Capstone Project - 2

BIKE SHARING DEMAND PREDICTION

BY

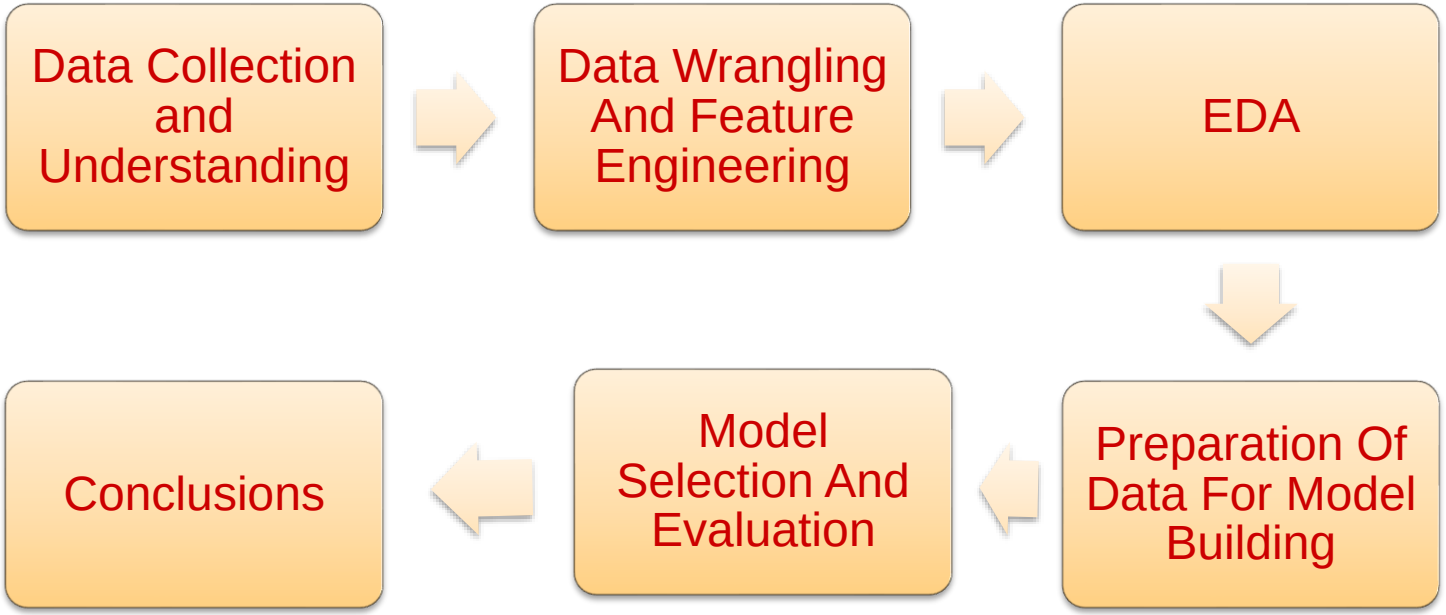
- Sumit Berde
- Omkar Desai

Problem Statement

- Rental bikes is a multi-billion dollar industry whose popularity is increasing among the masses for short distance travelling within the cities.
- To keep up with the demand, it is important to predict the demand for rented bikes per hourly basis to increase customer satisfaction by reducing the waiting time thus giving an edge over other competitors.
- The main aim of the project is to make prediction of rented bikes required at each hour for the stable supply of rented bikes.

Workflow

The steps involved are as follows:-



Data Collection And Understanding

- We had Seoul Bike Data for our analysis and model building
- The dataset contains weather information (Temperature, Humidity, Wind speed, Visibility, Dew point, Solar radiation, Snowfall, Rainfall), the number of bikes rented per hour and date information.
- In this we had total 8760 observations and 14 features including target variable.

Data Description:

- Date : year-month-day.
- Hour – Hour of the 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 – Yes/No
- Rented Bike count - Count of bikes rented at each hour (Target Variable i.e. Y variable).

Data Wrangling And Feature Engineering

Categorical features : Seasons, Holiday and Functioning day
Numerical features : Hour, Temperature, Humidity, Wind speed, Visibility, Dew point temperature, Solar radiation, Rainfall, Snowfall, Rented Bike count .

1. Renaming columns : We renamed columns because they had units mentioned in brackets and contained spaces in between.

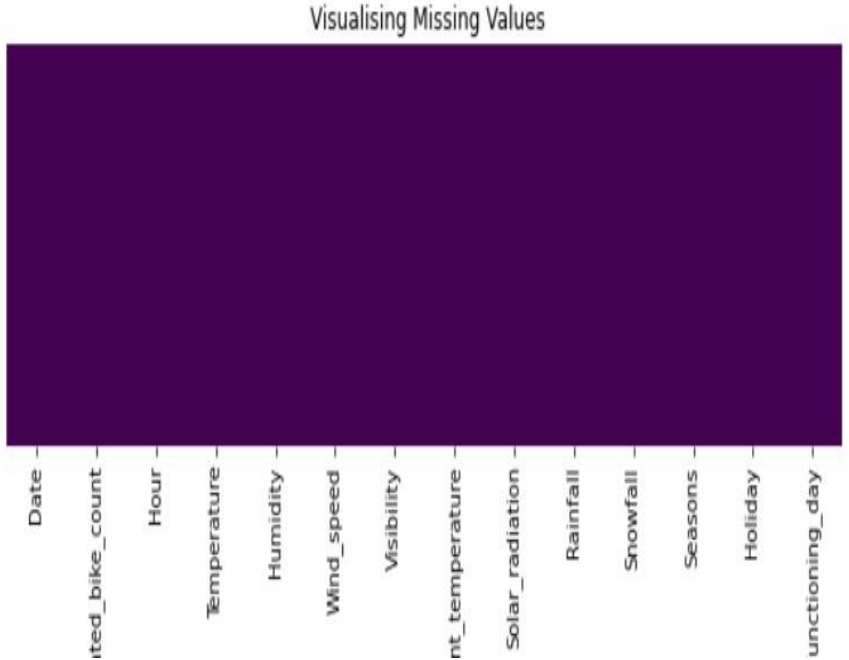
```
Index(['Date', 'Rented_bike_count', 'Hour', 'Temperature', 'Humidity',
       'Wind_speed', 'Visibility', 'Dew_point_temperature', 'Solar_radiation',
       'Rainfall', 'Snowfall', 'Seasons', 'Holiday', 'Functioning_day'],
      dtype='object')
```

2. Duplicate rows : We had zero duplicate rows

3. Nan values : We had zero missing values

```
# Missing Values/Null Values
df.isnull().sum()
```

```
Date                       0
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
```



