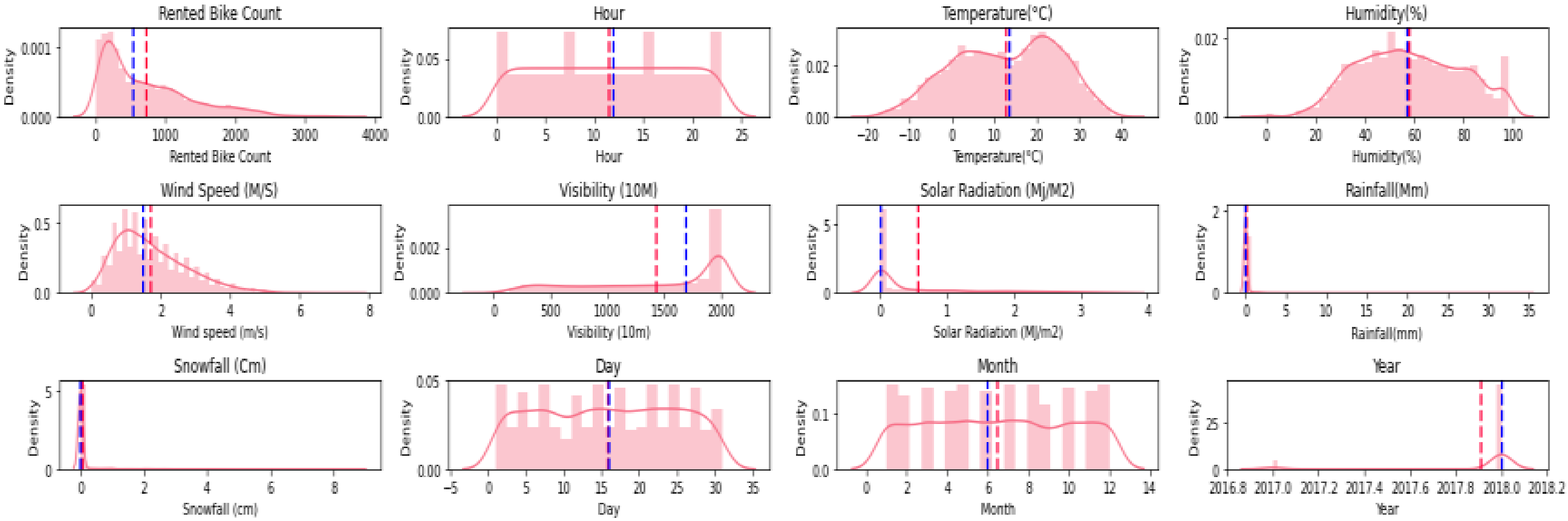
Bike Booking Monthly Trend

- **June** is the most preferred month for bike booking around **706K** bikes were rented in June.

- **July** and **May** are the second and third best.**691K** bikes were booked in **July**, and **665K** were booked in **May**.

- Demand for bikes was **least** in **Jan**, followed by **Feb** and **Dec**. **287K** bikes were rented in **Jan**, **264k** in **Feb**, and **311K** in **Dec.**





- Bike sharing is at its peak between 4pm- 6pm
- Bike sharing is least between 4am-6am.
- Most preferred temperature for bike renting is 20-30 Degree Celsius.
- Bike sharing is least when temperature is < 5 and >35 Degree Celsius.
- Humidity of 40%-60% is most favourable for bike sharing.
- Wind speed of 1m/s -2 m/s is most favourable for bike sharing.
- Bike sharing count is directly related to Visibility in the area.
- Optimum Solar Radiation, no rainfall and no snowfall leads to higher bike renting in Seoul.

