Capstone Project – 2

Bike Sharing Demand Prediction

by:-

Akshit Singh

Agenda

- ➔ Look at the problem statements
- ➔ Study the dataset
- ➔ Looking for null/duplicate values
- ➔ E.D.A and visualization
- ➔ Look at the correlation among variables
- ➔ Regression models and evaluation metrics for the model
- ➔ Hyperparameter tuning
- ➔ Model comparison
- ➔ Conclusion

Approach

- **Understanding Business problem**

- **Data collection & Preprocessing**
  - Data cleaning
  - Missing Data Handling

- **Exploratory Data Analysis**
  - Understanding categorical and numerical features
  - EDA conclusion

- **Data manipulation**
  - Feature Engineering
  - Outlier Detection & Treatment
  - Feature scaling
  - Categorical Data encoding

- **Modelling**
  - Train Test split
  - Fitting models to a Data
  - Hyperparameter Tuning

- **Model Performance & Evaluation**
  - Visualizing Model Performance
  - Base models v/s Tuned models

- **Conclusion and Recommendations**

Problem Statement

At present rental bikes are presented in numerous metropolitan urban cities to improve portability solace. It is essential to make the rental bikes accessible and open to general society as it reduces the waiting time. Ultimately, furnishing the city with a stable stock of rental bikes becomes a main pressing issue. The essential part is predicting the bike count expected every hour for the steady inventory of rental bikes.

Data Summary

➔ There are around 9,000 observations with various types of field in our Dataset.

➔ List of columns:-

Date (dd/mm/yyyy)	Solar radiation (MJ/m2)
Rented Bike Count	Rainfall (mm)
Hour	Snowfall (cm)
Temperature (°C)	Seasons
Humidity (%)	Holiday
Windspeed (m/s)	Functioning Day
Visibility (10m)	
Dew point temperature (°C)	

Null and duplicate values

- ➔ There are no null/missing values present in our dataset.

- ➔ No duplicates are found in the dataset.

```
# checking if any null values are present in our dataset
count_of_null_values = seoul_df.isnull().sum()
count_of_null_values
```

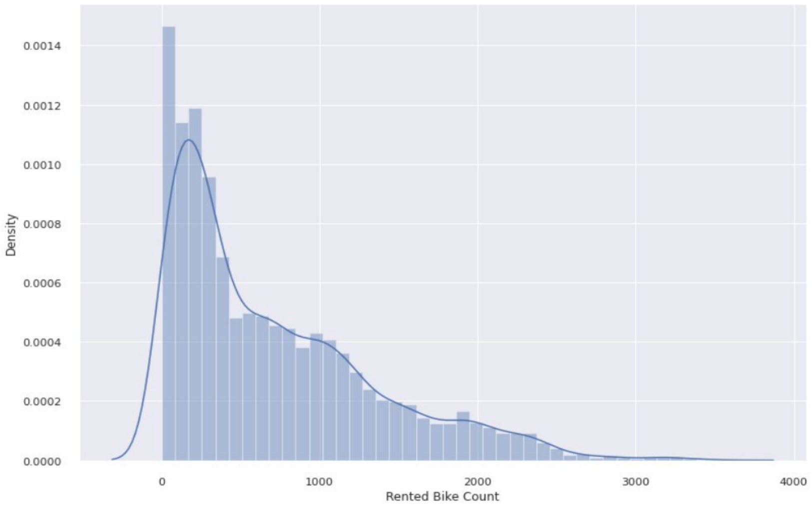
```
Date                         0
Rented Bike Count            0
Hour                         0
Temperature(°C)              0
Humidity(%)                  0
Wind speed (m/s)             0
Visibility (10m)             0
Dew point temperature(°C)    0
Solar Radiation (MJ/m2)      0
Rainfall(mm)                 0
Snowfall (cm)                0
Seasons                      0
Holiday                      0
Functioning Day              0
dtype: int64
```

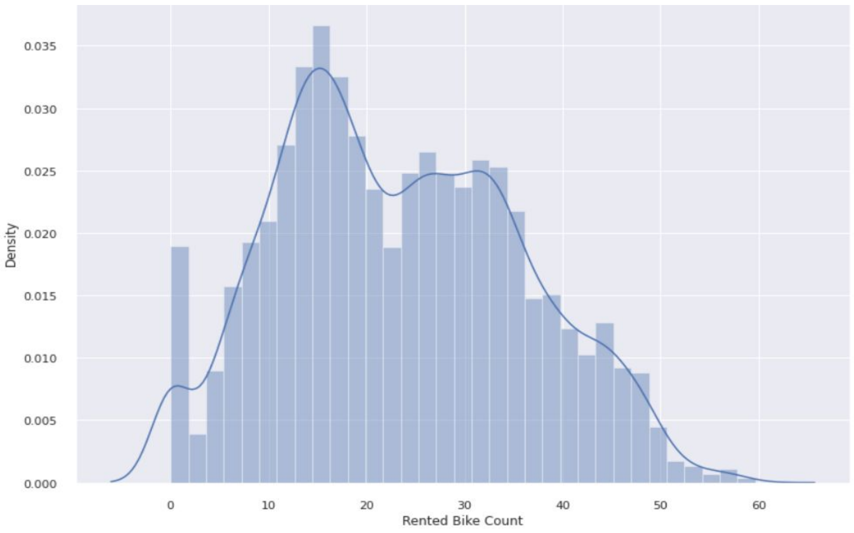
```
[ ] # checking duplicates in our dataset
value=len(seoul_df[seoul_df.duplicated()])
print("Total no. of duplicates = ",value)
```

```
Total no. of duplicates =  0
```

Exploratory Data Analysis

Dependent variable (Rented Bike Count)