4. Categorical value counts : Lets check the value counts for each categorical feature

```
# Checking value counts for categorical columns
df['Seasons'].value_counts()
```

```
Spring    2208
Summer    2208
Autumn    2184
Winter    2160
Name: Seasons, dtype: int64
```

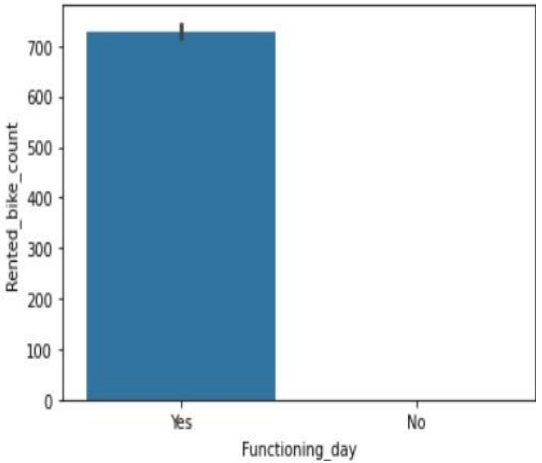
```
# Checking value counts for categorical columns
df['Holiday'].value_counts()
```

```
No Holiday    8328
Holiday        432
Name: Holiday, dtype: int64
```

```
# Checking value counts for categorical columns
df['Functioning_day'].value_counts()
```

```
Yes    8465
No      295
Name: Functioning_day, dtype: int64
```

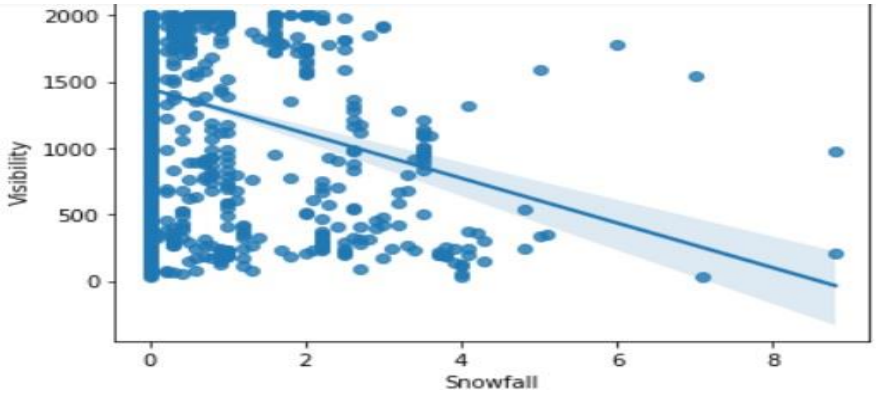
- If we group by Functioning day against rented_bike_count, we will observe that on non-functioning day zero bikes were rented.
- It is of no use to us as we already know that on non-functioning days zero bikes will be rented
- Dropping those 295 rows as we only want data for only functioning days. Then dropping ‘Functioning day’ feature as it has only one category which is of no use.



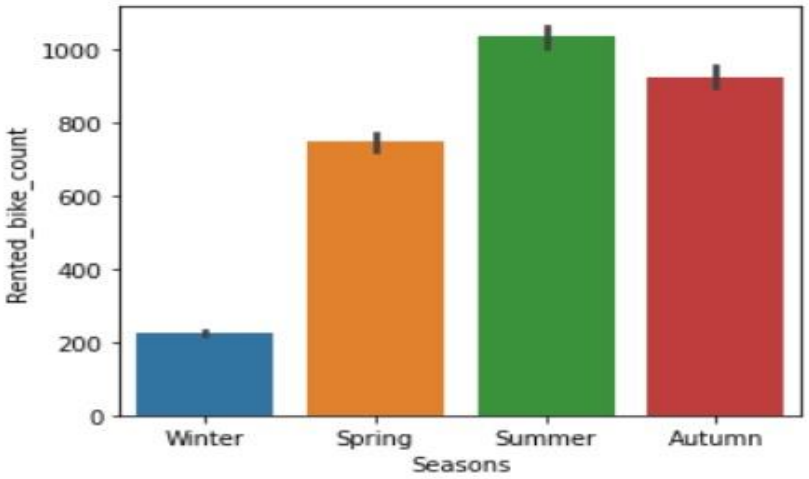
5. Feature Engineering : We have derived 3 extra features for analysis

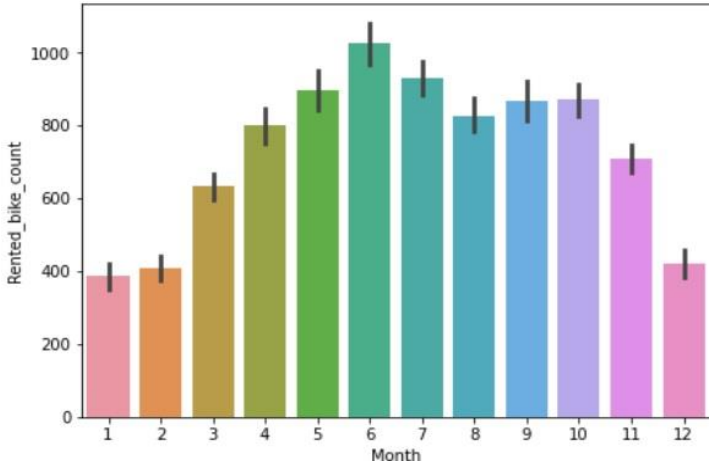
- Month : Extracted from ‘Date’ feature. Has month numbers from 1 to 12.
- Is weekend : Extracted from ‘Date’ feature. Tells whether it is weekend or not. “0” means not a weekend, “1” means weekend.
- Binned visibility : “Visibility” and “Snowfall” have a relationship that if snowfall is heavy then visibility is low which leads to less rental bike count. Similarly if the snowfall is less then visibility is high which increases rental bike count. Hence visibility is binned into Heavy snowfall, Medium snowfall and Low snowfall.Then we have dropped “Date", "Visibility” and “Snowfall” feature.

```
def visibility_binning(x):
 '''Creates bins for visibility feature'''
 try:
    if x['Visibility']<=500:
      return 'Heavy_snow'
    elif x['Visibility']<1000:
      return 'Medium_snow'
    else:
      return 'Light_snow'
 except:
    print('Check your code')
```