Feature Engineering on Data

1. Encode categorical data :
    a) One-Hot Encoding
    b) Label Encoding

2. Identify Inputs and Target (Independent and Dependent Variable)
    Input (Ind.) = Other all Variable except "Rented Bike Count"
    Output (Dep.) = Rented Bike Count

3. Scale values using:
    a) MinMaxScaler()
    b) StandardScalar
    c) RobustScaler()

4. Split the dataset into training and test sets.

Encoding Data

Since machine learning models can only be trained with numeric data, we need to convert categorical data to numbers. A common technique is to use one-hot encoding for categorical columns. OneHot encoding involves adding a new binary (0/1) column for each unique category of a categorical column

Index	Categorical column
1	Cat A
2	Cat B
3	Cat C

→

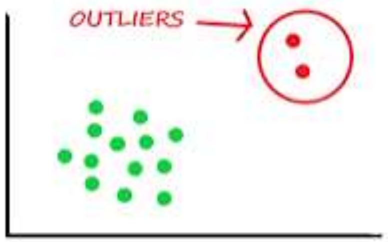
Index	Cat A	Cat B	Cat C
1	1	0	0
2	0	1	0
3	0	0	1

OneHot encoding approach eliminates the order but it causes the number of columns to expand vastly. So for columns with more unique values try using other techniques like LabelEncoding

Outliers

Outliers are those **data points that are significantly different from the rest of the dataset**. They are often abnormal observations that skew the data distribution, and arise due to inconsistent data entry, or erroneous observations.

Outliers brings skewness in the data. Thus decreasing the accuracy sometimes. So we dealt with this problem and made our distribution normal.



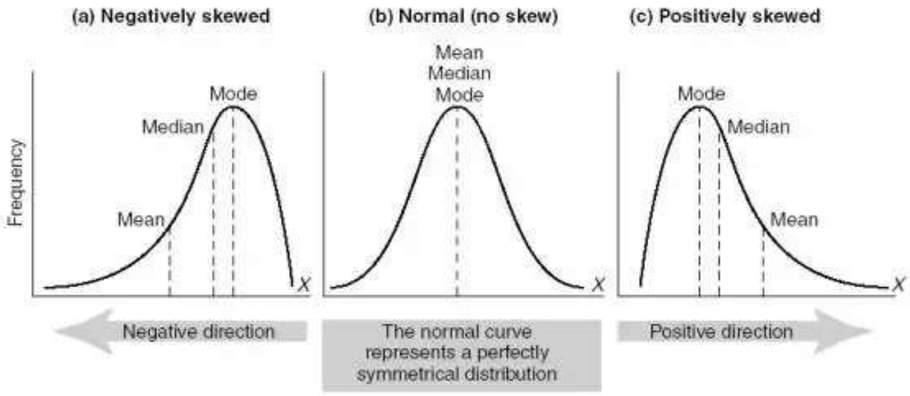
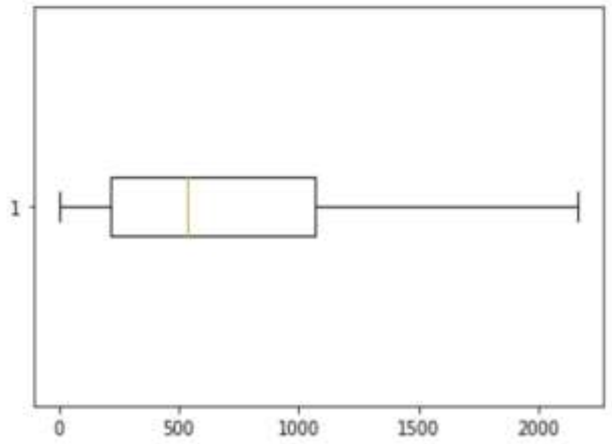