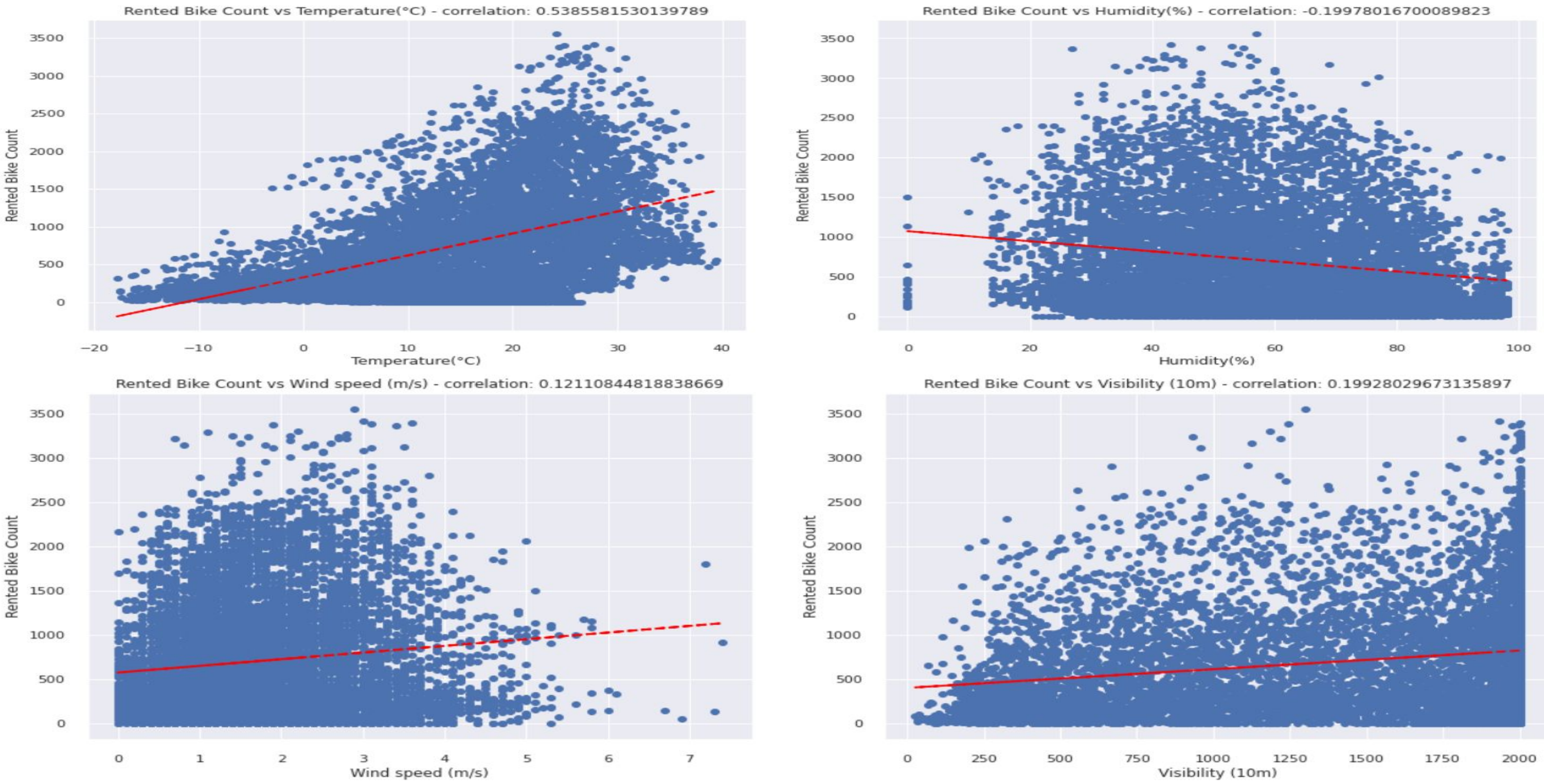


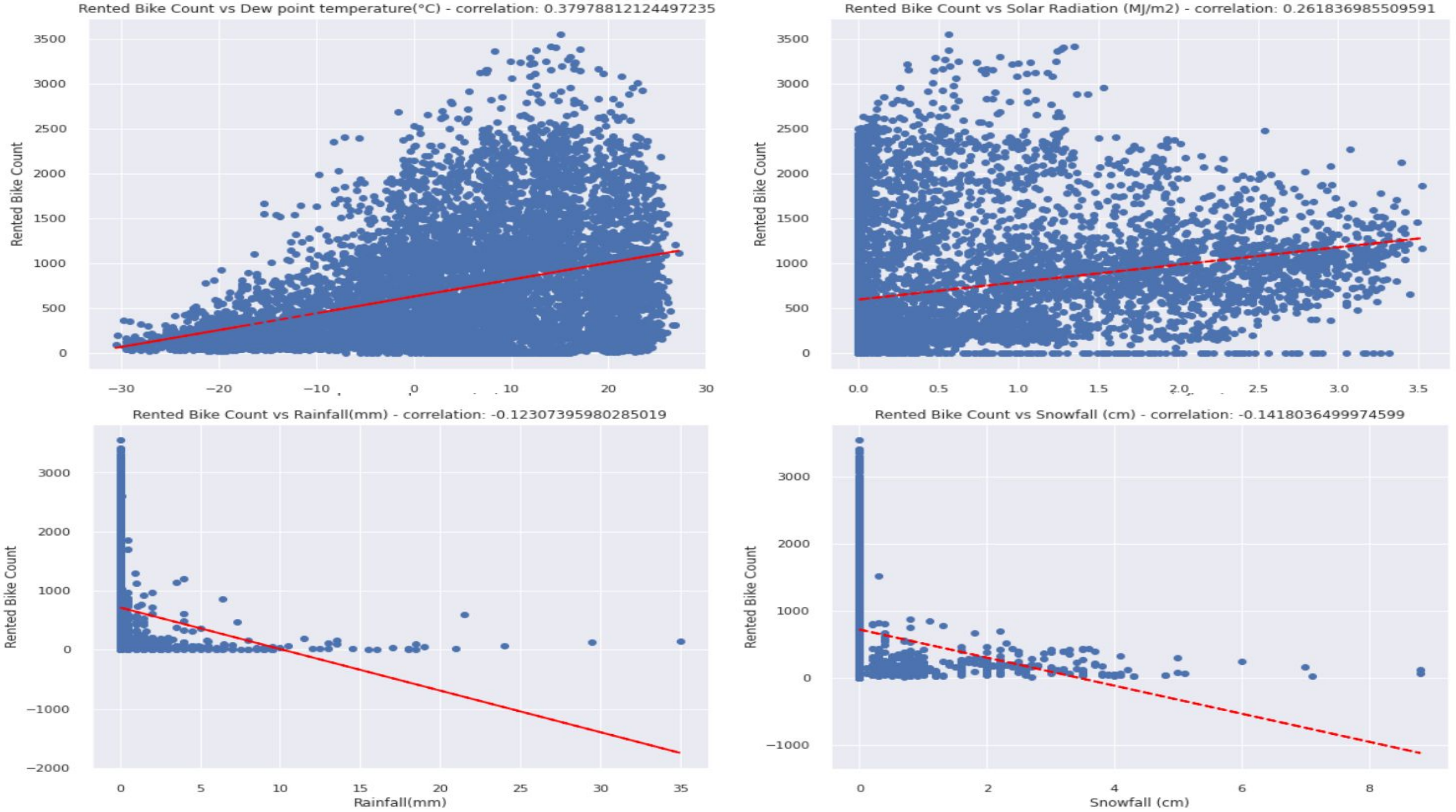


➔ The graph suggests that rented bike count has positive skewness.

➔ It is removed using square-root transformation

Relationship between dependent and independent variables




Rented Bike Count vs Dew point temperature(°C) - correlation: 0.37978812124497235
Rented Bike Count vs Solar Radiation (MJ/m2) - correlation: 0.261836985509591
Rented Bike Count vs Rainfall(mm) - correlation: -0.12307395980285019
Rented Bike Count vs Snowfall (cm) - correlation: -0.1418036499974599
Rented Bike Count
Rainfall(mm)
Snowfall (cm)

Heatmap

- There is a high correlation between Temperature and Dew point temperature with a value of 0.92.
- Temperature has the highest correlation value with Rented Bike Count followed by Dew point temperature.
- Humidity has a moderate correlation with Visibility and Dew point temperature.

Checking multicollinearity

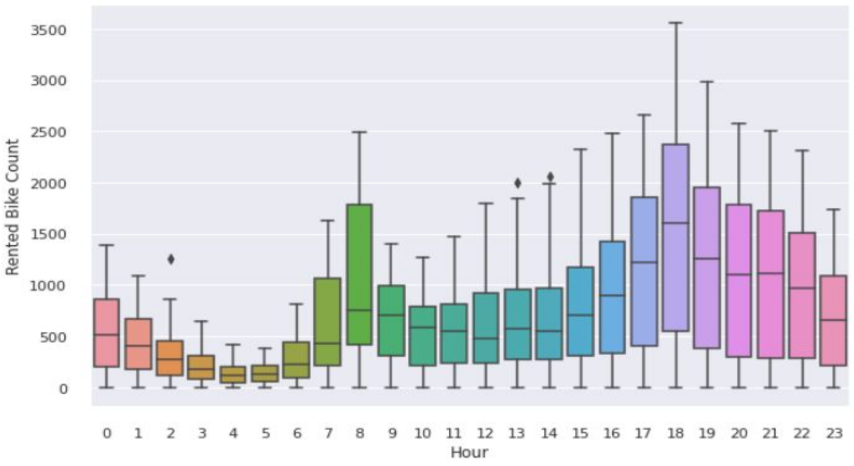
	independent_variables	VIF
0	Temperature	28.012254
1	Humidity	5.294308
2	Wind speed	4.640317
3	Visibility	8.924843
4	Dew point temperature	15.725322
5	Solar Radiation	2.363219
6	Rainfall	1.120123
7	Snowfall	1.124131

