Capstone Project - 2

Seoul Bike Sharing Demand Prediction

Team Members

Shreedarsh M
Sachin S Panchal
Ranjith K
Rohan A G
Kshipra Parihar

Contents :

- **Phase 1**

EDA

- Data Exploration
- Data Analysis
- Checking Outliers
- Outlier treatment
- Correlation heatmap
- Removing multicollinearity

- **Phase 2**

Modelling

- Linear Regression
- Polynomial Regression
- Random Forest Regressor
- Hyperparameter Tuning

- **Phase 3**

Regression Evaluation Metrics

- Mean squared error
- Mean Absolute error
- Root mean squared error
- R2 score
- comparision

Introduction

- 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
- The dataset contains weather information (Temperature, Humidity, Windspeed, Visibility, Dewpoint, Solar radiation, Snowfall, Rainfall), the number of bikes rented per hour and date information.

Attribute Information

- Date : year-month-day
- Rented Bike count - Count of bikes rented at each hour
- Hour - Hour of he day
- Temperature-Temperature in Celsius
- Humidity - %
- Windspeed - 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 - NoFunc(Non Functional Hours), Fun(Functional hours)

Problem Statement

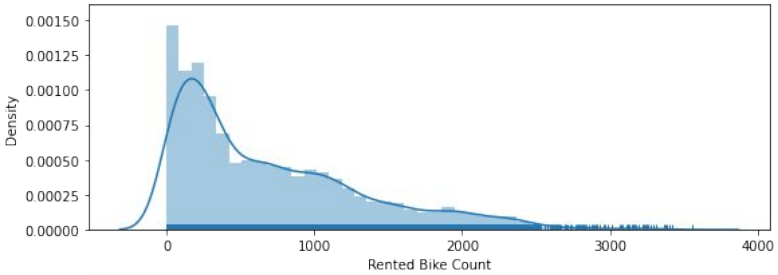
The project goal is to predict number of rental bikes required at a particular time of the day.

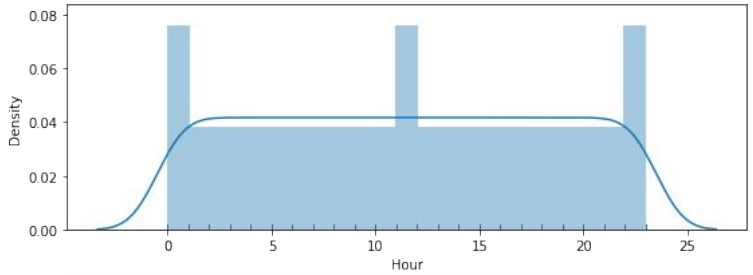
Exploratory Data Analysis

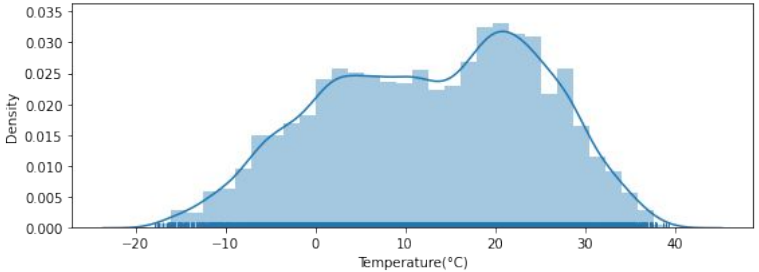
```
[ ] dataset.head()
```

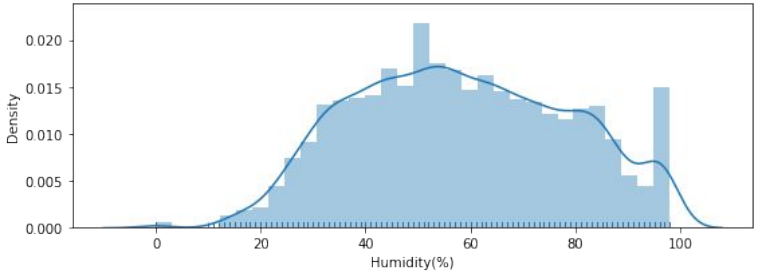
	Date	Rented Bike Count	Hour	Temperature(°C)	Humidity(%)	Wind speed (m/s)	Visibility (10m)	Dew point temperature(°C)	Solar Radiation (MJ/m2)	Rainfall(mm)	Snowfall (cm)	Seasons	Holiday	Functioning Day
0	01/12/2017	254	0	-5.2	37	2.2	2000	-17.6	0.0	0.0	0.0	Winter	No Holiday	Yes
1	01/12/2017	204	1	-5.5	38	0.8	2000	-17.6	0.0	0.0	0.0	Winter	No Holiday	Yes
2	01/12/2017	173	2	-6.0	39	1.0	2000	-17.7	0.0	0.0	0.0	Winter	No Holiday	Yes
3	01/12/2017	107	3	-6.2	40	0.9	2000	-17.6	0.0	0.0	0.0	Winter	No Holiday	Yes
4	01/12/2017	78	4	-6.0	36	2.3	2000	-18.6	0.0	0.0	0.0	Winter	No Holiday	Yes