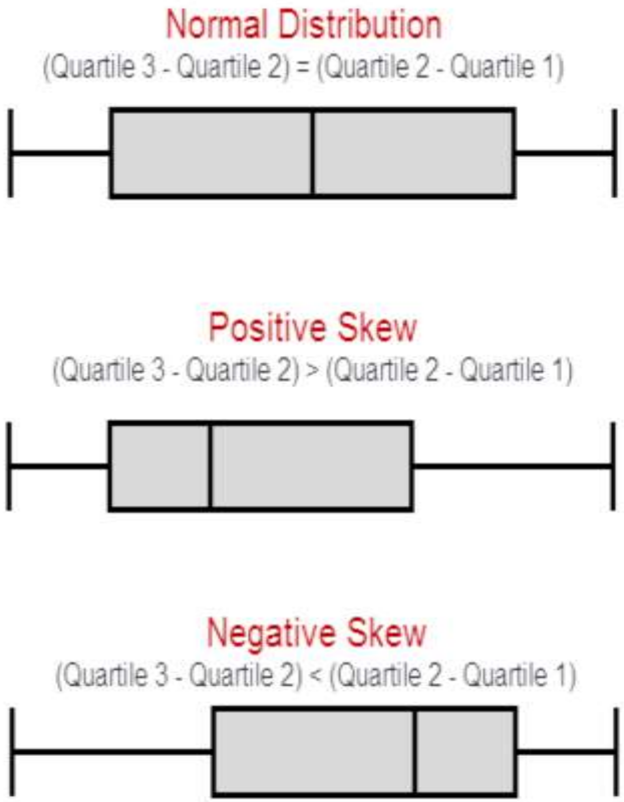
FIGURE 15.6 Examples of normal and skewed distributions

Outliers(continued)

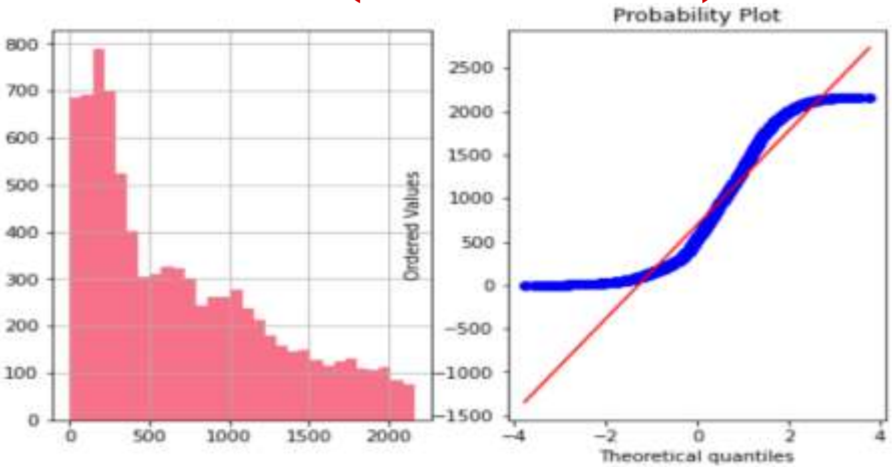
```
plt.boxplot(le_df['Rented Bike Count'],vert=False)
plt.show()
```



When we plotted our boxplot we noted that it is positively skewed (you can refer the figure on the right)

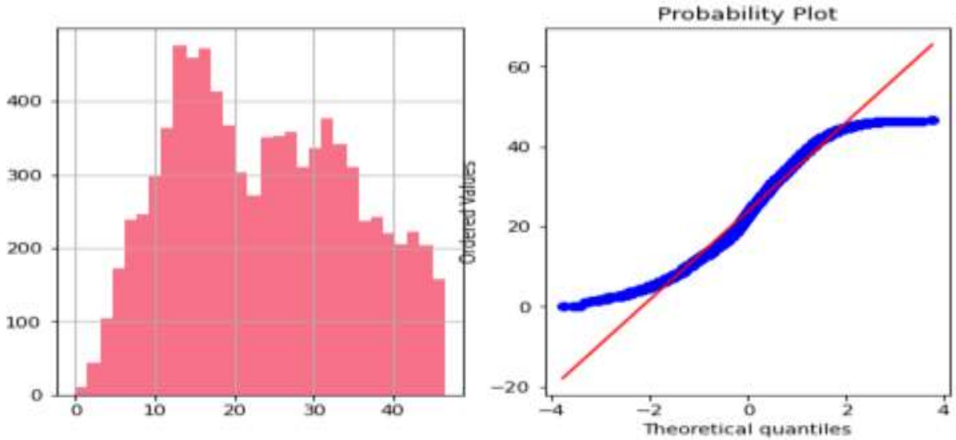


Outliers(continued)



In the following graph plots you can see positive skewed histogram and its corresponding skewed probability plot because of outliers present.