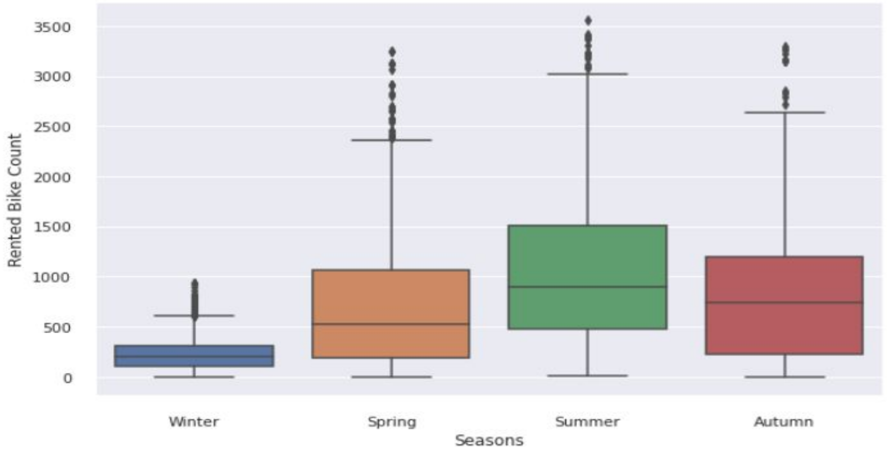
➔

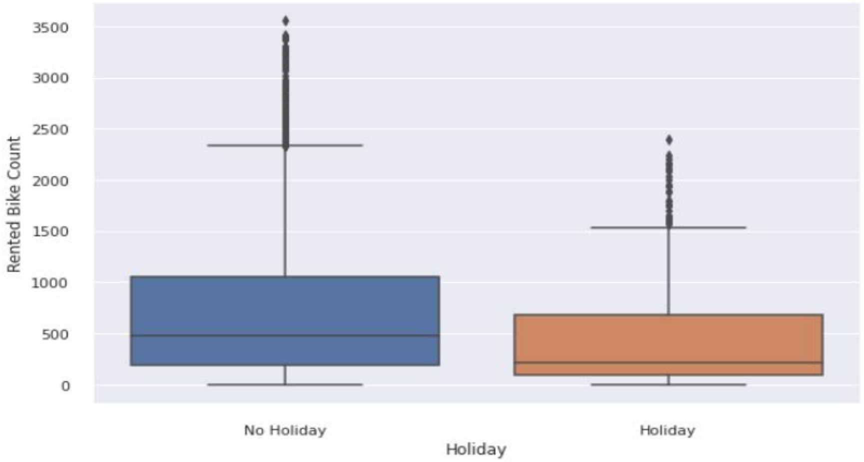
	independent_variables	VIF
0	Temperature	2.899187
1	Humidity	4.966258
2	Wind speed	4.155954
3	Visibility	4.388559
4	Solar Radiation	1.948663
5	Rainfall	1.118092
6	Snowfall	1.124130

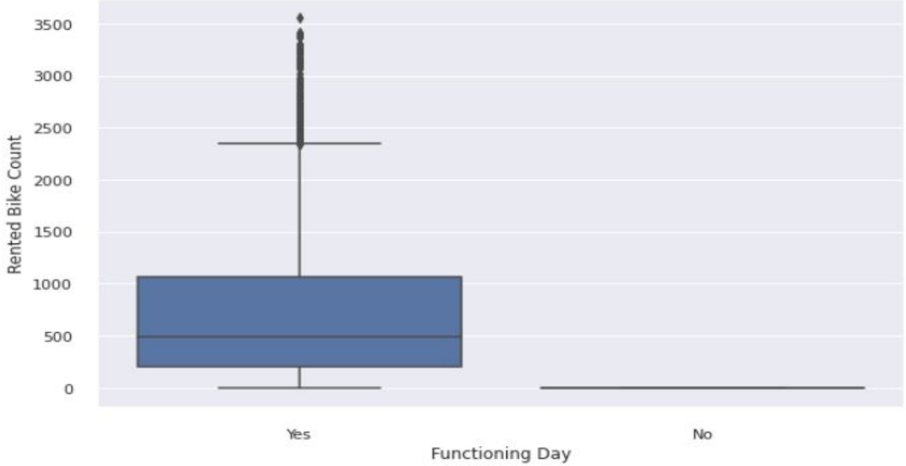
- ➔ Temperature and Dew point temperature have high VIF values.
- ➔ The VIF values for all the features are less than 5 after removing Dew point temperature.

