Capstone Project

Bike Sharing Demand Prediction

By,
Hark Pun
Data Science Trainee,
AlmaBetter, Bangalore.

Contents

What is Bike-Sharing system?

- Bike-sharing system is a shared micro-mobility service for short term bike rental. The service can be free of charge (e.g., paid by a city) or offered for a price.

- In many cities, renting a bike has become an everyday digital service. For unlocking the bike, the users only need a subscription card or their smartphone. The users can leave the bikes to suitable docking stations or areas.

- This makes bike-sharing a convenient alternative to both public transportation and private cars.

- Bike-sharing customers don't depend on transportation routes and can use the service on demand. They can reach their destination without traffic jams or parking costs while contributing to a cleaner city environment as well as their health.

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 bike count required at each hour for the stable supply of rental bikes.

Attributes

- **Date** : Year-Month-Day
- **Rented Bike Count** - Count of bikes rented at each hour
- **Hour** - Hour of the day
- **Temperature** - Temperature in Celsius
- **Humidity** - %
- **Wind Speed** - 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** – Nonfunctional Hours & Functional Hours

NUMERICAL
- Humidity
- Wind Speed
- Visibility
- Dew-Temp
- Solar radiation
- Rainfall
- Snowfall

CATEGORICAL
- Seasons
- Holiday
- Hour
- Functioning day

TARGET FEATURE
- Rented Bike Count

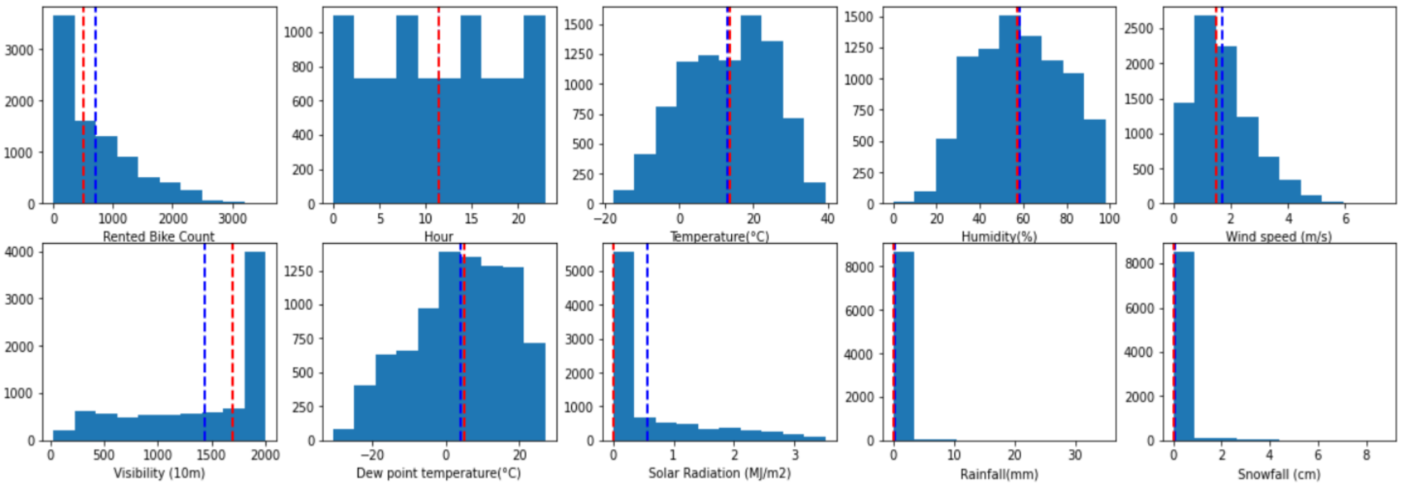
Data Summary

	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
460	20/12/2017	38	4	-8.5	73	1.0	1723	-12.4	0.00	0.0	2.0	Winter	No Holiday	Yes
7856	24/10/2018	2108	8	8.1	85	1.2	772	5.7	0.26	0.0	0.0	Autumn	No Holiday	Yes
4851	21/06/2018	394	3	20.0	83	1.7	607	17.0	0.00	0.0	0.0	Summer	No Holiday	Yes
835	04/01/2018	417	19	-2.7	39	1.4	1616	-14.7	0.00	0.0	0.0	Winter	No Holiday	Yes
2590	18/03/2018	70	22	8.1	76	0.8	492	4.1	0.00	0.0	0.0	Spring	No Holiday	Yes

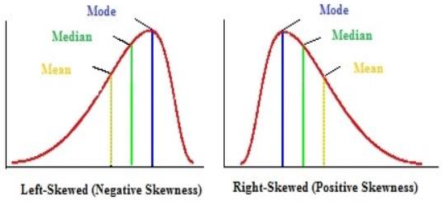
```
RangeIndex: 8760 entries, 0 to 8759
Data columns (total 14 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   Date                       8760 non-null   object
 1   Rented Bike Count          8760 non-null   int64
 2   Hour                       8760 non-null   int64
 3   Temperature(°C)            8760 non-null   float64
 4   Humidity(%)                8760 non-null   int64
 5   Wind speed (m/s)           8760 non-null   float64
 6   Visibility (10m)           8760 non-null   int64
 7   Dew point temperature(°C)  8760 non-null   float64
 8   Solar Radiation (MJ/m2)    8760 non-null   float64
 9   Rainfall(mm)               8760 non-null   float64
 10  Snowfall (cm)              8760 non-null   float64
 11  Seasons                    8760 non-null   object
 12  Holiday                    8760 non-null   object
 13  Functioning Day            8760 non-null   object
dtypes: float64(6), int64(4), object(4)
```

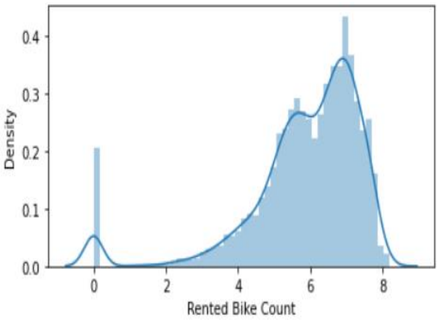
- This Dataset contain 8760 rows and 14 columns.
- There are No Missing Values present.
- There are No Duplicate values present.
- There are No null values.
- The dataset shows hourly rental data for one year from “December 01, 2017” to “November 31, 2018”.

Skewness Detection

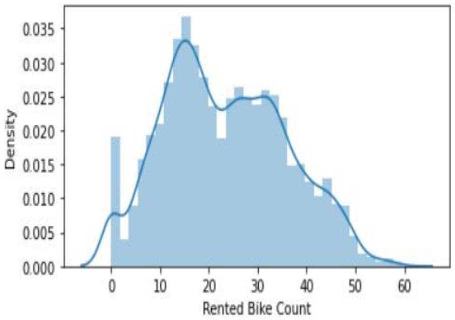


- Blue line represent mean value.
- Red line represent median value.