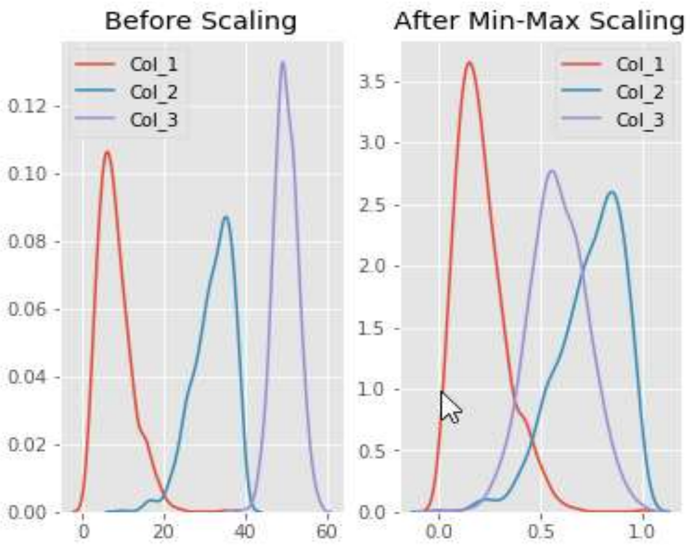
To correct the skewness we have applied <u>square root transform</u> and got the normal distribution from positive skewed data.



Scaling

Types of scaling :

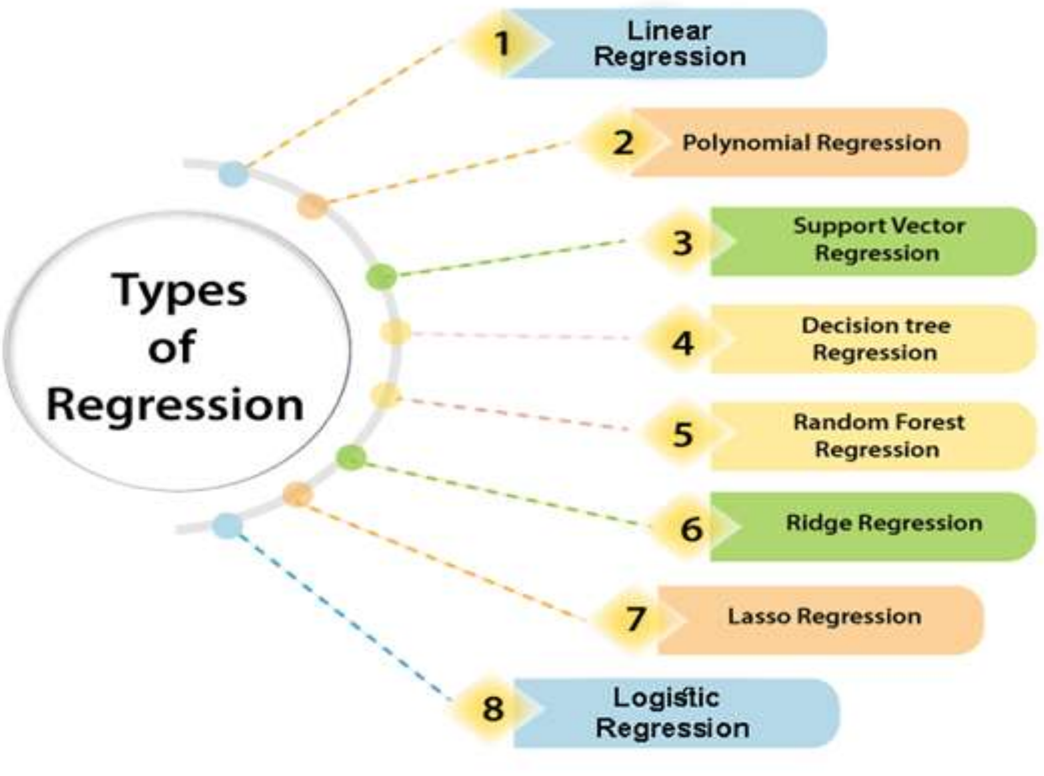
1)MinMaxScalar- scales all the data features in the range [0, 1] or else in the range [-1, 1] if there are negative values. It scales the values to a specific value range Without changing the shape of the original distribution.