Understanding Categorical features

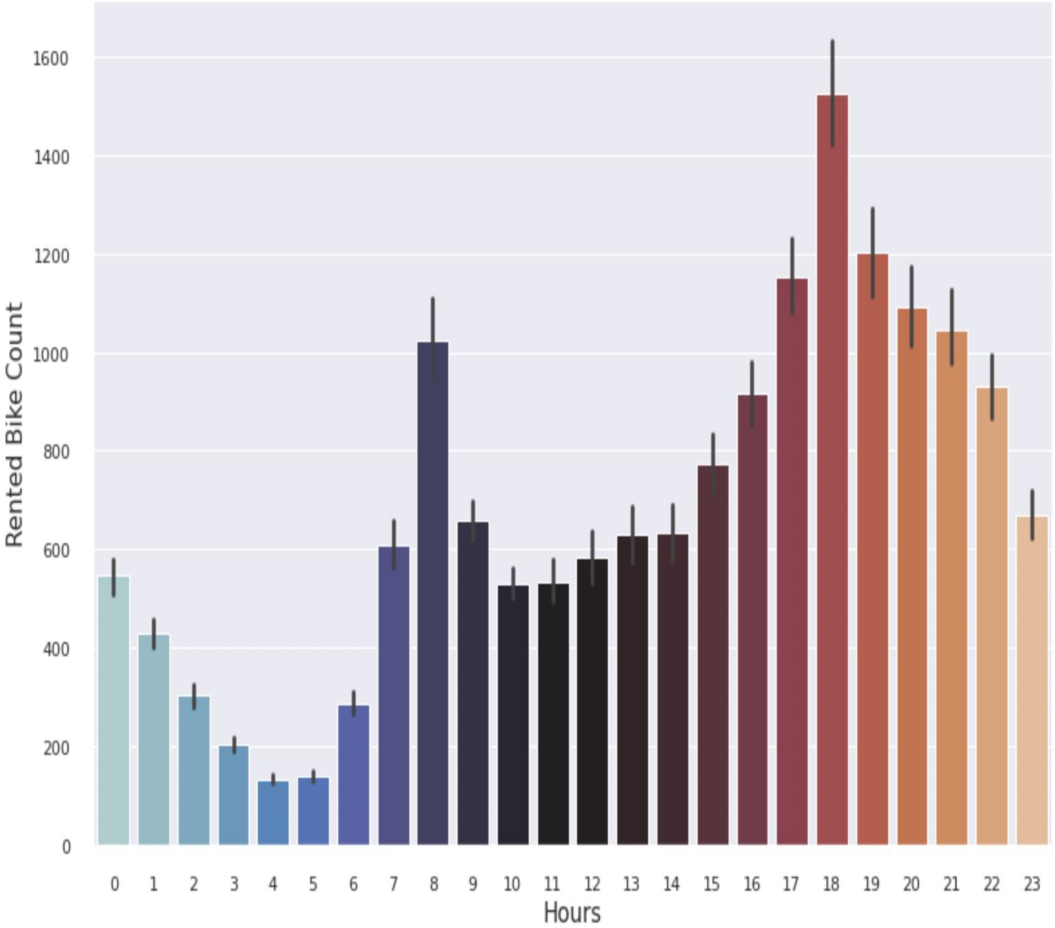








Number of rented bikes for each hour

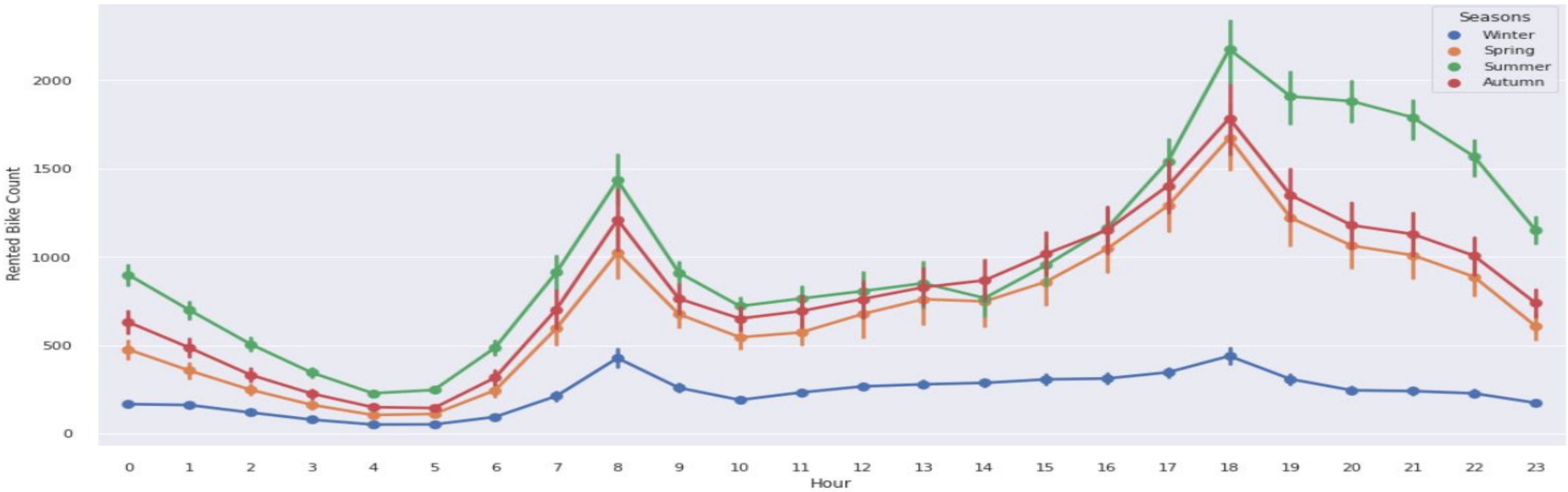


➔ The bikes are in high demand in the evening with a peak time at 6 pm.

➔ The demand is high at 8 am in the morning as well.

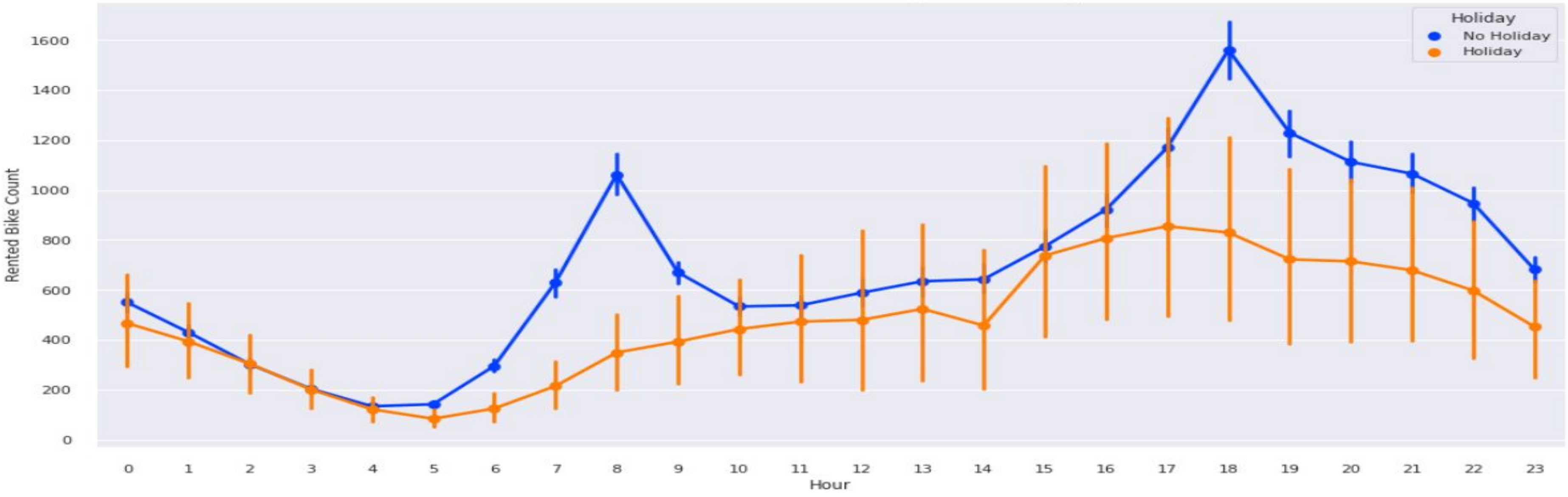
➔ People generally use rented bikes in the course of their operating hours i.e. from 7 am to 9 am and 5 pm to 7 pm.

Demand of bikes according to seasons



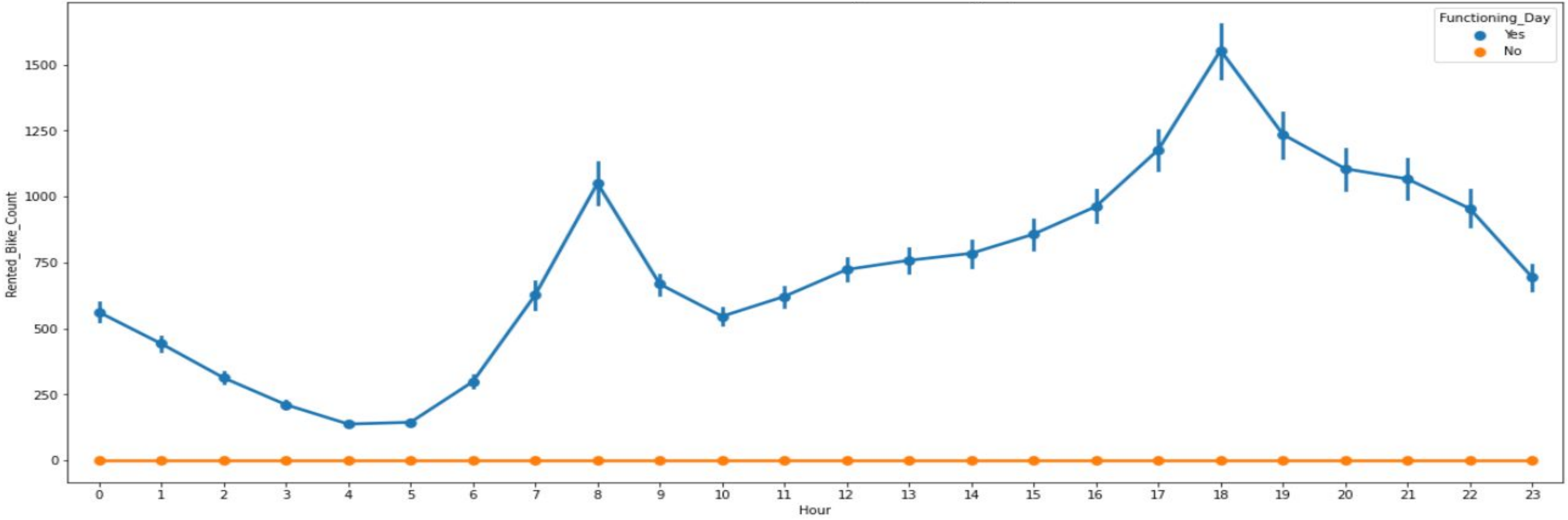
- ➔ The demand for rented bikes is high in summer with the peak time of 7-9 AM & 5-7 PM followed by Autumn.
- ➔ The demand is least in the winter season (due to snowfall which is negatively correlated with our target variable).