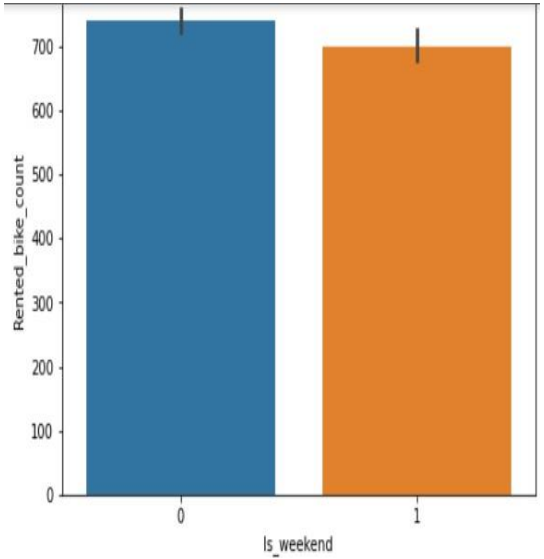
EDA(Exploratory Data Analysis)

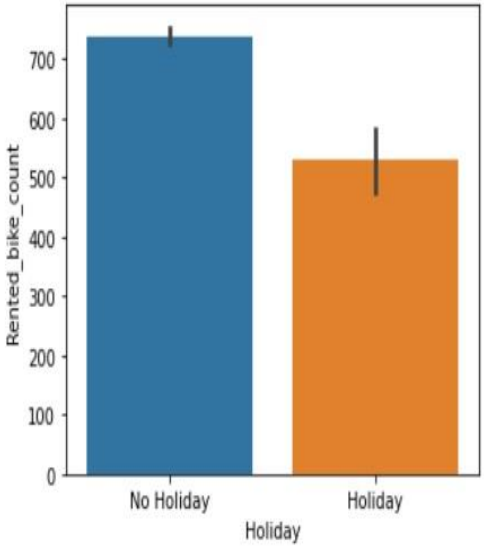


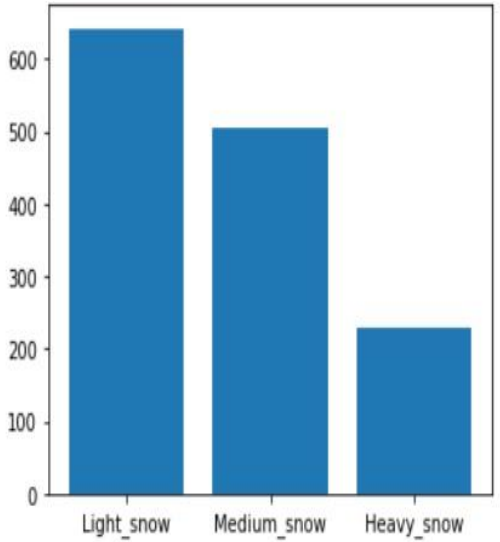


Observations:

- In summer season, most bikes were rented and in winter season least bikes were rented. It seems like people don't like to rent bikes in winter season due to snowfall and rainfall.
- From March bike rent count increases and it was highest in June.

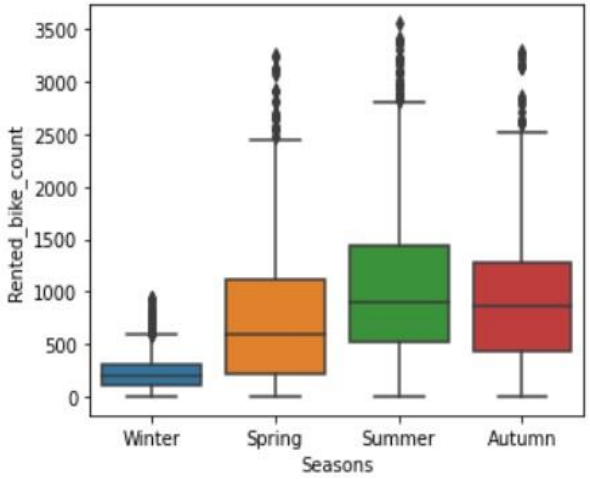


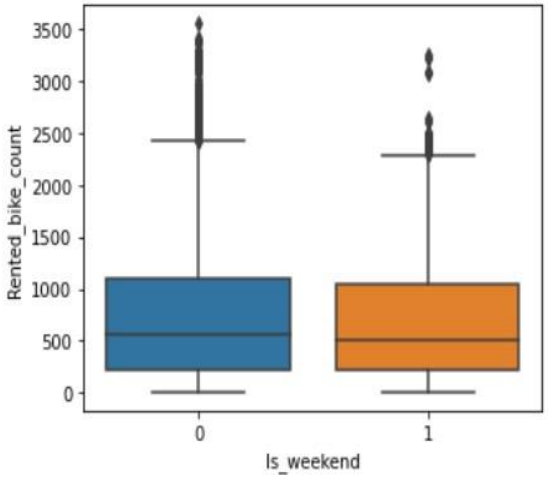


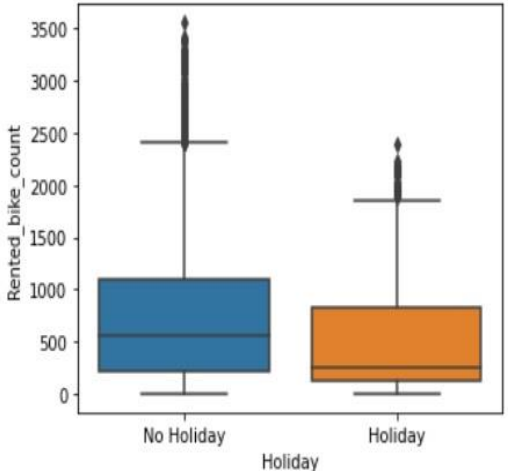


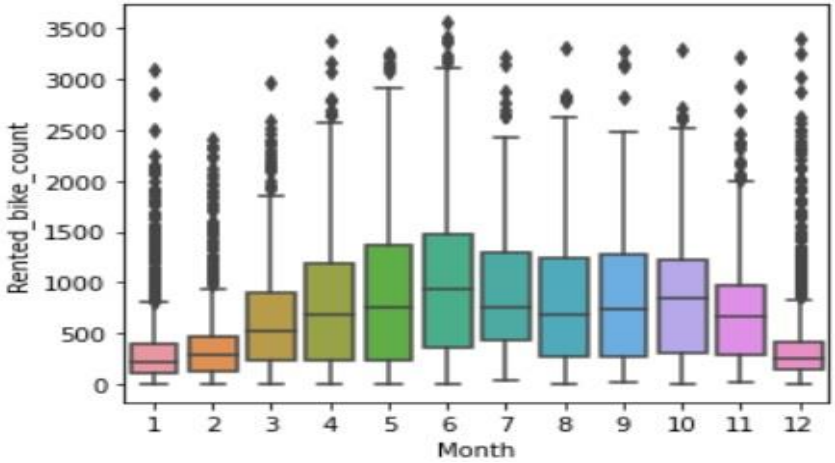
Observations:

- Bike rented were more or less same on weekdays and weekends.
- Bike rented on “No Holiday” were more than on “Holiday”. One of the reason may be as people travel to reach their offices. This gives an indication to open bike rental outlets near railway stations where people need to travel further using buses from railway stations to reach their offices.
- During heavy snowfall, visibility is less hence rented bike count is less.