2)StandardScalar-In Machine Learning, StandardScaler is used to resize the distribution of values so that the mean of the observed values is 0 and the standard deviation is 1.

3)RobustScalar-This Scaler removes the median and scales the data according to the quantile range (defaults to IQR: Interquartile Range).

Scaling Data and Model Building



We checked the accuracy of our models using different scaling methods.

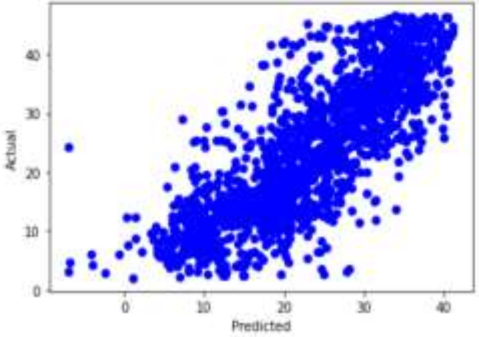
We applied all 3 different scalers and checked the accuracy difference between them.

Checking difference between Actual test value and Predicted value

Using RobustScaler

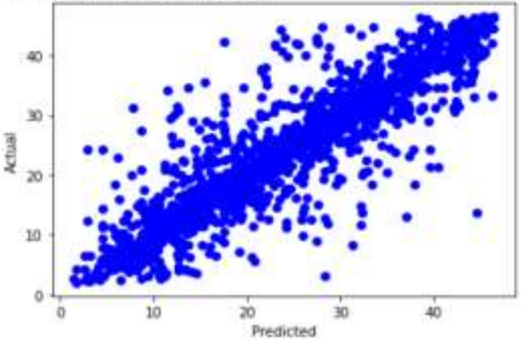
```
predict(LinearRegression(),x,y)
```

```
R^2 is 0.6484023843668458
 Adj R^2 is 0.645680068105243
 RMSE is: 6.782028398532071
```



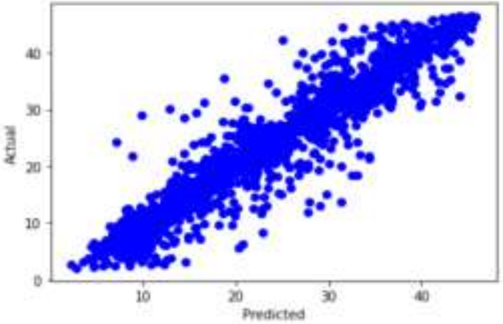

```
predict(DecisionTreeRegressor(),x,y)
```

```
R^2 is 0.8069036990534749
 Adj R^2 is 0.8054086115535912
 RMSE is: 5.026013149561544
```



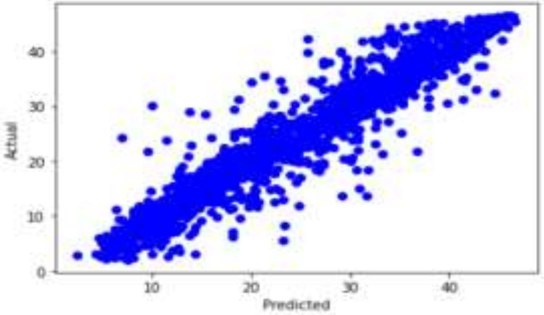

```
predict(RandomForestRegressor(),x,y)
```

```
R^2 is 0.9005566706729107
 Adj R^2 is 0.8997867104101042
 RMSE is: 3.6068200175658416
```




```
predict(LGBMRegressor(),x,y)
```