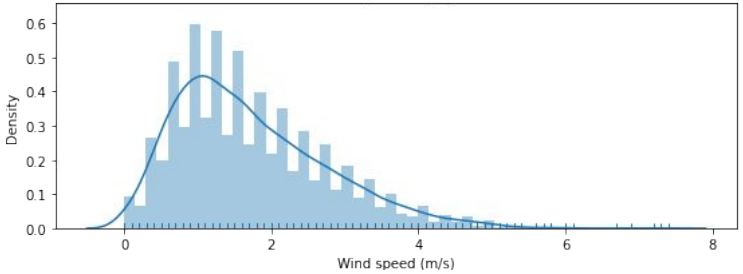
Exploratory Data Analysis

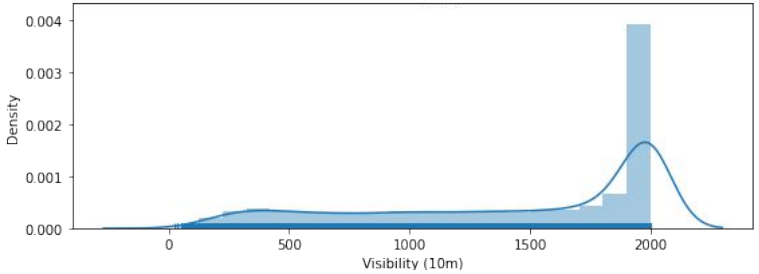




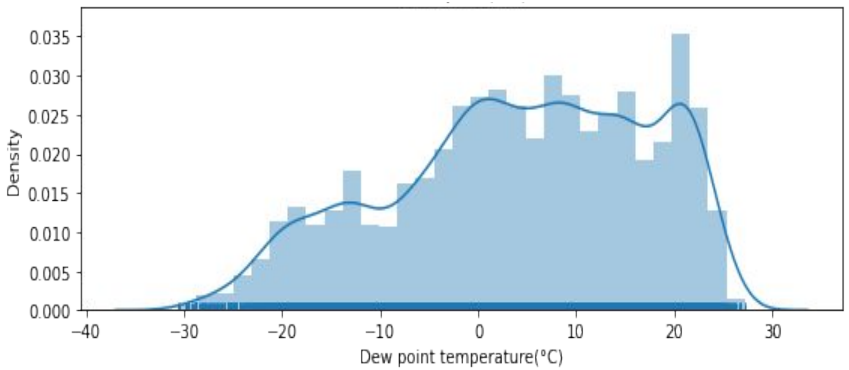


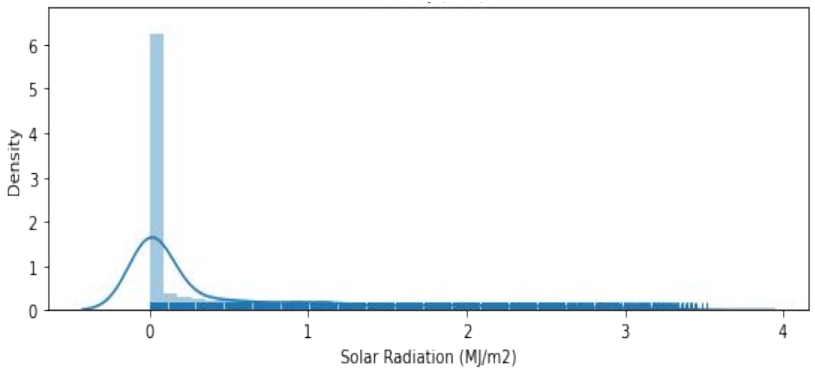