Rented bike count on holidays for every hour



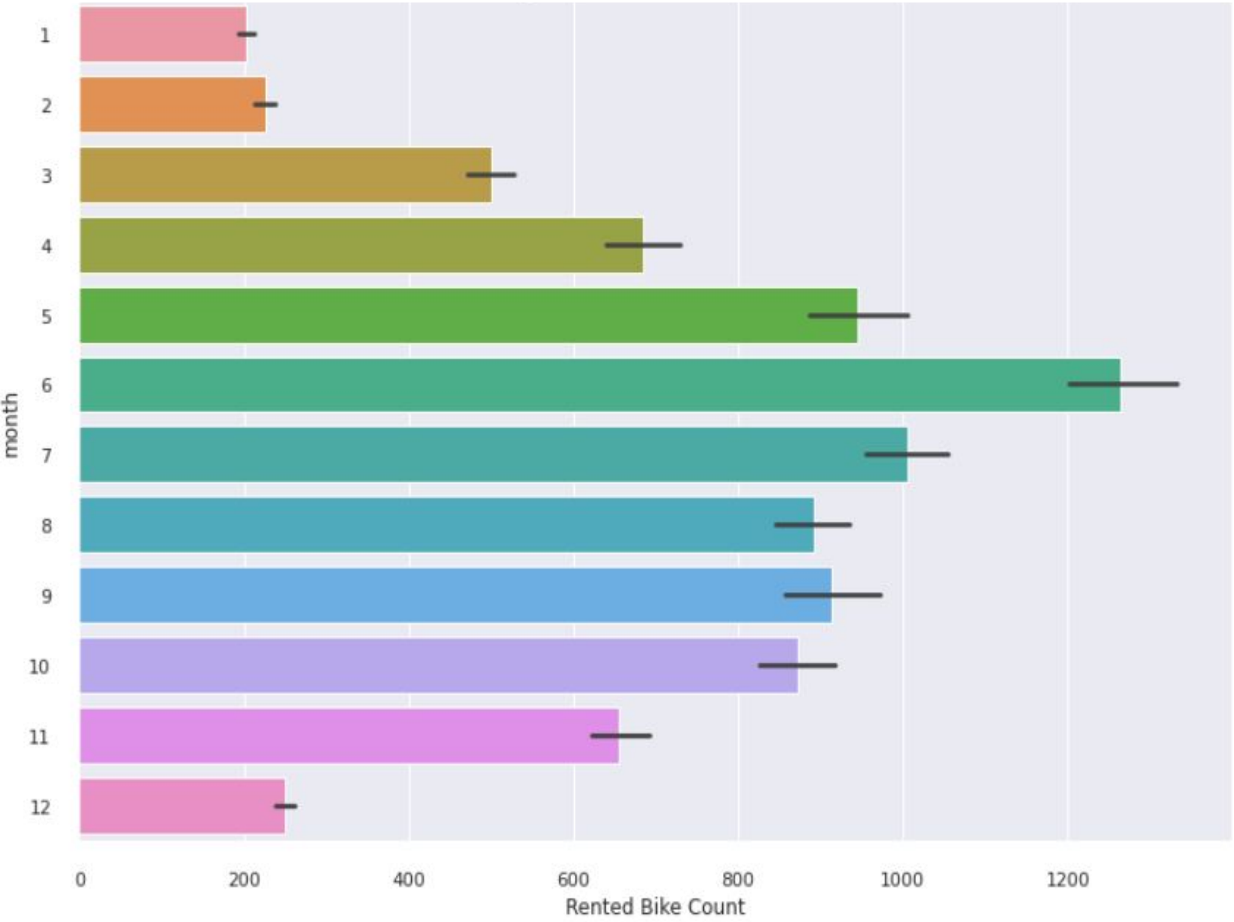
- ➔ When there are no holidays, two spikes can be seen one at 8 am and one at 6 pm. So in the morning and evening, the demand for rented bikes is high.
- ➔ When there are holidays the demand is low during the entire day.

Rented bike count on functioning days for every hour



➔ From the above plot we can see that rented bike count is high on functioning days whereas it is zero on non-functioning data.

Monthly count of rented bikes



→ During the month of June, the demand for bikes is high followed by July and May.

→ Demand is lowest in December, January and February.

ML Regression Algorithms

- Linear Regression
- Polynomial Regression
- Lasso Regression
- Ridge Regression
- Elastic-net Regression
- Decision Tree

(we have also perform Hyperparameter Tuning to improve our model performance)

Linear regression

```
# evaluating our model on training data
reg_eval_metrics(np.square(y_train), np.square(y_train_pred))
```

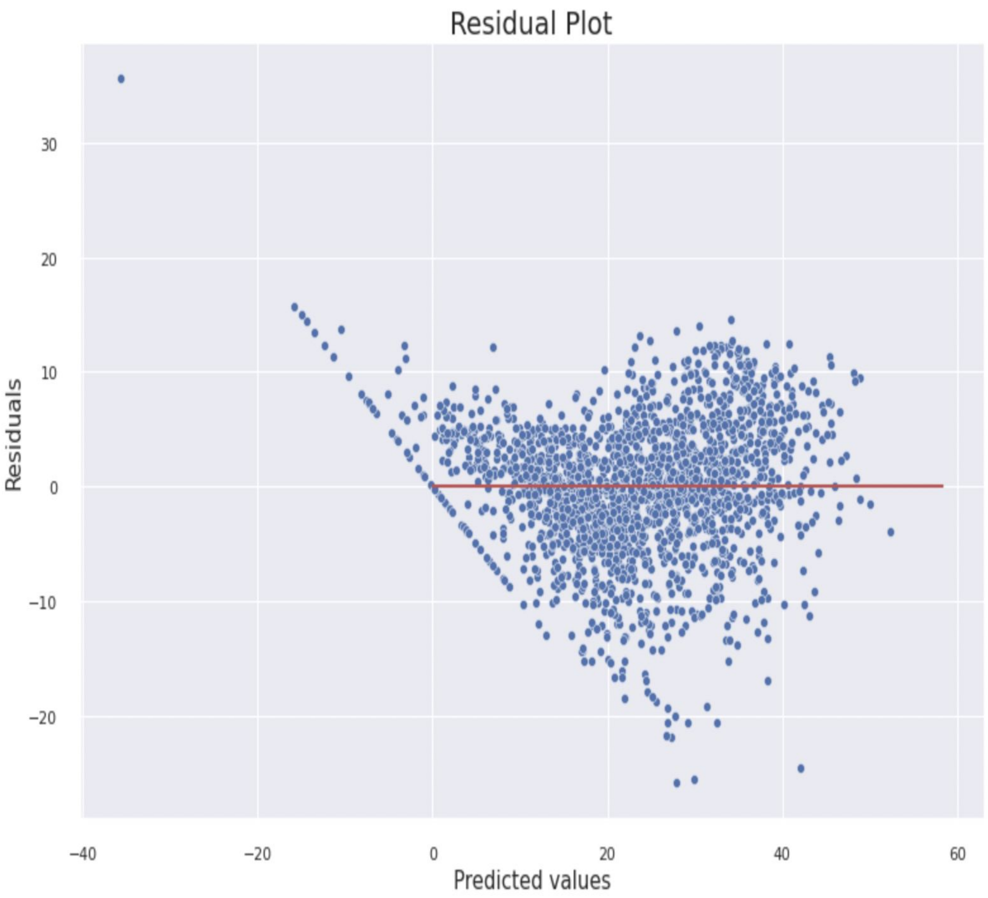
```
MSE: 94976.90138308439
RMSE: 308.18322696584966
MAE: 209.86672481683206
R2: 0.783670414018 7147
Adjusted R2: 0.7818706670471567
```

```
# evaluating our model on test data
reg_eval_metrics(np.square(y_test), np.square(y_test_pred))
```

```
MSE: 92653.40379245035
RMSE: 304.3902163218298
MAE: 208.32012921035465
R2: 0.7682432377375812
Adjusted R2: 0.7623610864111239
```

```
# evaluating our model score by cross-validation technique
accuracies = cross_val_score(reg, X_train, y_train, cv = 10)
print('Cross val score for Training Data:', accuracies.mean()*100)
```

```
Cross val score for Training Data: 79.26409655795959
```



Polynomial Regression

```
# evaluating our model on training data
reg_eval_metrics(np.square(y_train), np.square(y_train_pred))
```

```
MSE:  28389.664723228707
RMSE:  168.49232838093462
MAE:  101.63207436756537
R2:  0.9353366521092108
Adjusted R2:  0.9347986874844955
```