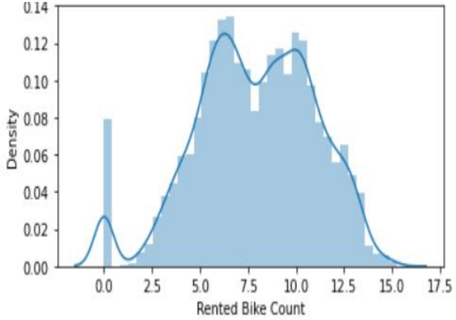




Log Transform



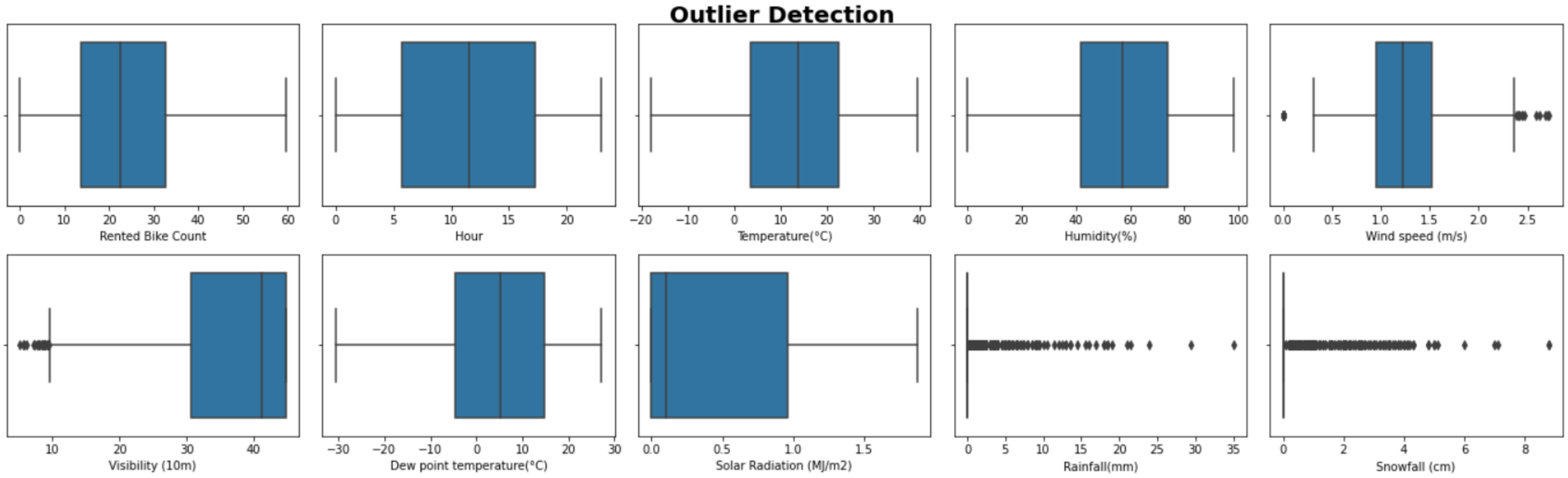
Square root Transform



Cube root Transform

- Removing skewness we used Square root Transform.

Outlier Detection

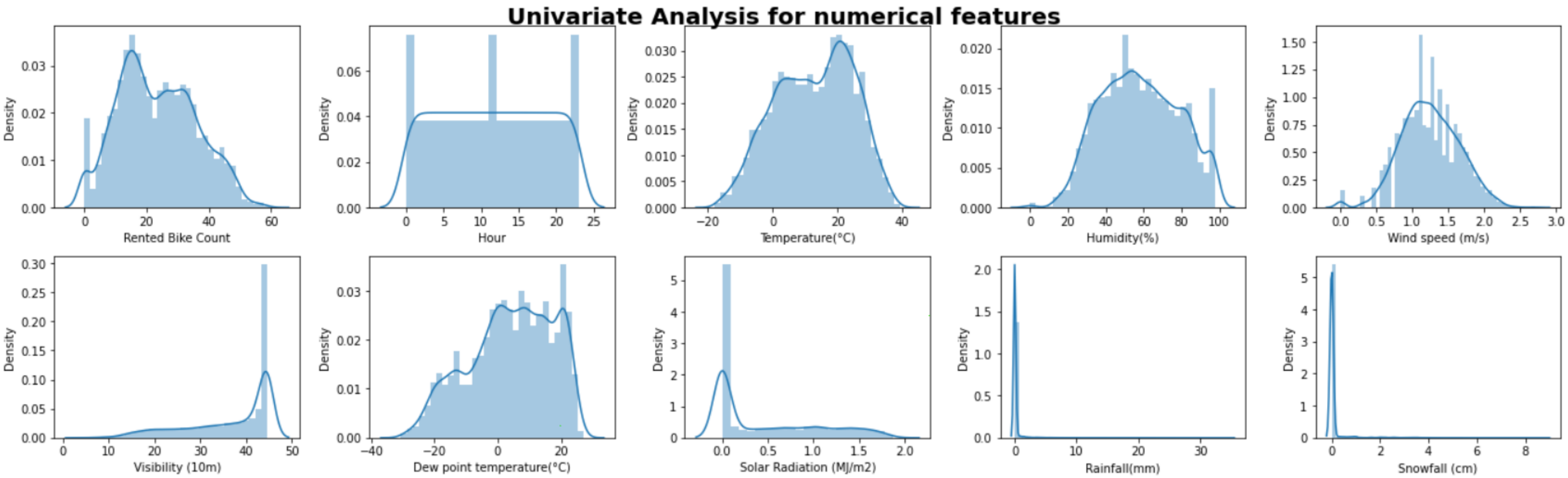


Data that we have is from "Seoul" city bike sharing data.

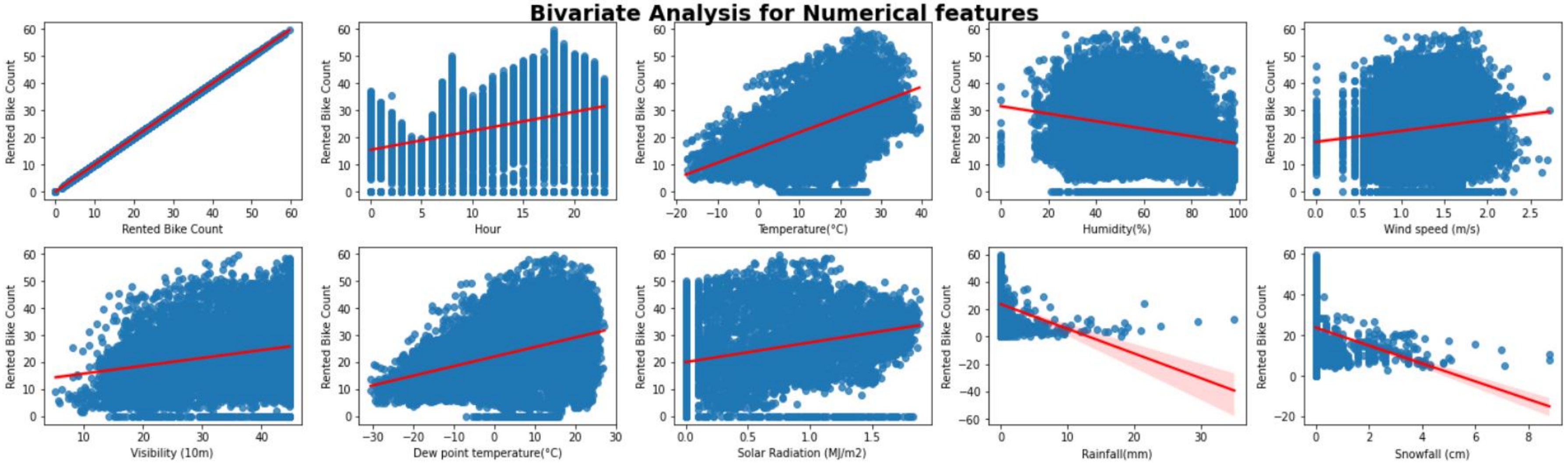
- ➢ Outlier present in Wind speed, Visibility, Rainfall and Snowfall.
- ➢ Considering Seoul city environmental aspects or weather condition it is possible to having these outlier.

So, we are keeping outlier data as it is.