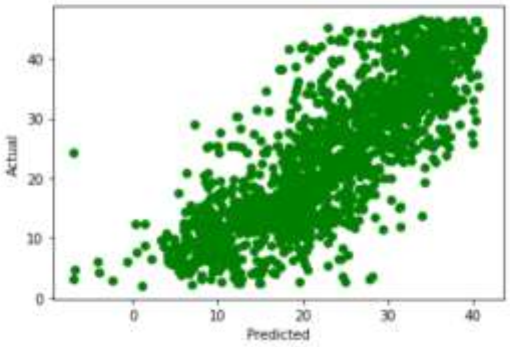
```
R^2 is 0.9092926042308541
 Adj R^2 is 0.9085902837156672
 RMSE is: 3.4447522507533312
```



Using MinMaxcaler

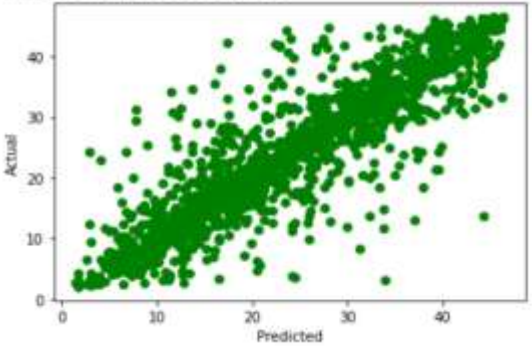
```
predict_mm(LinearRegression(),x,y)
```

```
R^2 is 0.6484023843668462
 Adj R^2 is 0.6456800681052435
 RMSE is: 6.782028398532068
```



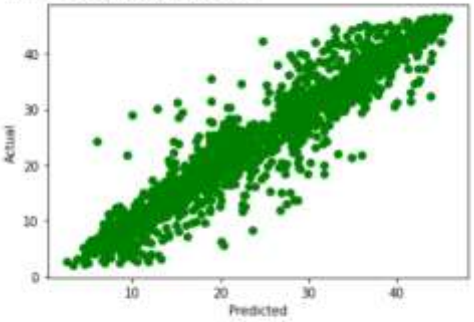

```
predict_mm(DecisionTreeRegressor(),x,y)
```

```
R^2 is 0.7977426057530496
 Adj R^2 is 0.7961765866195115
 RMSE is: 5.143856533499763
```



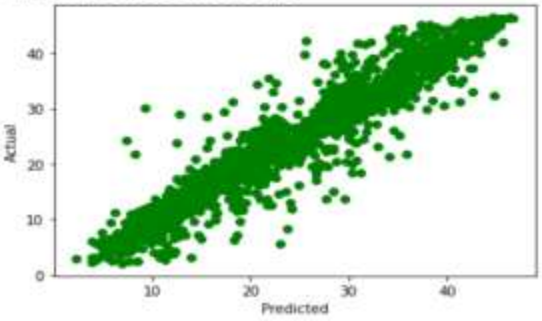

```
predict_mm(RandomForestRegressor(),x,y)
```

```
R^2 is 0.9008458581207129
 Adj R^2 is 0.9000781369507125
 RMSE is: 3.601571769649944
```




```
predict(LGBMRegressor(),x,y)
```

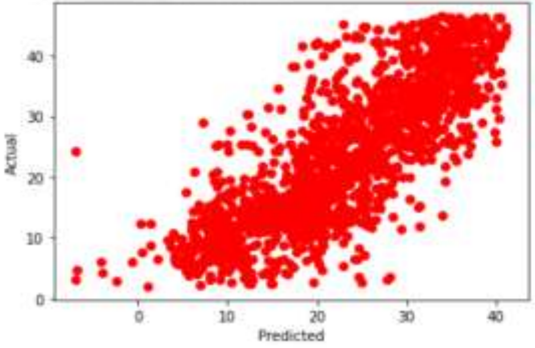
```
R^2 is 0.9086899452289576
 Adj R^2 is 0.9079829585035117
 RMSE is: 3.4561767553073417
```