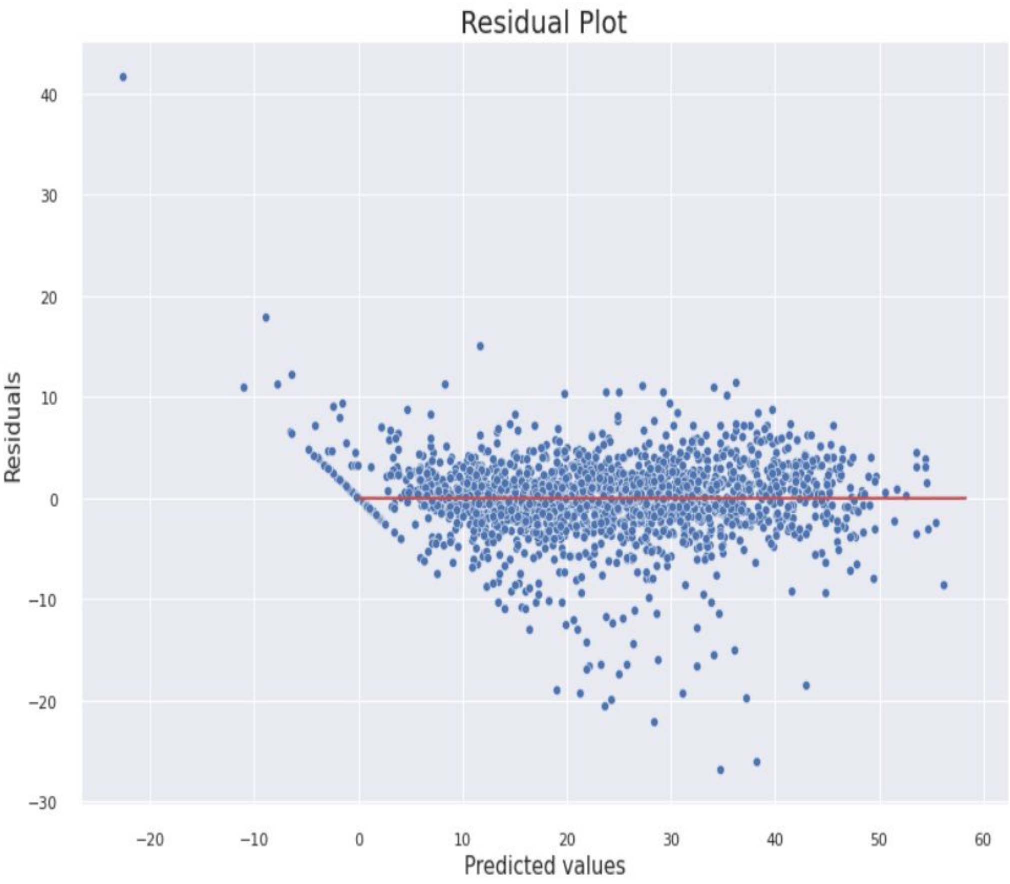
```
# evaluating our model on test data
reg_eval_metrics(np.square(y_test), np.square(y_test_pred))
```

```
MSE:  34720.99476580314
RMSE:  186.33570448468308
MAE:  117.37459935096749
R2:  0.9131513252607715
Adjusted R2:  0.9109470441760195
```

Checking the homoscedasticity

```
[ ]  mean_of_residuals = np.mean(y_test_pred - y_test)
     mean_of_residuals
```

```
0.02132886377835526
```



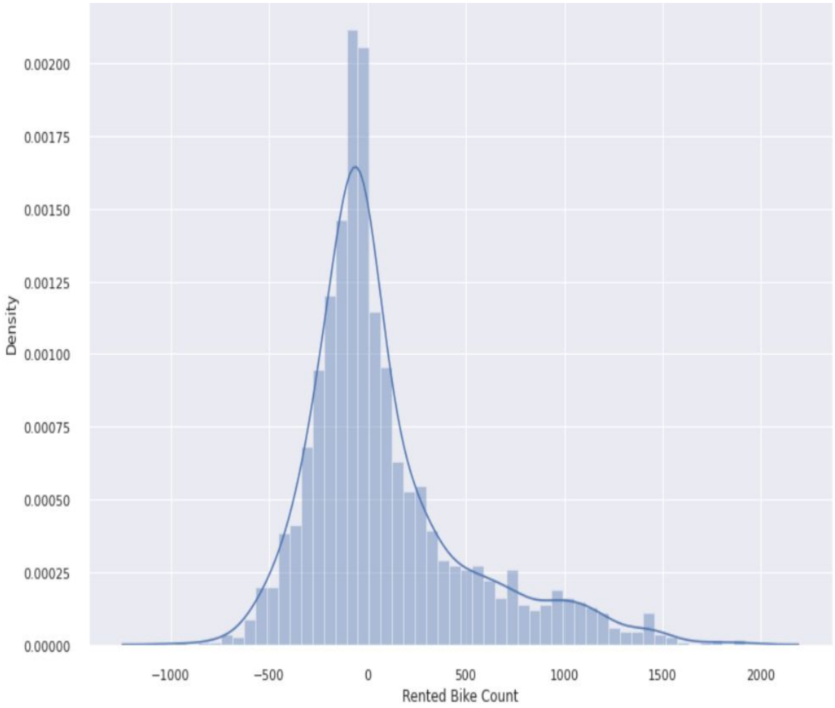
Lasso regression

```
# evaluating our model on training data
reg_eval_metrics(np.square(y_train), np.square(y_pred_train_lasso))
```

```
MSE: 197460.81588895767
RMSE: 444.36563310966983
MAE: 297.39000379262995
R2: 0.5502420491010782
Adjusted R2: 0.5465003024213867
```

```
# evaluating our model on test data
reg_eval_metrics(np.square(y_test), np.square(y_pred_test_lasso))
```

```
MSE: 179092.35603361562
RMSE: 423.1930481867768
MAE: 284.6044289170446
R2: 0.552030871167184
Adjusted R2: 0.5406610963237115
```



- ➔ R2 score is quite low.
- ➔ The distribution is positively skewed.

Hyperparameter Tuning on Lasso Regression

```
# hyperparameter tuning
lasso = Lasso()
parameters = {'max_iter': [100, 500, 1000], 'alpha': [1e-10,1e-8,1e-5,1e-4,1e-3,1e-2,1e-1,1,5,10,20,30,40,50,60,100]}
lasso_regressor = GridSearchCV(lasso, parameters, scoring='neg_mean_squared_error', cv=5)
lasso_regressor.fit(X_train, y_train)
```

```
GridSearchCV(cv=5, estimator=Lasso(),
             param_grid={'alpha': [1e-10, 1e-08, 1e-05, 0.0001, 0.001, 0.01,
                                   0.1, 1, 5, 10, 20, 30, 40, 50, 60, 100],
                         'max_iter': [100, 500, 1000]},
             scoring='neg_mean_squared_error')
```

```
print("The best fit alpha value is found out to be :" ,lasso_regressor.best_params_)
print("\nUsing ",lasso_regressor.best_params_, " the negative mean squared error is: ", lasso_regressor.best_score_)
```

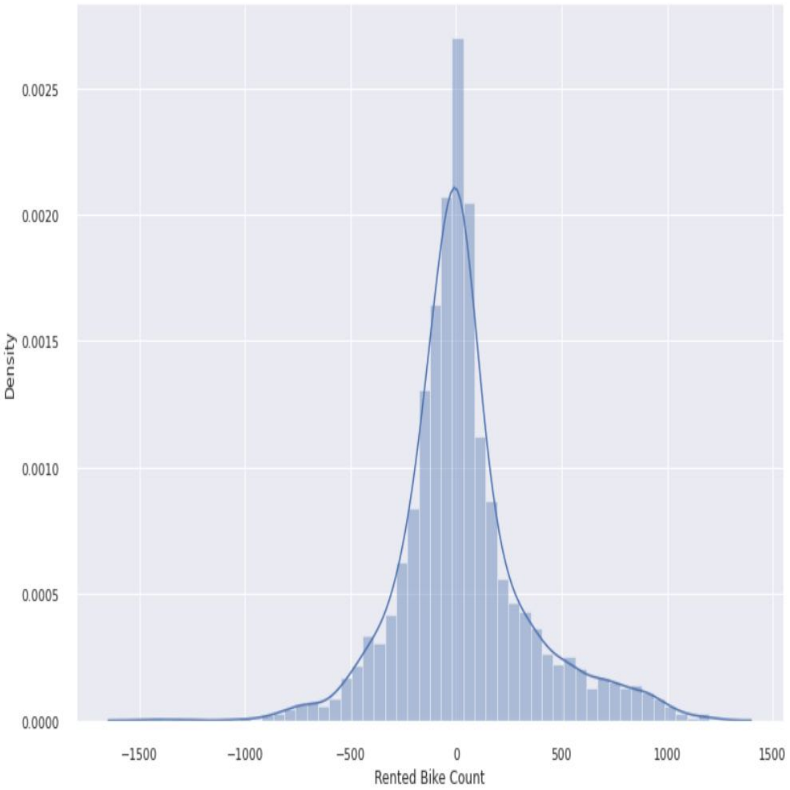
```
The best fit alpha value is found out to be : {'alpha': 0.001, 'max_iter': 500}

Using  {'alpha': 0.001, 'max_iter': 500}  the negative mean squared error is:  -33.2559113888446
```

```
# predicting on test set
y_pred_lasso = lasso_regressor.predict(X_test)
```

```
# evaluating our model on test data
reg_eval_metrics(np.square(y_test), np.square(y_pred_lasso))
```

```
MSE: 92892.8673380389
RMSE: 304.78331210556604
MAE: 208.257600566763
R2: 0.7676442603257011
Adjusted R2: 0.7617469065268611
```