Data Distribution



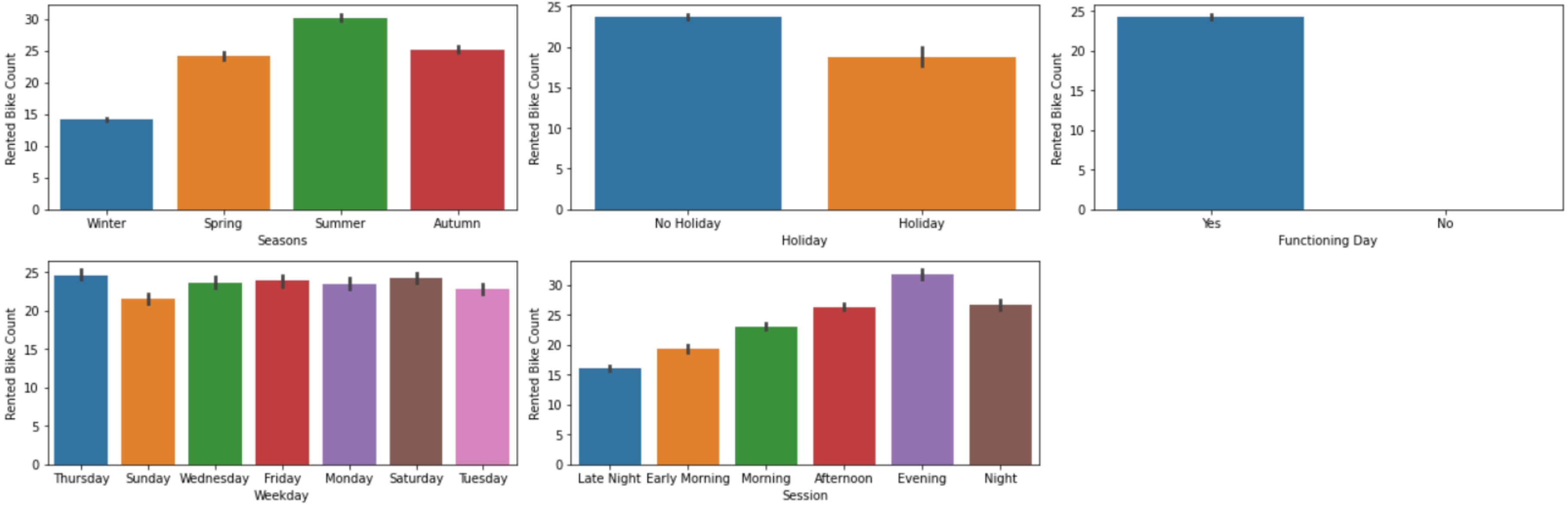
Bivariate analysis with respect to Target variable



- ➢ Temperature, Dew point temperature, Wind speed, Solar radiation are positively correlated with Target variable.
- ➢ Humidity, Rainfall, Snowfall are negatively correlated with target variable.

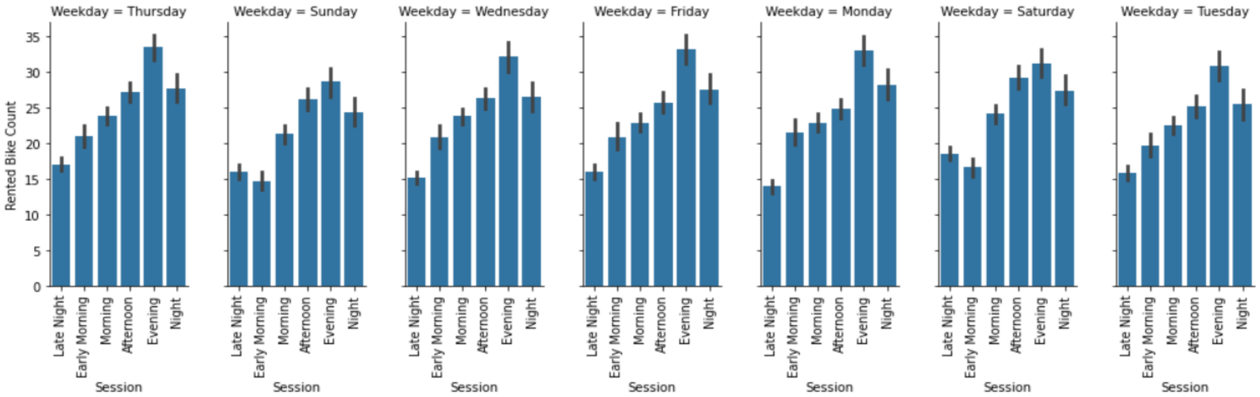
Bivariate Analysis

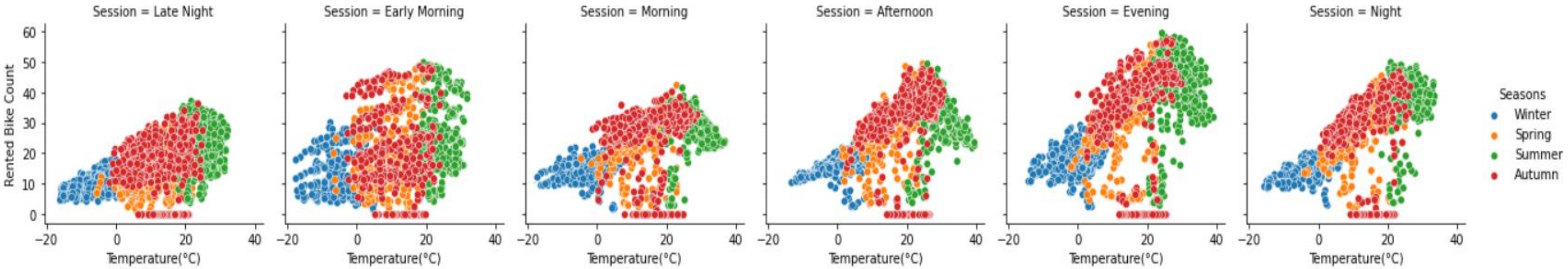
Bivariate analysis for categorical features



- Most of the rented bikes are count in summer season and lesser count in winter season, may be because of coldness or snowfall peoples are less prefer bike sharing during winter season.
- People prefer bike sharing mostly during No Holiday comparatively Holiday.
- Almost all people prefer bike sharing during Functioning days only.
- Most of the organization prefer Sunday as Holiday, that's why the Rented bike count less on Sunday.
- At evening high demand on bike sharing because of office leave time. Lesser bike count at Late Night.