Using StandardScaler

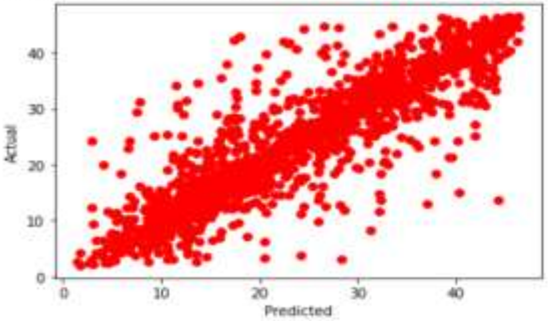
```
predict_ss(LinearRegression(),x,y)
```

```
R^2 is 0.6484023843668462
 Adj R^2 is 0.6456800681052435
 RMSE is: 6.782028398532068
```



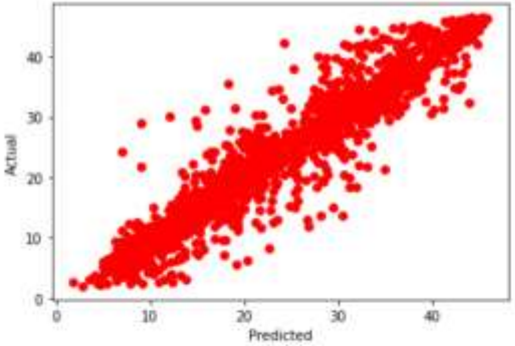

```
predict_ss(DecisionTreeRegressor(),x,y)
```

```
R^2 is 0.8020976046249035
 Adj R^2 is 0.8005653049585091
 RMSE is: 5.088176511331331
```



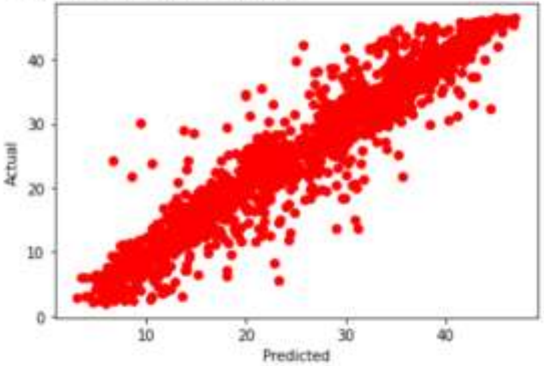

```
predict_ss(RandomForestRegressor(),x,y)
```

```
R^2 is 0.9003154064765299
 Adj R^2 is 0.8995435781764672
 RMSE is: 3.6111927058693256
```




```
predict_ss(LGBMRegressor(),x,y)
```

```
R^2 is 0.9070138170111401
 Adj R^2 is 0.9062938525210537
 RMSE is: 3.487754065892132
```



Models List

In this project we used total twelve models, so that we can compare the final Root mean square error and R2 score of this models.

```
models = [
        ['LinearRegression: ',          LinearRegression()],
        ['Lasso: ',                     Lasso()],
        ['Ridge: ',                     Ridge()],
        ['KNeighborsRegressor: ',       neighbors.KNeighborsRegressor()],
        ['SVR:' ,                       SVR(kernel='rbf')],
        ['DecisionTree ',               DecisionTreeRegressor(random_state=42)],
        ['RandomForest ',               RandomForestRegressor(random_state=42)],
        ['ExtraTreeRegressor :',        ExtraTreesRegressor(random_state=42)],
        ['GradientBoostingRegressor: ', GradientBoostingRegressor(random_state=42)],
        ['XGBRegressor: ',              xgb.XGBRegressor(random_state=42)] ,
        ['Light-GBM: ',                 lightgbm.LGBMRegressor(num_leaves=41, n_estimators=200,random_state=42)],
        ['MLPRegressor: ', MLPRegressor(  activation='logistic', solver='sgd',learning_rate='adaptive',max_iter=1000,learning_rate_init=0.01]
]
```