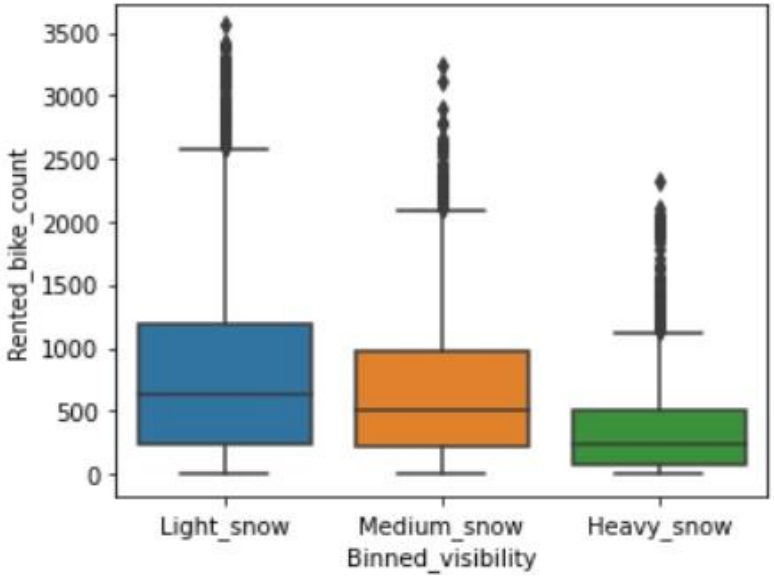


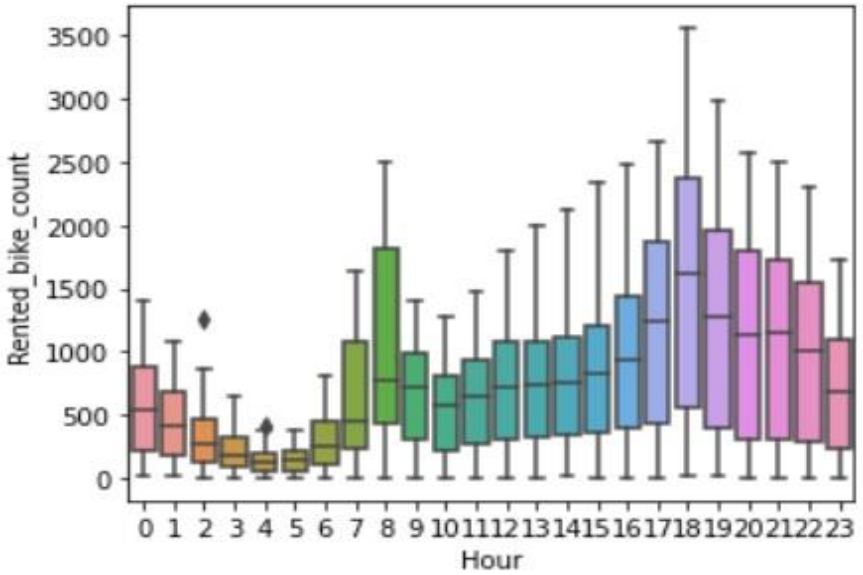




Observations:

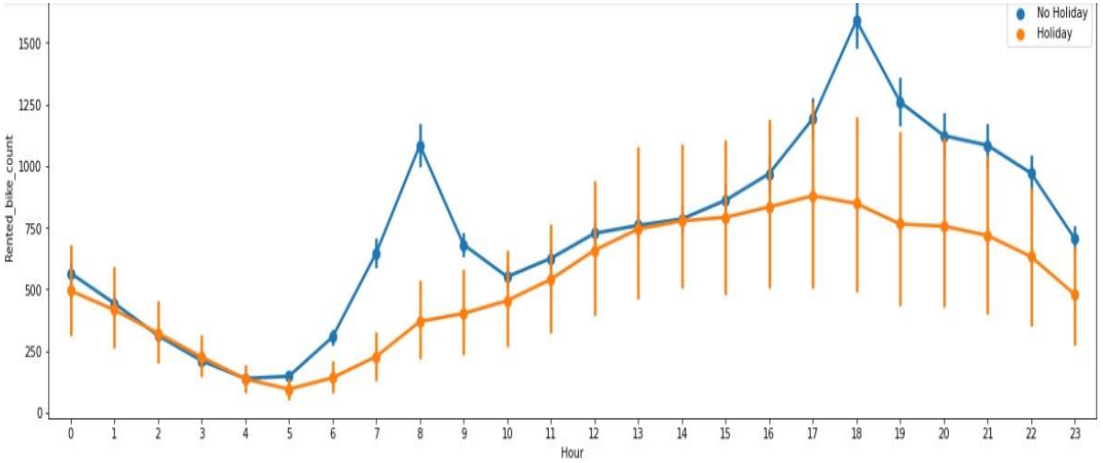
- Every categorical features has outliers.
- Interestingly December month has large number of outliers.

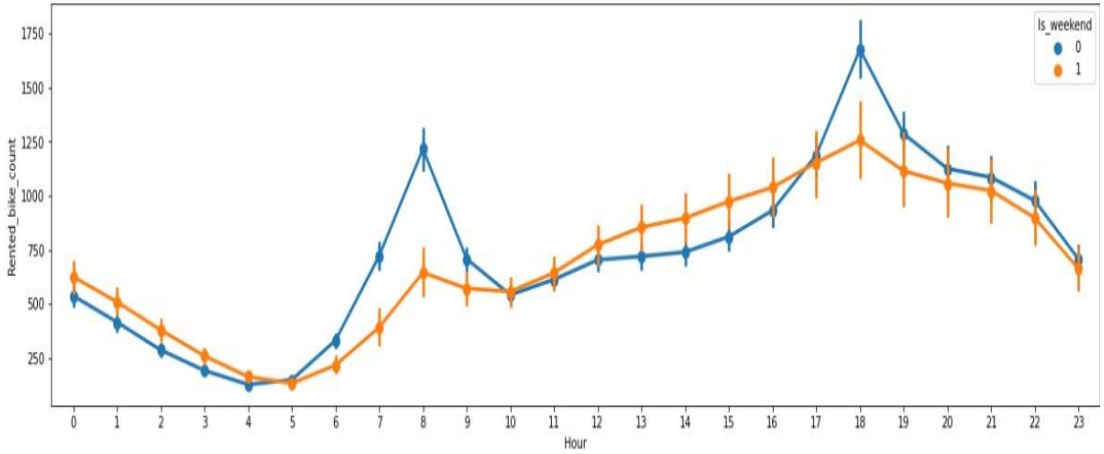




Observations:

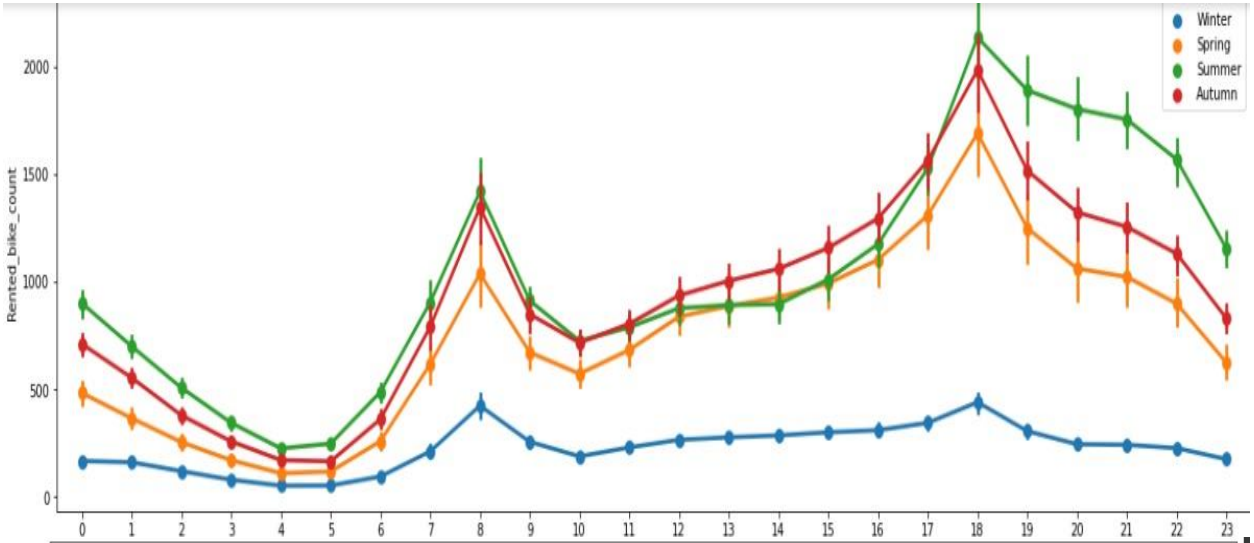
- Binned visibility feature has some outliers.
- Hour feature does not have any outliers.

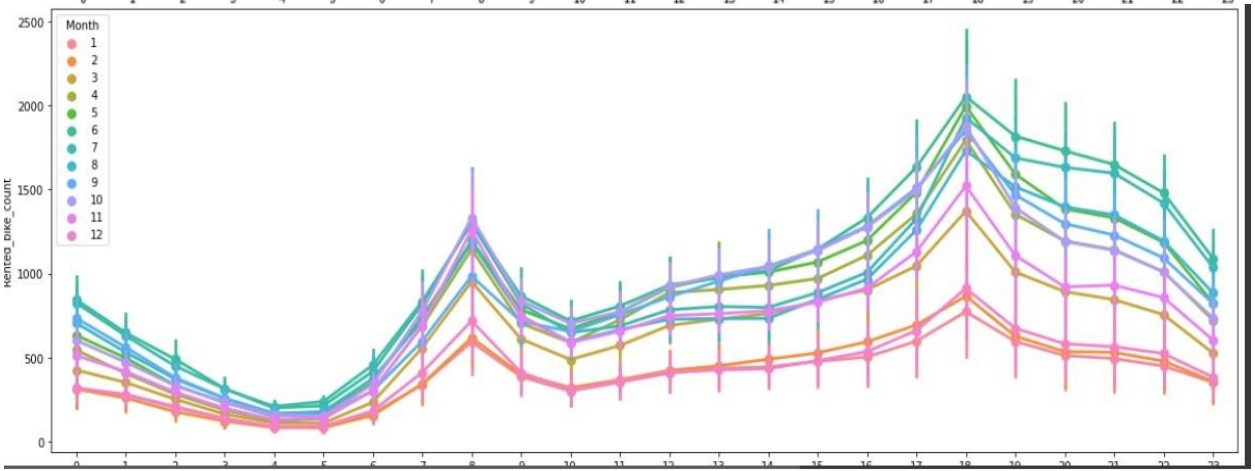




Observations:

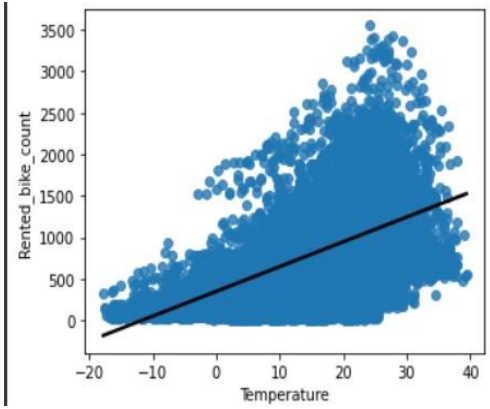
- It is clear that maximum bikes are rented during office opening(7AM to 9AM) and closing hours(5PM to 7PM).
- Maximum bikes are rented during night time from 5PM to 10PM.

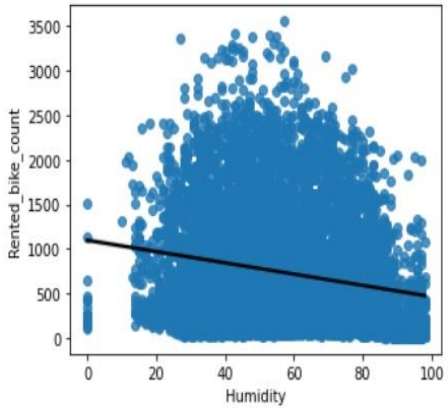


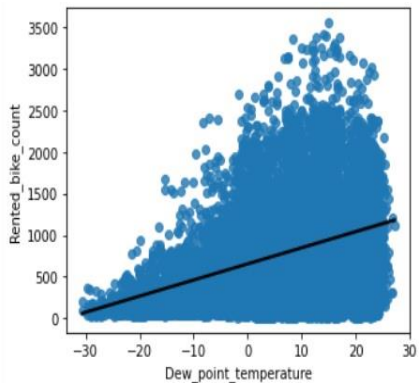


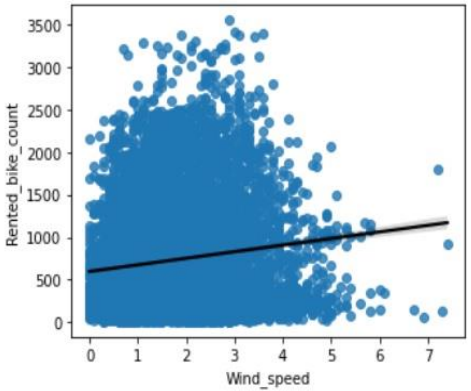
Observations:

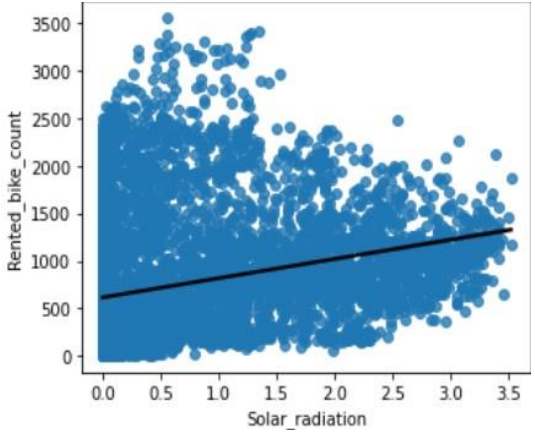
- Best time for rental business is during summer season during night time from 5PM to 10PM
- Same trend as observed in different seasons, holidays and is_weekend can be seen across months that is peak during night time.

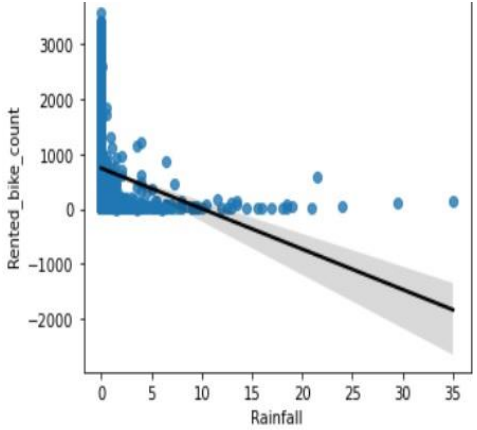






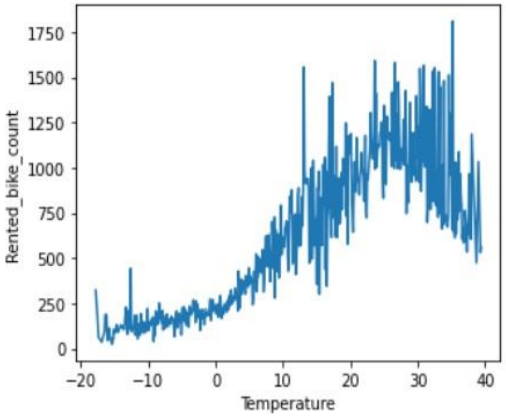


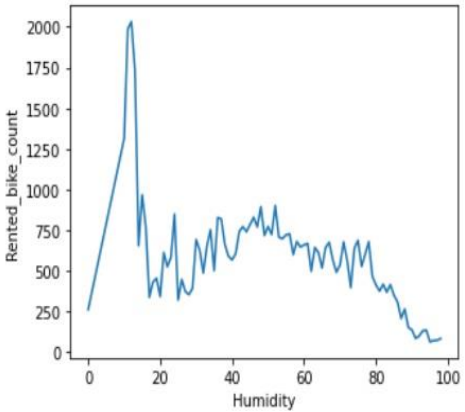


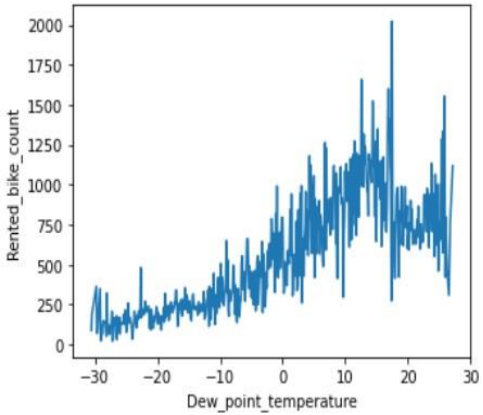


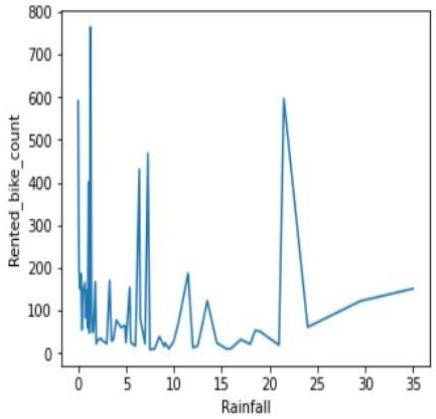
Observations:

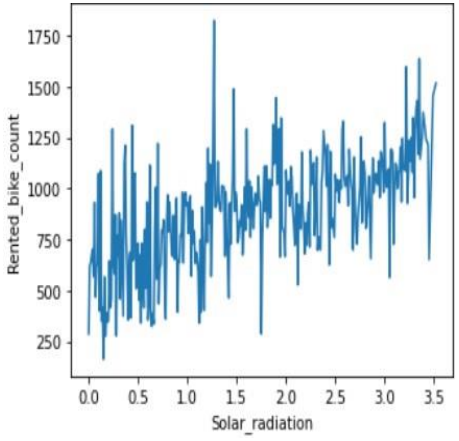
- Temperature and Dew point temperature show positive correlation with rented bike count.
- Rainfall and Humidity show negative correlation with rented bike count. Humidity is little less correlated with rented bike count.
- Solar radiation and wind speed are not strongly correlated with rented bike count.