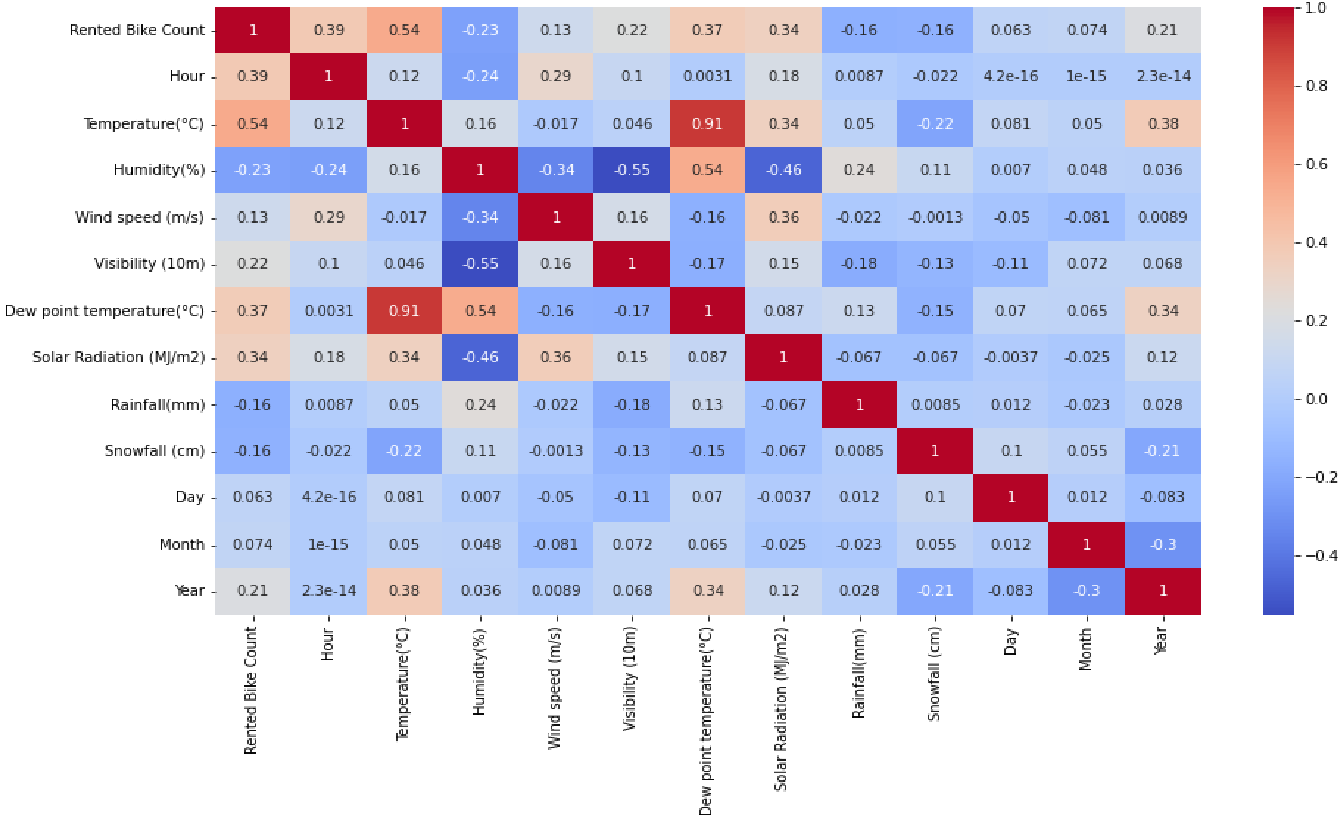
Multivariate Analysis

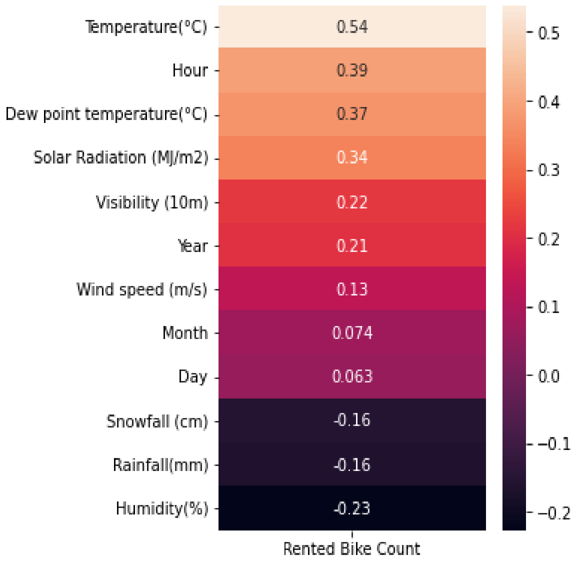




- ➢ Demand of bike got increase during Summer and Autumn season.
- ➢ People are less preferring bike during winter season.
- ➢ At evening high demand on bike sharing because of office leave time.
- ➢ High number of counted bike shows in between 20-35 Temperature.

Correlation Heatmap





- Multicollinearity allows us to look at correlations (that is, how one variable changes with respect to another). In words, **the statistical technique that examines the relationship and explains whether, and how strongly, pairs of variables are related to one another is known as correlation.**

Removing Multicollinearity

Using VIF analysis

	VIF Factor	features
0	4.415801	Hour
1	185.442727	Temperature(°C)
2	190.667081	Humidity(%)
3	13.770537	Wind speed (m/s)
4	26.986929	Visibility (10m)
5	125.836311	Dew point temperature(°C)
6	3.346340	Solar Radiation (MJ/m2)
7	1.104653	Rainfall(mm)
8	1.153595	Snowfall (cm)
9	4.423350	Day
10	4.715164	Month
11	457.179480	Year

Decision trees are not affected by multicollinearity.

- using VIF to remove multicollinearity from Numerical features.
- A **variance inflation factor (VIF)** provides a measure of multicollinearity among the independent variables in a linear regression model.
- Multicollinearity exists when there is a correlation between multiple independent variables in a multiple regression model.
- **Detecting multicollinearity is important** because -
  - The coefficient estimates can swing wildly based on which other independent variables are in the model. The coefficients become very sensitive to small changes in the model.
  - Multicollinearity reduces the precision of the estimated coefficients, which. weakens the statistical power of your regression model.

	VIF Factor	features
0	3.743836	Hour
1	3.028381	Temperature(°C)
2	7.036287	Humidity(%)
3	8.562949	Visibility (10m)
4	2.321899	Solar Radiation (MJ/m2)
5	1.086941	Rainfall(mm)
6	1.141712	Snowfall (cm)
7	3.926467	Day
8	4.644208	Month

$$VIF_i = \frac{1}{1 - R_i^2}$$

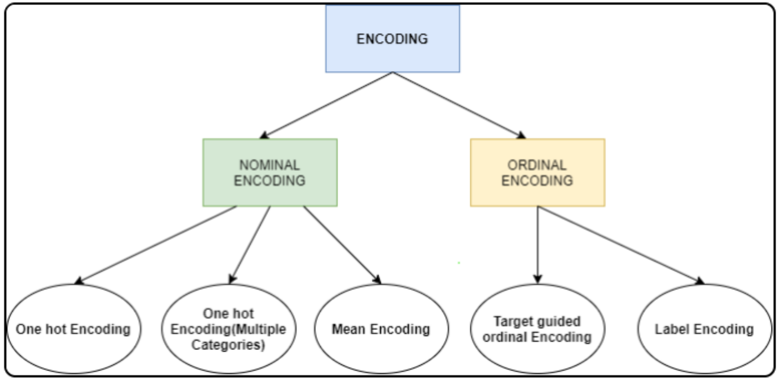
Encoding

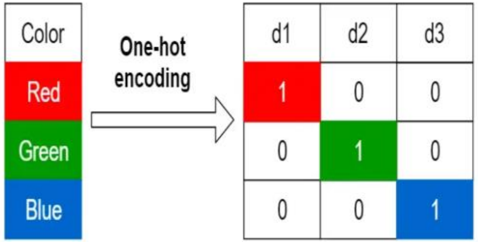
- Machine learning models require all input and output variables to be numeric. For this reason, **it is necessary to transform the categorical values of the relevant features into numerical ones**. This process is called feature encoding.

- If your data contains categorical data, you must encode it to numbers before you can fit and evaluate a model.