Models Accuracy and Results

1 to 12 of 12 entries Filter

index	Name	Train_Time	Train_R2_Score	Test_R2_Score	Test_RMSE_Score
0	LinearRegression:	0.026958227157592773	0.6478907877040745	0.6583635068923448	6.4352389952296525
1	Lasso:	0.03720688819885254	0.6362247150339719	0.6434832997542703	6.573890897623658
2	Ridge:	0.02260303497314453	0.6478907426327043	0.6583560977217399	6.435308776306985
3	KNeighborsRegressor:	0.07252621650695801	0.7595401403043068	0.634924859314125	6.65232844723656
4	SVR:	4.194986581802368	0.45718840668401606	0.4651504542193533	8.051899927587922
5	DecisionTree	0.06784844398498535	1.0	0.802263313270086	4.89582831149827
6	RandomForest	3.4867093563079834	0.985537289576449	0.888481120923255	3.6706330850384896
7	ExtraTreeRegressor :	2.0492005348205566	1.0	0.8965501742971964	3.541175367844362
8	GradientBoostingRegressor:	0.9909980297088623	0.8888860544961422	0.8721765871820792	3.9362960771190374
9	XGBRegressor:	0.3925619125366211	0.8866652042037074	0.8704475104859376	3.9628299303029495
10	Light-GBM:	0.393587589263916	0.970814771075317	0.907160067519356	3.354671205399188
11	MLPRegressor:	4.288397312164307	0.03218944551773395	0.02595415501917342	10.866063834702375

As per above results

- Train_R2 and Test_R2 Score being near to 1 is considered as a good model.
- Lightgbm, ExtraTreeRegressor and RandomForestRegressor give us max R2 score and least Root mean square error on test set.

So, In above results best models are

No	Model Name	Model Accuracy Score in %
6	RandomForest	88%
7	ExtraTreeRegressor	89%
8	GradientBoostingRegressor	87%
9	XGBRegressor	87%
10	Light-GBM	90%

Hyperparameter Tuning of GradientBoostingRegressor

In hyperparameter tuning we have chosen the important hyperparater such as learning_rate, max_depth, and the n_estimators. The max_depth and n_estimators are the same parameters that we chose in a random forest. Here we are taking an extra that is the learning_rate.

```
[ ] gbr = GradientBoostingRegressor()
    gbr_params = {
        "n_estimators":[250,500,1000],
        "max_depth":[2,4,6],
        "learning_rate":[0.01,0.1,1],
        "loss": ['ls','huber','quantile'],
    }
```

We call the Boosting Regressor Constructor and define the parameters. Here we have applied all relevant possible values for each the hyperparamter

```
[ ] regressor = GridSearchCV(gbr, gbr_params, verbose=1,cv=3,n_jobs=-1)
    regressor.fit(X_train,y_train)

    Fitting 3 folds for each of 81 candidates, totalling 243 fits
    GridSearchCV(cv=3, estimator=GradientBoostingRegressor(), n_jobs=-1,
                 param_grid={'learning_rate': [0.01, 0.1, 1],
                             'loss': ['ls', 'huber', 'quantile'],
                             'max_depth': [2, 4, 6],
                             'n_estimators': [250, 500, 1000]},
                 verbose=1)
```

```
[ ] regressor.best_params_

    {'learning_rate': 0.1, 'loss': 'ls', 'max_depth': 6, 'n_estimators': 500}
```

After Hyperparameter tuning, the accuracy of the model went from **87%** to **91.7%**

```
Model Accuracy: 0.917
The mean squared error (MSE) on test set: 9.8680
Root Mean Squared Error is 3.2018
```

Conclusions

- Most numbers of Bikes were rented in **Summer**, followed by **Autumn**, **Spring**, and **Winter**. **May-July** is the peak Bike renting Season, and **Dec-Feb** is the least preferred month for bike renting.

- Majority of the client in the bike rental sector belongs to the Working class. This is evident from EDA analysis where bike demand is more on weekdays, working days in Seoul.

- **Temperature** of **20-30 Degrees**, evening time **4 pm- 8 pm, Humidity** between **40%-60%** are the most favorable parameters where the Bike demand is at its peak.

- **Temperature, Hour** of the day, **Solar radiation**, and **Humidity** are major driving factors for the Bike rent demand.

- Feature and Labels had a weak linear relationship, hence the prediction from the linear model was very low. Best predictions are obtained with GradientBoostingRegressor with applied hyperparamter tuning with r2 score of **0.917** and RMSE of **3.2018**

THANK YOU