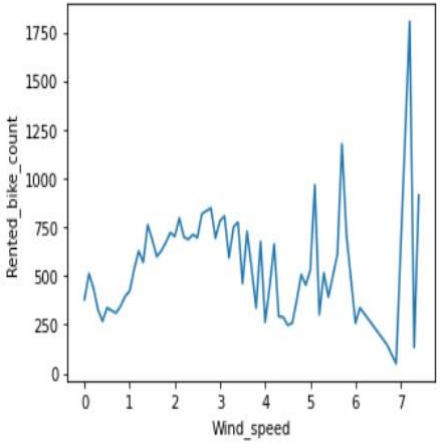








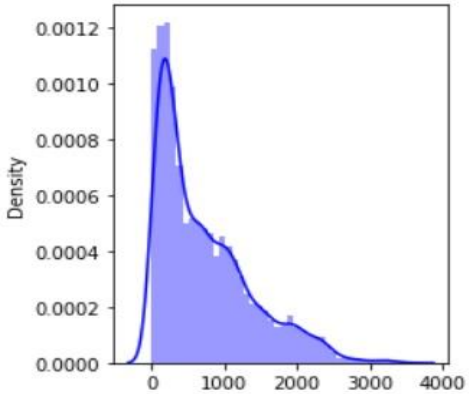


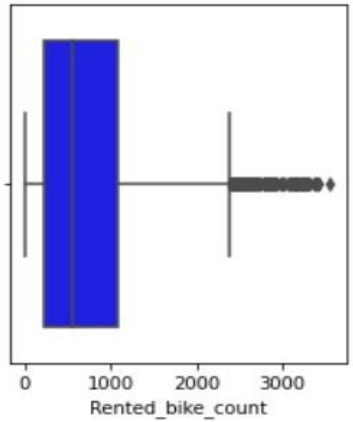


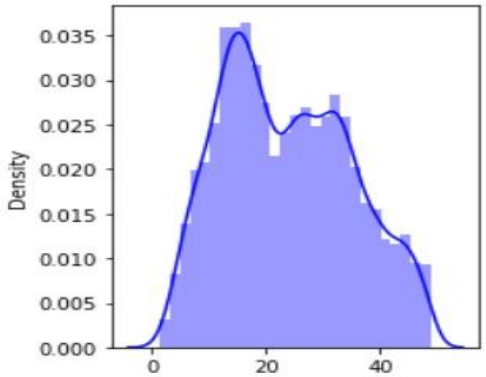
Observations:

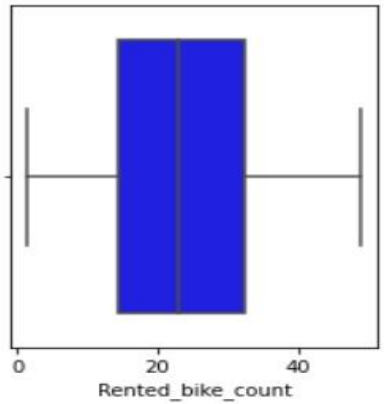
- People prefer temperature ranges of (20*C to 30*C) for renting bikes. Very cold temperatures has less rented bike counts. After 30*C there is a decreasing trend.
- Humidity and Rainfall has a decreasing trend.
- There is no such significant relationship between rented bike count and solar radiation, windspeed

Outliers Detection : Lets check the distribution of target feature rented bike count









Observations:

- Rented bike count is highly right skewed. Hence to remove outliers we have used IQR technique.
- In order to make rented bike as approx normal distribution we have took square root of the feature.

```
# Removing outliers based on IQR as it is a skewed distribution

# Calucating IQR
IQR = df['Rented_bike_count'].quantile(0.75)-df['Rented_bike_count'].quantile(0.25)

# Calculating Lower and Upper bounds
lower_bridge = df['Rented_bike_count'].quantile(0.25) - IQR*(1.5)
upper_bridge = df['Rented_bike_count'].quantile(0.75) + IQR*(1.5)

# Removing outliers
df = df[df['Rented_bike_count'] <= upper_bridge]
```

Preparation Of Data For Model Building

1. Categorical Encoding : We have 3 encoding techniques

- One Hot Encoding : To encode more than 2 nominal categories in a feature this method is used. To avoid the problem of dummy trap one column is dropped to reduce multi-collinearity. In “Season” feature we have used one hot encoding.
- Binary Label Encoding : To label yes or no types of values in a feature. In “Holiday” feature we have used binary label encoding.
- Label Encoding : To encode ordinal categories in a feature, this method is used. In “Binned_visibility” feature we have used label encoding.

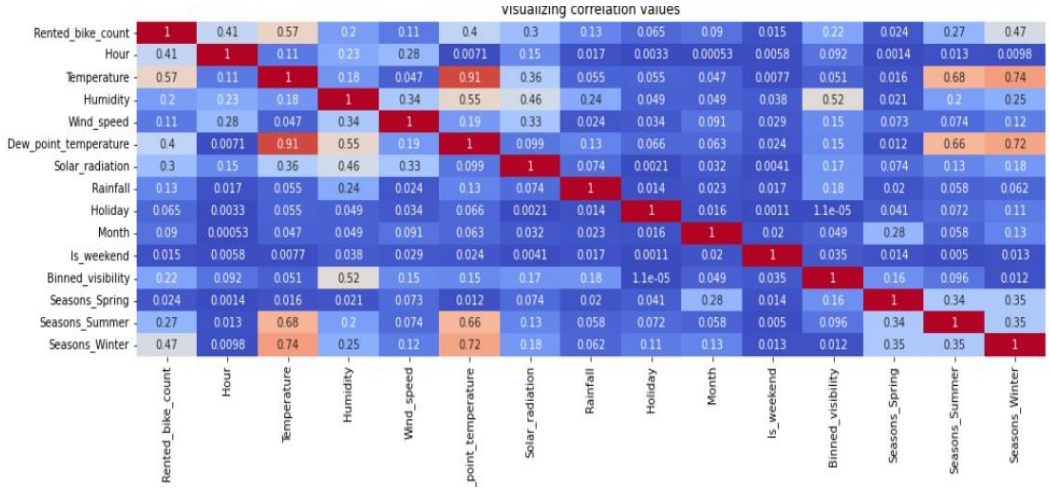
```
# Encode your categorical columns

# Creating One_Hot_Encoding for 'Seasons' feature
df=pd.get_dummies(df,columns=['Seasons'],prefix='Seasons',drop_first=True)

# Using binary label encoding for 'Holiday' feature. Holiday=1 and No_holiday=0
df['Holiday']=df['Holiday'].map({'No Holiday':0, 'Holiday':1})

# Using label encoding for 'binned_visibility' feature. Light_snow=2,Medium_snow=1,Heavy_snow=0
df['Binned_visibity']=df['Binned_visibity'].map({'Light_snow':2,'Medium_snow':1,'Heavy_snow':0})
```

2. Feature Selection: For feature selection, we have 2 methods Pearson's Correlation Coefficient and Variance Inflation Factor(VIF)



	variables	VIF
0	Hour	3.912488
1	Temperature	7.816929
2	Humidity	6.690061
3	Rainfall	1.090155
4	Holiday	1.074127
5	Binned_visibity	5.195621
6	Month	4.691074
7	Is_weekend	1.403837
8	Seasons_Spring	2.077006

Observations:

- “Dew point temperature” has high correlation with “Temperature”.
- “Solar radiation” has high correlation with “Humidity”.
- Also removing all those features which as VIF greater than 8.
- Hence from both these methods we have decided to drop dew point temperature, solar radiation and wind speed.