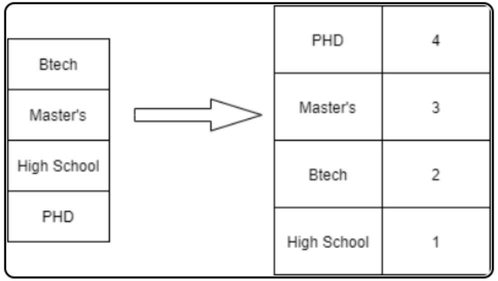
Mostly used two type of encoding-

- **One-Hot Encoding**
- **Label Encoding**



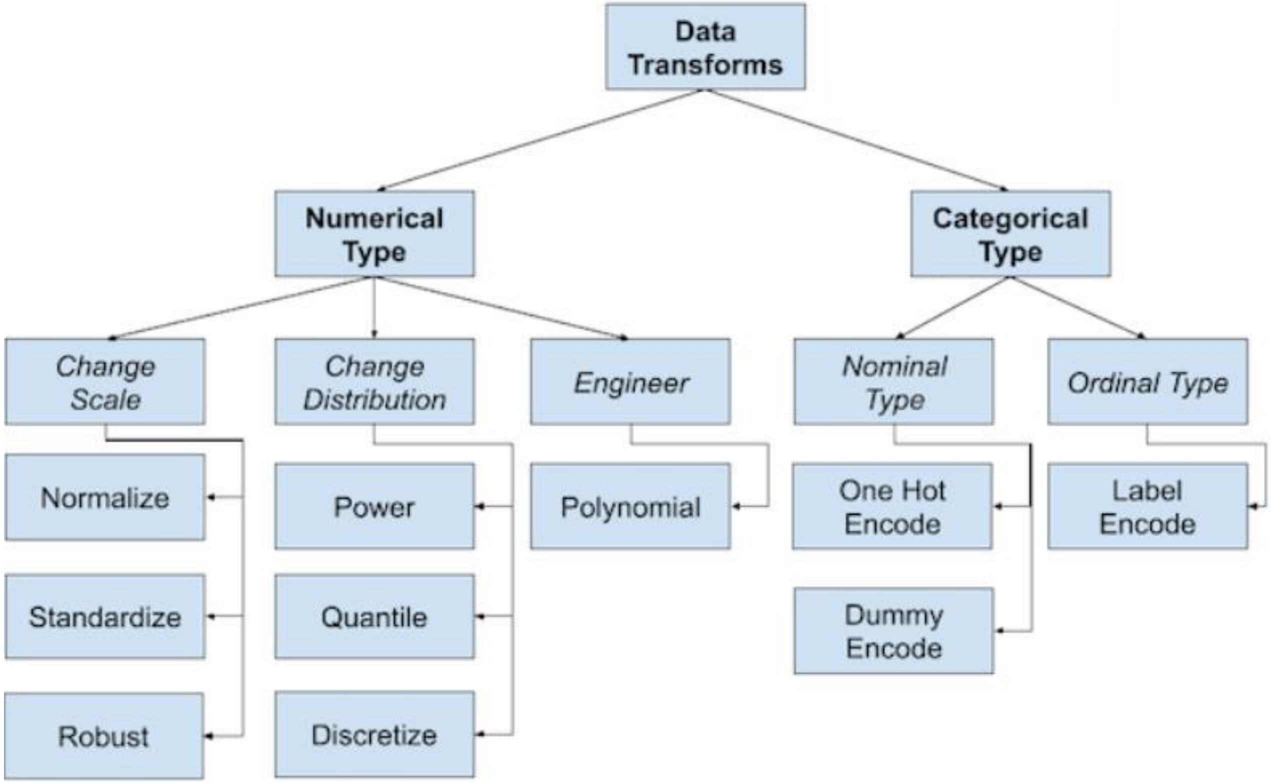



```
# Ordinal Encoding
df1['Functioning Day'] = df1['Functioning Day'].map({'Yes':1, 'No':0})

df1['Holiday'] = df1['Holiday'].map({'Holiday':1, 'No Holiday':0})

df1['Weekday'] = df['Weekday'].map({'Monday':1,'Tuesday':2,'Wednesday':3,'Thursday':4,'Friday':5,'Saturday':6,'Sunday':7})


# One hot encoding
df1 = pd.get_dummies(df1, columns=['Seasons'], drop_first=True)
```



Label Encoding

Overview of Data Transformation



Features that are provided to fit the ML model

```
X = df1[independent_features]
y = df1['Rented Bike Count']
```

Train Test Split

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=33)
```

Feature Scaling

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()

X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

Final Independent Features

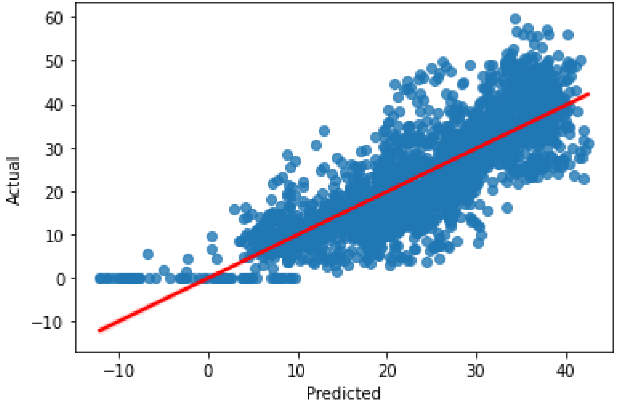
```
independent_features
```

```
['Hour',
 'Temperature(°C)',
 'Humidity(%)',
 'Visibility (10m)',
 'Solar Radiation (MJ/m2)',
 'Rainfall(mm)',
 'Snowfall (cm)',
 'Day',
 'Month',
 'Holiday',
 'Functioning Day',
 'Weekday',
 'Seasons_Spring',
 'Seasons_Summer',
 'Seasons_Winter']
```

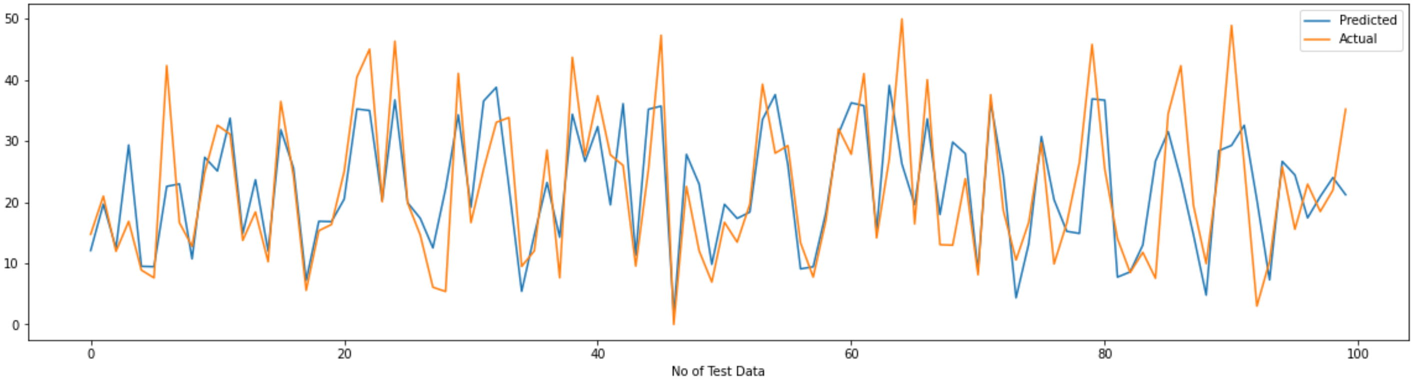
Linear Regression model

```
MSE : 52.55731926600699
RMSE : 7.249642699196078
R2 : 0.6538958268675931
Adjusted R2 :  0.6515078035939104
```

	Feature	Coefficient
0	Hour	3.509785
1	Temperature(°C)	5.163127
2	Humidity(%)	-2.595127
3	Visibility (10m)	0.303863
4	Solar Radiation (MJ/m2)	0.161833
5	Rainfall(mm)	-1.655532
6	Snowfall (cm)	-0.040651
7	Day	0.060297
8	Month	0.270051
9	Holiday	-0.613442
10	Functioning Day	5.273960
11	Weekday	-0.245119
12	Seasons_Spring	-1.243774
13	Seasons_Summer	-1.054131
14	Seasons_Winter	-3.392068

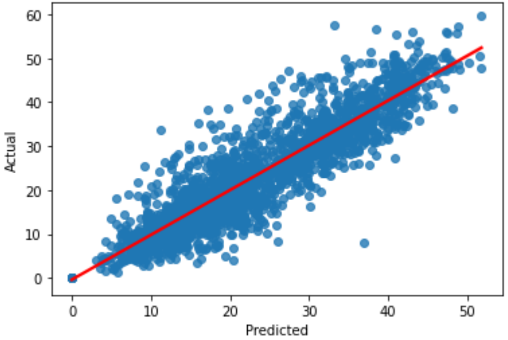


	y_actual	y_predict	error difference
1425	14.730920	12.077426	2.653494
7993	20.976177	19.668303	1.307874
897	11.958261	12.490552	-0.532291
5813	16.852300	29.342656	-12.490356
1708	8.888194	9.500206	-0.612012

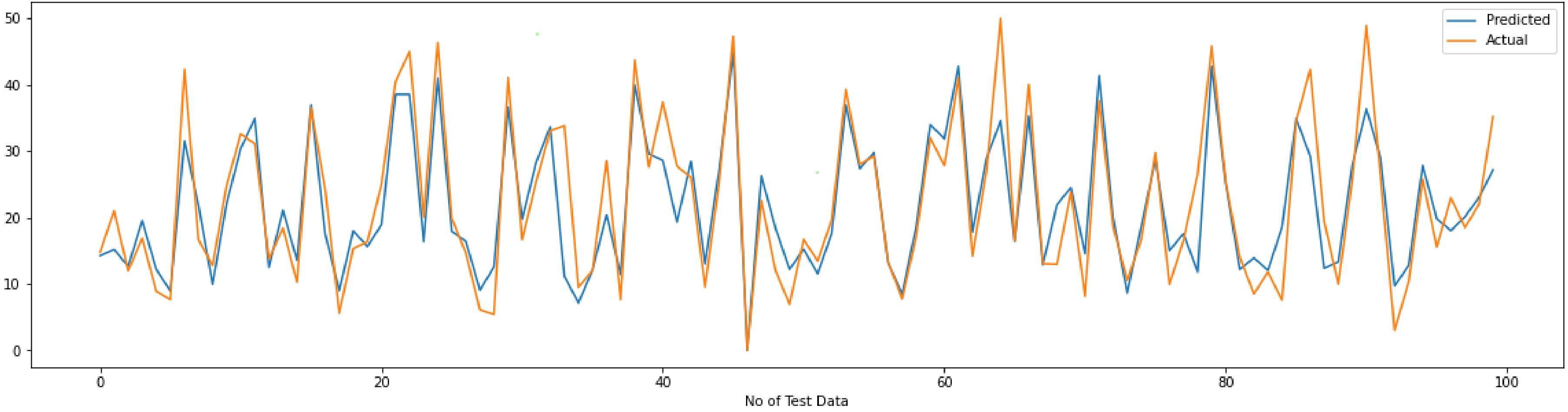


KNN Regressor model