- ➔ RMSE and MAE are reduced significantly for lasso regression.
- ➔ R2-score has been increased by 20%.

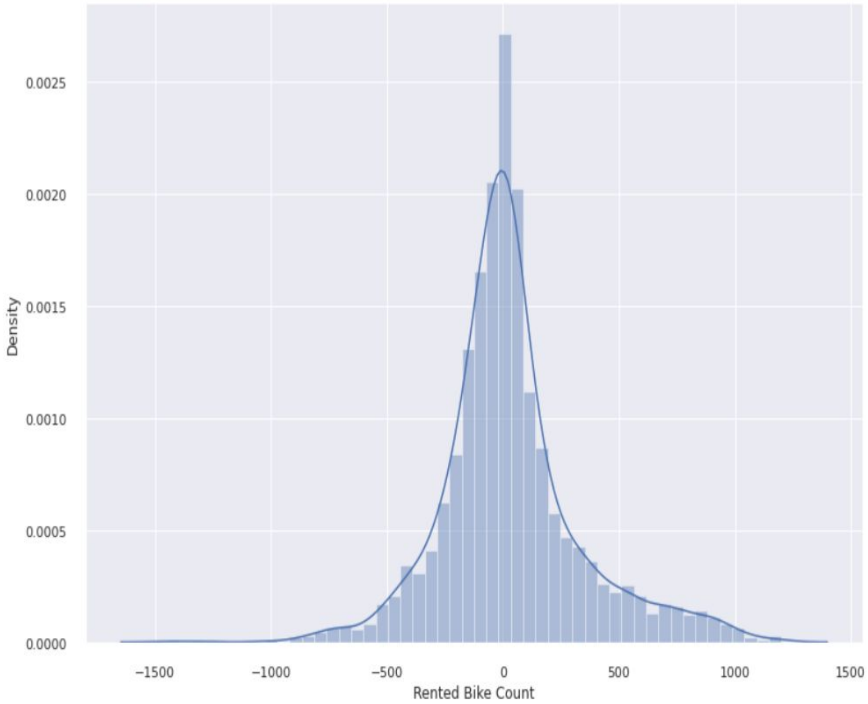
Ridge Regression

```
# evaluating our model on training data
reg_eval_metrics(np.square(y_train), np.square(y_pred_train_ridge))
```

```
MSE: 95205.44082441159
RMSE: 308.5537891914659
MAE: 209.84515140275636
R2: 0.7831498680543513
Adjusted R2: 0.7813457904175323
```

```
# evaluating our model on test data
reg_eval_metrics(np.square(y_test), np.square(y_pred_test_ridge))
```

```
MSE: 92875.91940735908
RMSE: 304.75550759151025
MAE: 208.25416138904723
R2: 0.7676866526974935
Adjusted R2: 0.761790374847176
```



➔ We can see that residuals are normally distributed when we use ridge regression.

Hyperparameter tuning on Ridge Regression

AI

```
# hyperparameter tuning
ridge = Ridge()
parameters = {'max_iter': [100, 500, 1000, 5000, 10000], 'alpha': [1e-10,1e-8,1e-4,1e-3,1e-2,0.1,1,5,10,20,30,40,50,60,100]}
ridge_regressor = GridSearchCV(ridge, parameters, scoring='neg_mean_squared_error', cv=5)
ridge_regressor.fit(X_train,y_train)
```

```
GridSearchCV(cv=5, estimator=Ridge(),
             param_grid={'alpha': [1e-10, 1e-08, 0.0001, 0.001, 0.01, 0.1, 1, 5,
                                   10, 20, 30, 40, 50, 60, 100],
                         'max_iter': [100, 500, 1000, 5000, 10000]},
             scoring='neg_mean_squared_error')
```

```
# finding the best value for alpha
print("The best fit alpha value is found out to be :" ,ridge_regressor.best_params_)
```

```
The best fit alpha value is found out to be : {'alpha': 5, 'max_iter': 100}
```

```
# predicting on test set
y_pred_ridge = ridge_regressor.predict(X_test)
```

```
# evaluating our model on test data
reg_eval_metrics(np.square(y_test), np.square(y_pred_ridge))
```

```
MSE: 92898.65981639555
RMSE: 304.7928145747461
MAE: 208.28786988786354
R2: 0.7676297714243282
Adjusted R2: 0.7617320498868746
```

→ After hyperparameter tuning there is no significant change in R2-score for ridge regression.

→ The RMSE and MAE are almost same before and after tuning.

→ The distribution of residuals is also normal distribution.

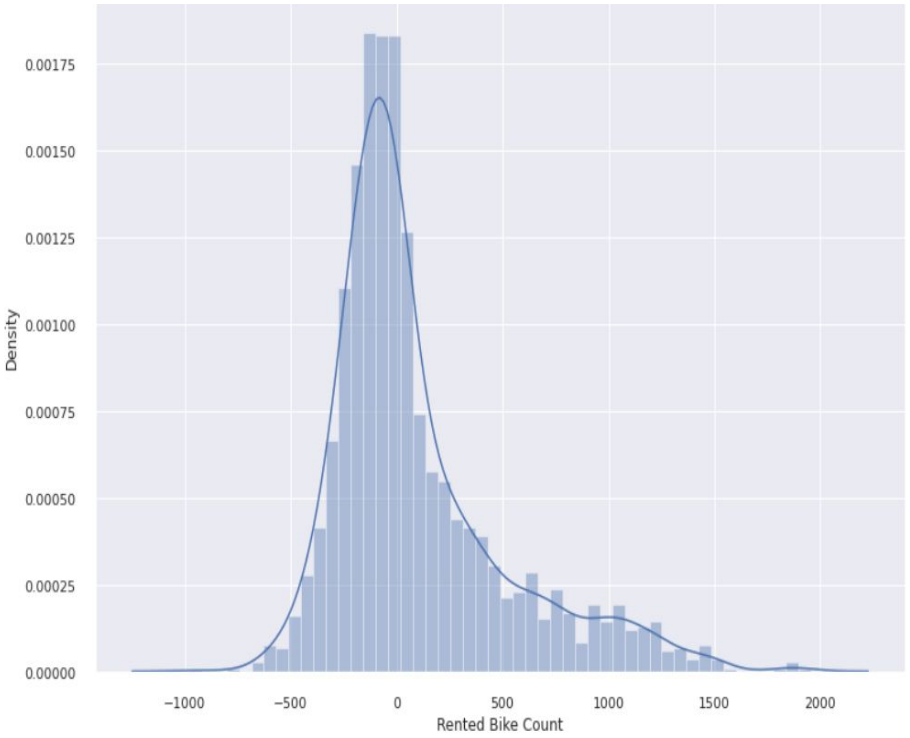
Elastic net

```
# evaluating our model on training data
reg_eval_metrics(np.square(y_train), np.square(y_pred_train_en))
```

```
MSE: 196998.62125117582
RMSE: 443.8452672398071З
MAE: 296.79748767686414
R2: 0.551294793222839l
Adjusted R2: 0.5475618048137114
```

```
# evaluating our model on test data
reg_eval_metrics(np.square(y_test), np.square(y_pred_test_en))
```

```
MSE: 179652.49525377314
RMSE: 423.85433258818193
MAE: 285.0853513579629
R2: 0.5506297779880202
Adjusted R2: 0.5392244424039598
```



- ➔ R2 score is low.
- ➔ There is positive skewness in the distribution of residuals for elastic net.