3. Data Splitting : 30% data is used for testing and 70% data for training to strike a balance between bias and variance

```
# Split your data to train and test. Choose Splitting ratio wisely.
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X.values, y.values, test_size=0.3, random_state=7)
```

4. Data Scaling : Standard scaler is used for scaling to get data to range from -3 to 3 to get approx normal distribution.

```
[ ]  # Scaling your data
     from sklearn.preprocessing import StandardScaler

     stn_scalar = StandardScaler()
     X_train= stn_scalar.fit_transform(X_train)
     X_test = stn_scalar.transform(X_test)
```

Model Selection And Evaluation

Grid Search CV and Bayesian Optimization are the two hyper-parameter tuning methods that were used

The various models that were used were

- Linear Regression
- Lasso Regularization(alpha=0.01)
- Ridge Regularization(alpha=0.4)
- Decision Tree(criterion=Friedman mse, max depth=10, min impurity decrease=41.99, min samples leaf=20, min samples split=43)
- Random Forest(n estimators=50, max depth=10, min impurity decrease=41.99, min samples leaf=20, min samples split=20)
- Gradient Boosting(n estimators=300, learning rate=0.0957, loss=squared error, max depth=20, min impurity decrease=2, min samples leaf=50, min samples split=50)
- XG Boosting(n estimators=284, eta=0.0387, gamma=2, max depth=8, subsample=0.9, col sample bytree=0.834)

Train Data frame

	Model	MSE	RMSE	R2_score
0	Linear Regression	150712.15587	388.21664	0.56502
1	Lasso CV	150800.61602	388.33055	0.56476
2	Ridge CV	150713.85665	388.21883	0.56501
3	Decision Tree CV	52885.52356	229.96853	0.84736
4	Random Forest CV	49629.37027	222.77650	0.85676
5	Gradient Boosting Regressor CV	18001.34065	134.16907	0.94804
6	XGB Regressor CV	2492.63550	49.92630	0.99281

Test Data frame

	Model	MSE	RMSE	R2_score
0	Linear Regression	146672.04177	382.97786	0.57160
1	Lasso CV	146789.67023	383.13140	0.57126
2	Ridge CV	146674.68771	382.98132	0.57159
3	Decision Tree CV	64149.08377	253.27669	0.81263
4	Random Forest CV	57601.96199	240.00409	0.83176
5	Gradient Boosting Regressor CV	35155.15940	187.49709	0.89732
6	XGB Regressor CV	34495.60614	185.72993	0.89925

Based on R2 score Gradient Boosting(Train R2=0.948, Test R2=0.897) is the model that we have selected for deployment

Conclusion

Linear Regression with Lasso and Ridge Regularization:

1. Linear Regression gave underfitted model as the feature relation with the target variable was nonlinear one.
2. Also the lasso and ridge regularization was not helpful as the model did not overfitted, it underfitted due to non-linear relationship.

Decision Tree:

1. Decision Tree gave good improvement in R2 score as compared to linear regression.
2. Train R2=0.847 and Test R2=0.812 were obtained.

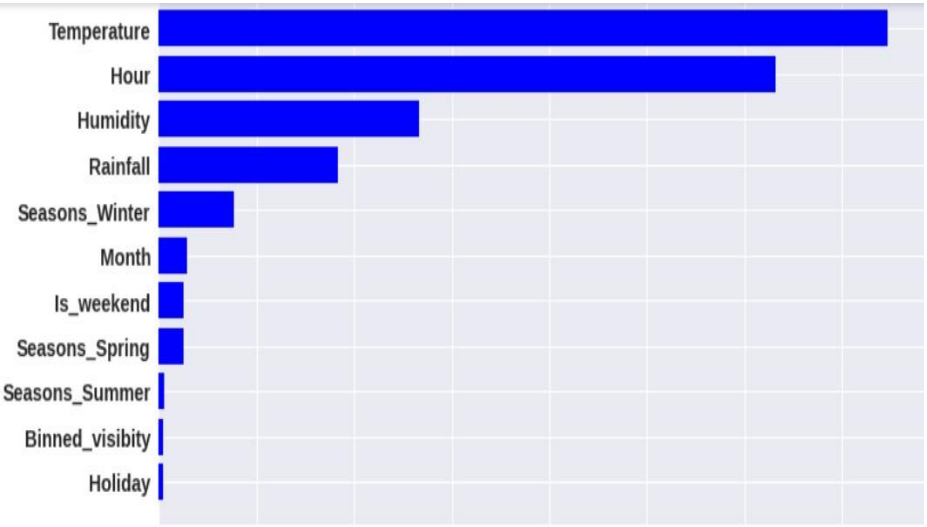
Random Forest:

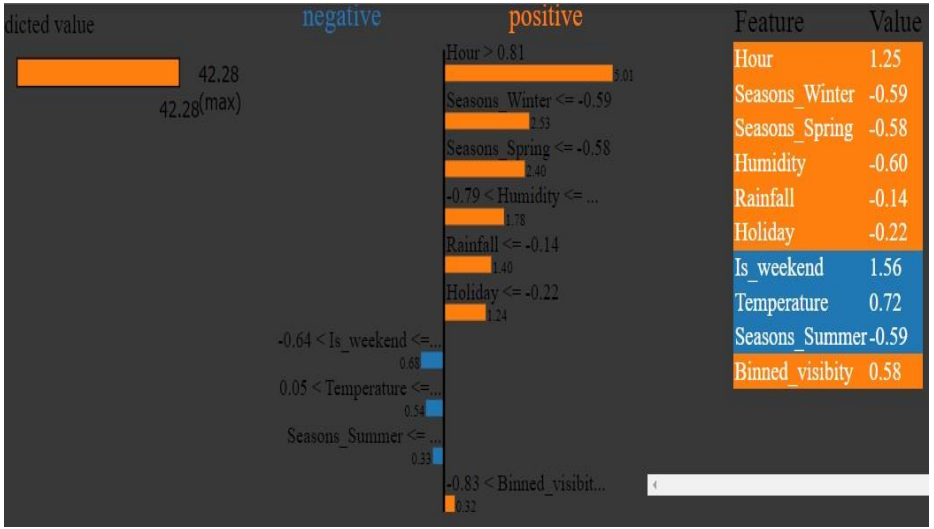
1. Gave little gain as compared to decision tree.
2. Train R2=0.856 and Test R2=0.831 were obtained.

Gradient Boosting and XGB Boosting

1. Both boosting techniques gave very good score, better than any other model.
2. The reason for choosing Gradient Boosting over XGB Boosting is because very small difference between train R2 score and test R2 score for Gradient Boosting which is 5.1% as compared to XGB Boosting which is 9.3%.
3. Gradient Boosting seems to have generalized well on the given data.

Feature Importance's





- Temperature, Hour, Humidity, Rainfall and Seasons Winter are 5 most important features.
- Using LIME we have explained one test instance . Here is_weekend, temperature, seasons_summer are acting to bring down target variable while all other features are acting to increase the target variable, with hour being the most important feature.

Thank You