```
train MSE : 20.44206970207864
train RMSE : 20.44206970207864
train R2 : 0.8690012166415065
train Adj R2 :  0.8687014025203016
-----------------------------------
test MSE : 25.657481880430726
test RMSE : 5.065321498229972
test R2 : 0.8310385370695705
test Adj R2 :  0.8298727496068491
```

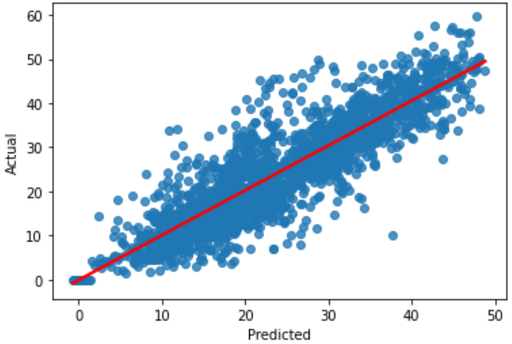


	1425	7993	897	5813	1708
y_actual	14.730920	20.976177	11.958261	16.852300	8.888194
y_predict	12.077733	19.668351	12.490658	29.342607	9.500175
error_difference	2.653187	1.307826	-0.532397	-12.490308	-0.611980

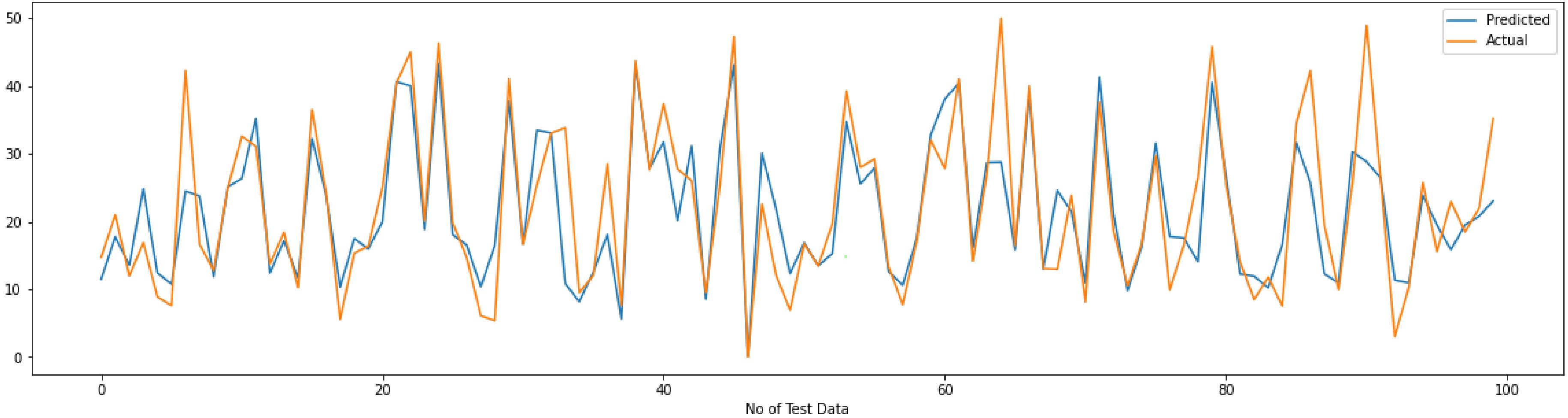


Support Vector Regressor model

```
train MSE : 30.4772849624247
train RMSE : 30.4772849624247
train R2 : 0.8046926114461996
train Adj R2 :  0.8042456155920179
-----------------------------------
test MSE : 30.04669338657633
test RMSE : 5.481486421270815
test R2 : 0.8021343912673642
test Adj R2 :  0.8007691731758326
```

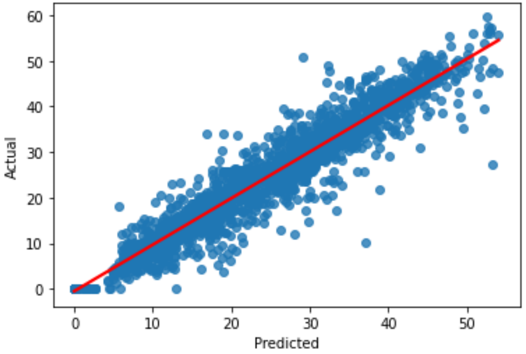


	1425	7993	897	5813	1708
y_actual	14.730920	20.976177	11.958261	16.852300	8.888194
y_predict	11.494308	17.737703	13.576893	24.801924	12.399290
error_difference	3.236611	3.238474	-1.618632	-7.949625	-3.511095

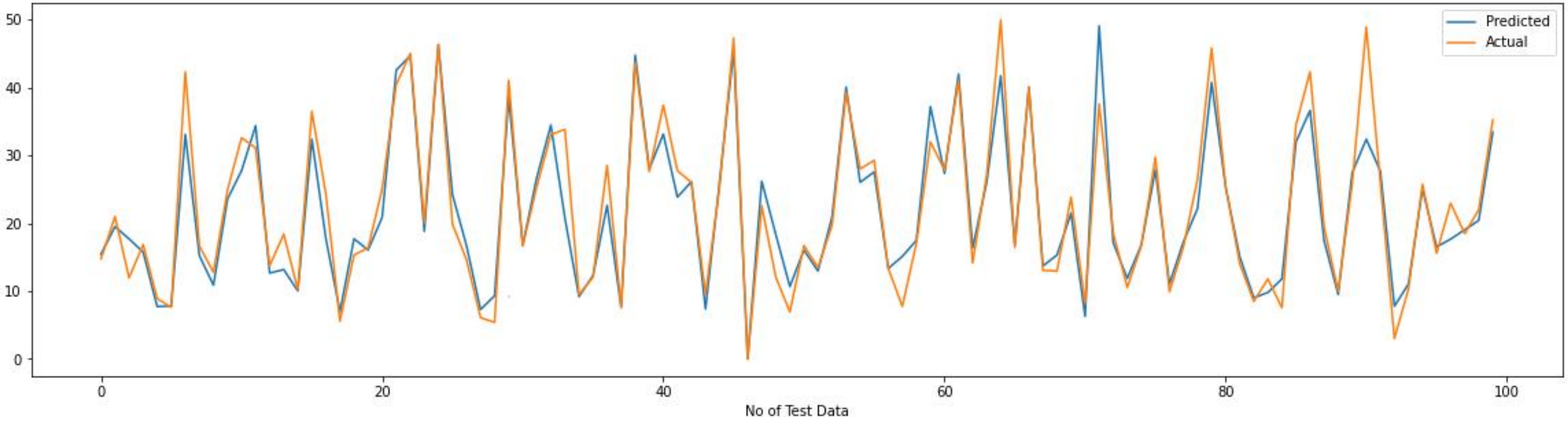


Random Forest Regressor