Hyperparameter Tuning on Elastic net

```
# hyperparameter tuning
elastic = ElasticNet()
parameters = {'alpha': [1e-10,1e-8,1e-5,1e-4,1e-3,1e-2,1e-1,1,5,10,20,30,40,50,60,100],'l1_ratio':[0.3,0.4,0.5,0.6,0.7
elastic_regressor = GridSearchCV(elastic, parameters, scoring='neg_mean_squared_error',cv=5)
elastic_regressor.fit(X_train, y_train)
```

```
GridSearchCV(cv=5, estimator=ElasticNet(),
             param_grid={'alpha': [1e-10, 1e-08, 1e-05, 0.0001, 0.001, 0.01,
                                   0.1, 1, 5, 10, 20, 30, 40, 50, 60, 100],
                         'l1_ratio': [0.3, 0.4, 0.5, 0.6, 0.7, 0.8]},
             scoring='neg_mean_squared_error')
```

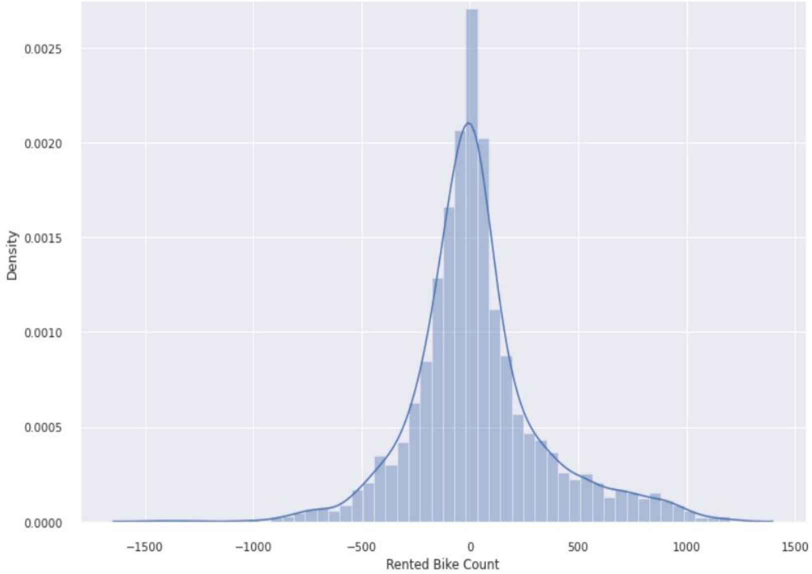
```
# finding the best value for alpha
print("The best fit alpha value is found out to be :" ,elastic_regressor.best_params_)
```

```
The best fit alpha value is found out to be : {'alpha': 0.001, 'l1_ratio': 0.4}
```

```
# predicting on test set
y_pred_elastic = elastic_regressor.predict(X_test)
```

```
# evaluating our model on test data
reg_eval_metrics(np.square(y_test), np.square(y_pred_elastic))
```

```
MSE: 92898.92462153142
RMSE: 304.7932489763043
MAE: 208.2799568695203
R2: 0.7676291090592298
Adjusted R2: 0.7617313707104792
```



- ➔ R2 score for elastic net increased by 20%.
- ➔ For all the regularization techniques, R2 score comes out to be around 0.76. RMSE and MAE is also the same.
- ➔ Now the residuals are normally distributed.

Decision Tree Regressor

```
# evaluating our model on test dataset
reg_eval_metrics(y_test, y_pred)
```

```
MSE: 73648.6976744l861
RMSE: 271.3829354886l657
MAE: 158.4275113310243
R2: 0.8157802841641768
Adjusted R2: 0.811104656858699
```

```
# evaluating our model score by cross-validation technique
accuracies = cross_val_score(dt_reg, X_train, y_train, cv = 10)
print('Cross val score for Training Data:', accuracies.mean()*100)
```

```
Cross val score for Training Data: 81.9095818992463
```

- ➔ Unlike linear regression, decision trees have no prior assumptions, so we don't have to standardize our variables. Plus we don't have to take the square-root of our dependent variable to make it normally distributed.
- ➔ R2 score is more than other regression techniques by 5%.

Hyperparameter Tuning on Decision Tree

```
[ ] # hyperparameter tuning using GridSearchCV
    dtr = DecisionTreeRegressor()
    criterion = ['squared_error', 'absolute_error']
    max_depth = [25, 50, 100, 150, 200, 300]
    min_samples_split = np.arange(2, 20, 3)
    max_leaf_nodes = np.arange(200, 300, 20)
    hyperparameters = {'criterion': criterion, 'max_depth': max_depth, 'min_samples_split': min_samples_split, 'max_leaf_nodes': max_leaf_nodes}
    predictor = GridSearchCV(dtr, hyperparameters, cv = 5, verbose = 0)
    predictor.fit(X_train, y_train)
```

```
GridSearchCV(cv=5, estimator=DecisionTreeRegressor(),
             param_grid={'criterion': ['squared_error', 'absolute_error'],
                         'max_depth': [25, 50, 100, 150, 200, 300],
                         'max_leaf_nodes': array([200, 220, 240, 260, 280]),
                         'min_samples_split': array([ 2,  5,  8, 11, 14, 17])})
```

```
[ ] print("The best fit values are found out to be :", predictor.best_params_)
```

```
The best fit values are found out to be : {'criterion': 'absolute_error', 'max_depth': 300, 'max_leaf_nodes': 280, 'min_samples_split': 17}
```

```
[ ] # predicting on test set
    y_pred = predictor.predict(X_test)
```

```
[ ] # evaluating our model on test set after hyperparameter tuning
    reg_eval_metrics(y_test, y_pred)
```

```
MSE: 62872.99492825334
RMSE: 250.74488016359047
MAE: 151.93542800593767
R2: 0.8427338754768902
Adjusted R2: 0.8387423494737656
```

- ➔ We can see that the R2 score for the decision tree increased by 3% after hyperparameter tuning.
- ➔ RMSE and MAE values are also got reduced.

Model comparison

S.No	Algorithm	RMSE	MAE	R2	Adjusted R2
1.	Linear regression	304.39021	208.32013	0.76824	0.76236
2.	Polynomial regression (2nd degree)	186.33570	117.37459	0.91315	0.91094
3.	Lasso regression	304.78331	208.25760	0.76764	0.76174
4.	Ridge regression	304.79281	208.28786	0.76762	0.76173
5.	ElasticNet	304.79324	208.27995	0.76762	0.76173
6.	Decision tree regressor	250.74488	151.93542	0.84273	0.83874

Conclusion

- People prefer rented bikes when the temperature is high and humidity is low.
- When there are no holidays, the demand is high specifically around 8 am and 6 pm. When there are holidays the demand is quite low during the entire day.
- People like to take rented bikes when rainfall and snowfall are low.
- During the entire day, the demand is high in the morning and even higher in the evening which shows employees mostly prefer rented bikes before and after their working hours.
- The demand is high in summer especially in the month of June followed by July and May whereas the demand is lowest in the winter season i.e. during the month of December, January and February.

Contd.

- ➔ The R2 score for linear regression and all the regularization techniques like lasso, ridge and elastic net is almost the same, around 0.76.
- ➔ For decision tree, the R2 score comes out to be 0.84. Also, the RMSE and MAE are comparatively low from other regression algorithms.
- ➔ 2nd-degree polynomial regression has the highest value of R2 score of around 0.91 which means there is a non-linear relationship between the dependent and independent variables. It has also got the lowest RMSE and MAE values among all the models.