```
train MSE : 8.133627395065831
train RMSE : 8.133627395065831
train R2 : 0.9478773280507603
train Adj R2 :  0.9477580360032719
------------------------------------
test MSE : 13.17137325788206
test RMSE : 3.6292386609152696
test R2 : 0.9132629419821642
test Adj R2 :  0.9126644802203115
```

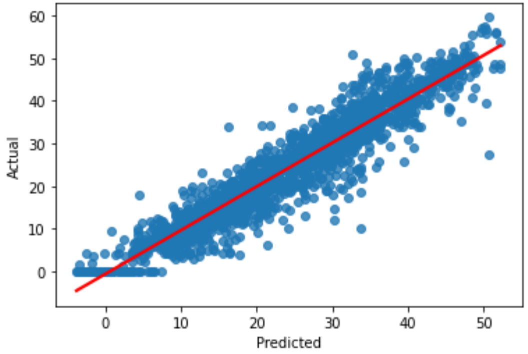


	1425	7993	897	5813	1708
y_actual	14.730920	20.976177	11.958261	16.852300	8.888194
y_predict	15.425562	19.497408	17.682382	15.748439	7.719125
error_difference	-0.694643	1.478769	-5.724121	1.103860	1.169069

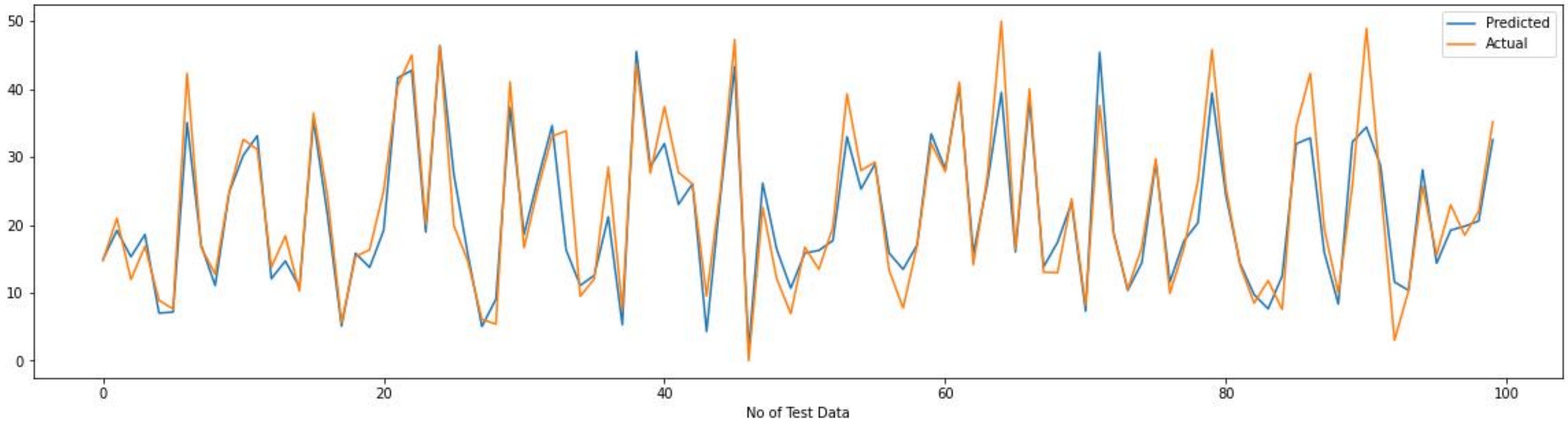


XGBoost Regressor model

```
train MSE : 12.992530047496613
train RMSE : 12.992530047496613
train R2 : 0.9167400535378442
train Adj R2 :  0.9165494982743513
-----------------------------------
test MSE : 14.588169318441341
test RMSE : 3.8194462057268646
test R2 : 0.903932956437139
test Adj R2 :  0.9032701203499988
```



	1425	7993	897	5813	1708
y_actual	14.730920	20.976177	11.958261	16.852300	8.888194
y_predict	14.874533	19.167187	15.301702	18.575001	7.006499
error_difference	-0.143613	1.808990	-3.343442	-1.722701	1.881696



Hyper Parameter Tunning

AI

- Hyperparameter tuning consists of **finding a set of optimal hyperparameter values for a learning algorithm while applying this optimized algorithm to any data set**.
- That combination of hyperparameters maximizes the model's performance, minimizing a predefined loss function to produce better results with fewer errors.

I decide to perform Hyper parameter tunning on the top three model which are-

1. Light-GBM
2. Random Forest
3. XGBoost

I performed Two method of Hyperparameter tunning-

1. **GridSearchCV()** – LGBM, XGBoost
2. **RandomizedSearchCV()** - RandomForest

- **Grid search** looks at every possible combination of hyperparameters to find the best model, **Random search** only selects and tests a random combination of hyperparameters.

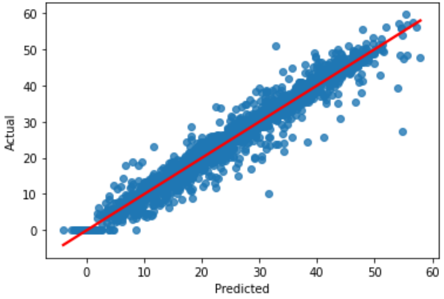
	Name	Train_R2_Score	Test_R2_Score	Test_RMSE_Score
10	Light-GBM	0.962296	0.936700	3.100372
7	RandomForest	0.948612	0.913242	3.629672
9	XGBRegressor	0.916740	0.903933	3.819446
4	DecisionTree	0.989836	0.873863	4.376582
5	KNeighborsRegressor	0.869001	0.831039	5.065321
6	SVR	0.804693	0.802134	5.481486
8	AdaBoostRegressor	0.695343	0.690261	6.858219
0	LinearRegression	0.650455	0.653896	7.249643
2	Ridge	0.650455	0.653895	7.249649
1	Lasso	0.649177	0.651581	7.273841
3	Elastic Net	0.646994	0.648617	7.304717

LGBM Regressor Tuning model

```
lgb_params = {'n_estimators': [900],
          'max_depth': [7,9],
          'min_samples_split': [4,5],
          'min_samples_leaf': [4,6],
          'loss': ['huber','rmse'],
          'learning_rate':[0.1]}
```

```
from sklearn.model_selection import GridSearchCV

lgb = LGBMRegressor()
lgb_gridsearch = GridSearchCV(estimator=lgb, param_grid=lgb_params, cv=5, verbose=2, n_jobs=-1)
lgb_gridsearch.fit(X_train,y_train)
```

```
Fitting 5 folds for each of 16 candidates, totalling 80 fits
GridSearchCV(cv=5, estimator=LGBMRegressor(), n_jobs=-1,
             param_grid={'learning_rate': [0.1], 'loss': ['huber', 'rmse'],
                         'max_depth': [7, 9], 'min_samples_leaf': [4, 6],
                         'min_samples_split': [4, 5], 'n_estimators': [900]},
             verbose=2)
```

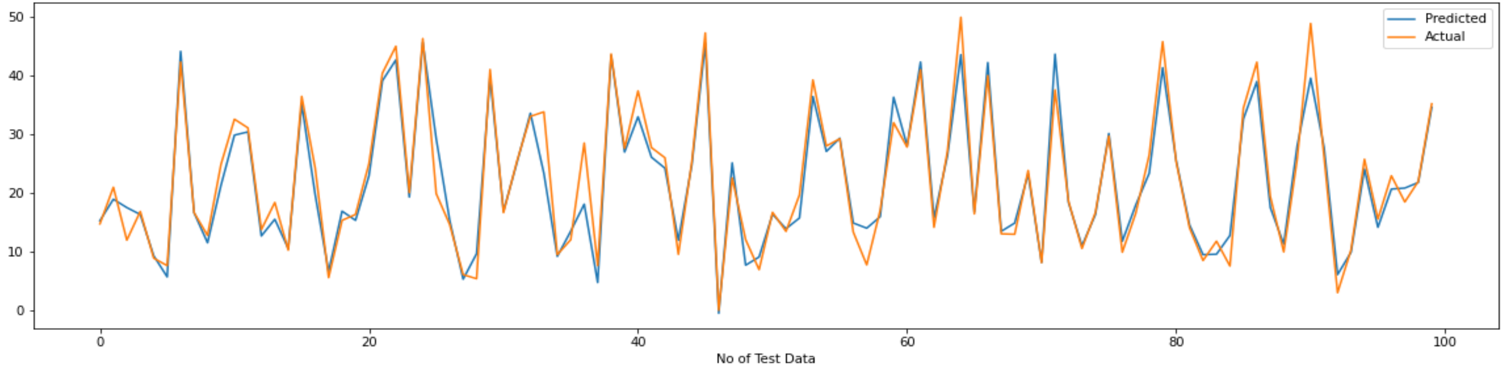



```
test MSE : 7.593878860925817
test RMSE : 2.7556993415330737
test R2 : 0.9499922522546101
test Adj R2 :  0.9496472125967532
train R2 : 0.9949148044209927
```

Hyperparameter tuning using Light Gradient Boosting algorithm

```
lgb_gridsearch.best_params_
```

```
{'learning_rate': 0.1,
 'loss': 'huber',
 'max_depth': 9,
 'min_samples_leaf': 4,
 'min_samples_split': 4,
 'n_estimators': 900}
```



Final Model Selection

- After performing Hyperparameter tuning on the top three model, performance of model got increased.
- **XGBRegressor_tuning** & **LGBMRegressor_tuning** are performed equally good; we can take any one of them as selection of final model.

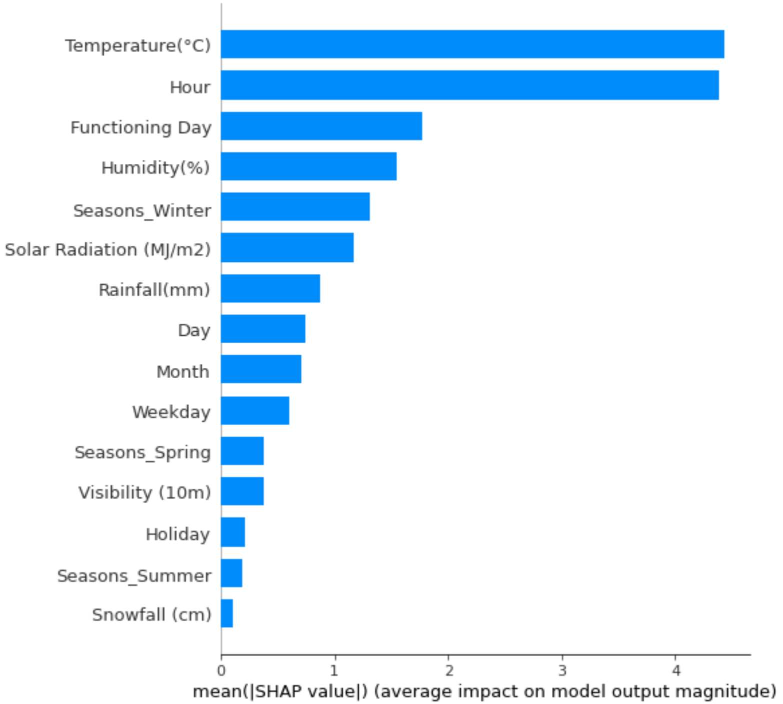
✓ We are selecting final model as ***LGBMRegressor_tuning.***

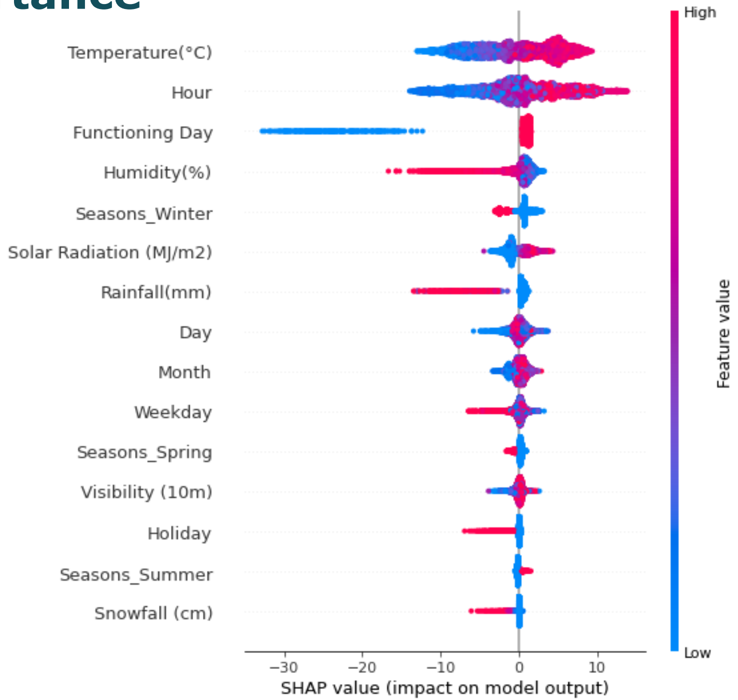
✓ because LGBM_tuning model has **lowest RMSE value** as well as **Highest R2 score** on the test data.

	Name	Train_R2_Score	Test_R2_Score	Test_RMSE_Score
2	LGBMRegressor_tuning	0.994915	0.949992	2.755699
1	XGBRegressor_tuning	0.998391	0.945458	2.877922
10	Light-GBM	0.962296	0.936700	3.100372
0	RandomForestRegressor_tuning	0.964946	0.920451	3.475603
7	RandomForest	0.948612	0.913242	3.629672
9	XGBRegressor	0.916740	0.903933	3.819446
4	DecisionTree	0.989836	0.873863	4.376582
5	KNeighborsRegressor	0.869001	0.831039	5.065321
6	SVR	0.804693	0.802134	5.481486
8	AdaBoostRegressor	0.695343	0.690261	6.858219
0	LinearRegression	0.650455	0.653896	7.249643
2	Ridge	0.650455	0.653895	7.249649
1	Lasso	0.649177	0.651581	7.273841
3	Elastic Net	0.646994	0.648617	7.304717

Model Explainability

Feature Importance





- ➢ Top variables contribute more to the model than the bottom ones and thus have high pred ictive power.

- ➢ **Feature importance**: Variables are ranked in descending order.
- ➢ **Impact**: The horizontal location shows whether the effect of that value is associated with a higher or lower prediction.
- ➢ **Feature value**: Color shows whether that variable is high (in red) or low (in blue) for that observation.

Project Summary

The project goal is to minimize the waiting time or make rental bike available at the right time, so company want to predict the bike count required at each hour for the stable supply.

- Given dataset have past records of **bike count during a day based** on weather condition, season, holiday and functional day.
- **Performing EDA** on data to understand how **independent variable behave with dependent variable.**
- **Checking multicollinearity** because after plotting heatmap correlation we understood there are two independent variables strongly corelate with each other, for that we **calculate the VIF** then dropping the one of the feature.
- For data preparation the **categorical features convert by using one hot encoding or label encoding.**
- Important **features selection** and building a models to **finalize final model based on r2 score and RMSE score** after that tune the hyperparameter.
- **Most Important feature** results have shown that **Temperature and Hour** of the day are most influence variable in hourly rented bike demand prediction.
- We get **best r2 score** and **low root mean square error** from **LGBM_tuning** model.
- To understand how affecting the feature for best model results we use **shaply for model explainability**.

Conclusion

Exploratory data analysis-

- **Working or Non-working Day -** We see 2 rental patterns across the day in bike rentals count.
  First for a Working Day where the rental count high at peak office hours (8am and 5pm).
  Second for a Non-working day where rental count is uniform across the day with a peak at around noon.

- **Temperature:** People generally prefer to bike at moderate to high temperatures. We see highest rental counts between 30 to 35 degrees Celsius.

- **Hour:** Demand of rental bike is high at Evening time in between 4PM to 8PM.

- **Weather:** As one would expect, we see highest number of bike rentals on a clear day and the lowest on a snowy or rainy day.

- **Season:** Demand of rental bikes is high during Summer and Autumn season.

- **Humidity:** With increasing humidity, we see decrease in the number of bike rental count.

From the above operations we are conclude that for LGBM(Light Gradient Boosting Algorithm) model performing very well than the other models. So, in future if we want to do some prediction with this data then the LGBM model will fit perfectly to do the prediction with the highest R2 score and lowest RMSE value.


Thank You