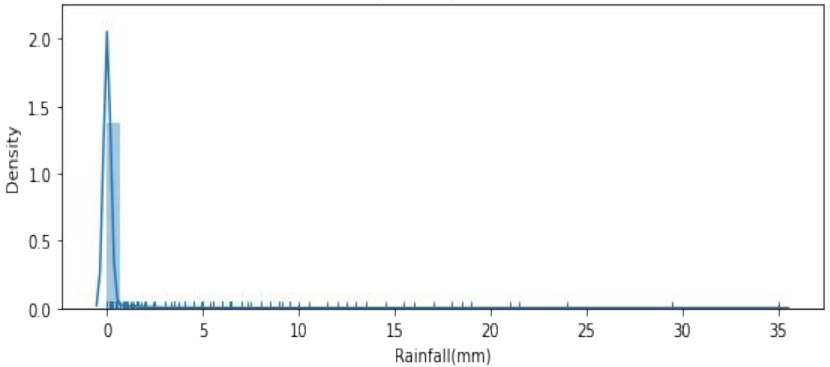


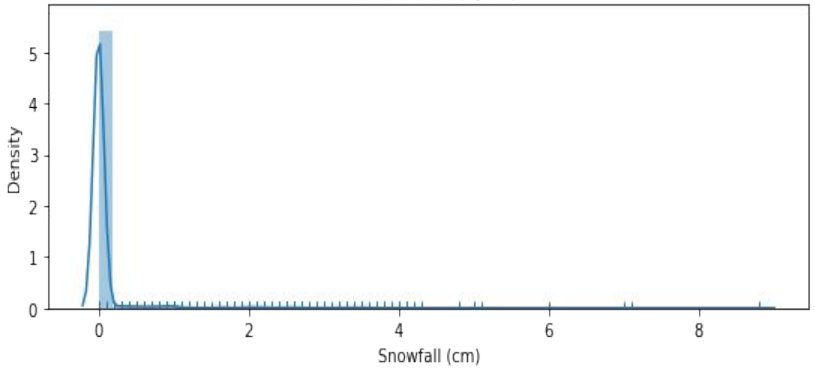


Exploratory Data Analysis

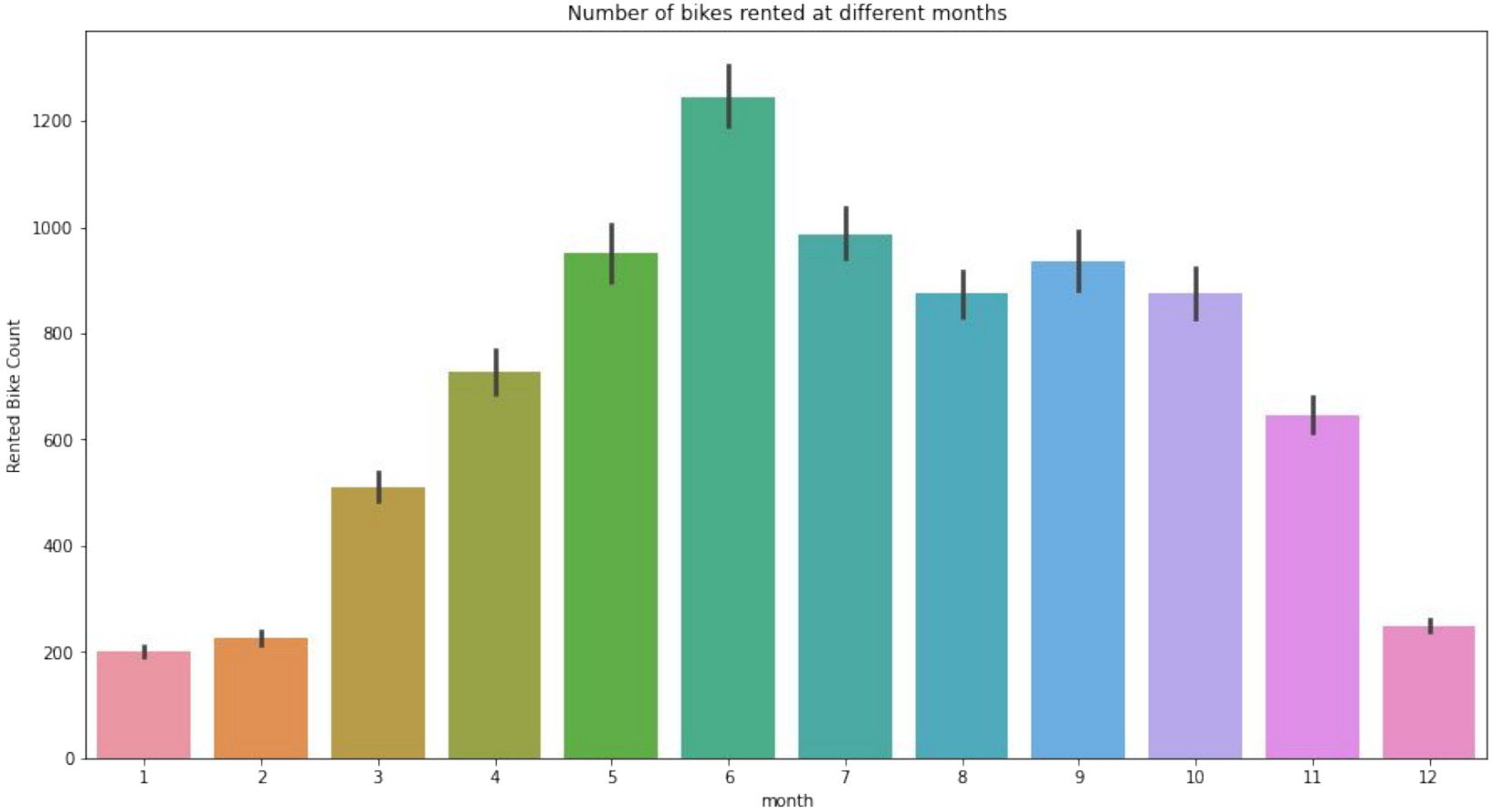






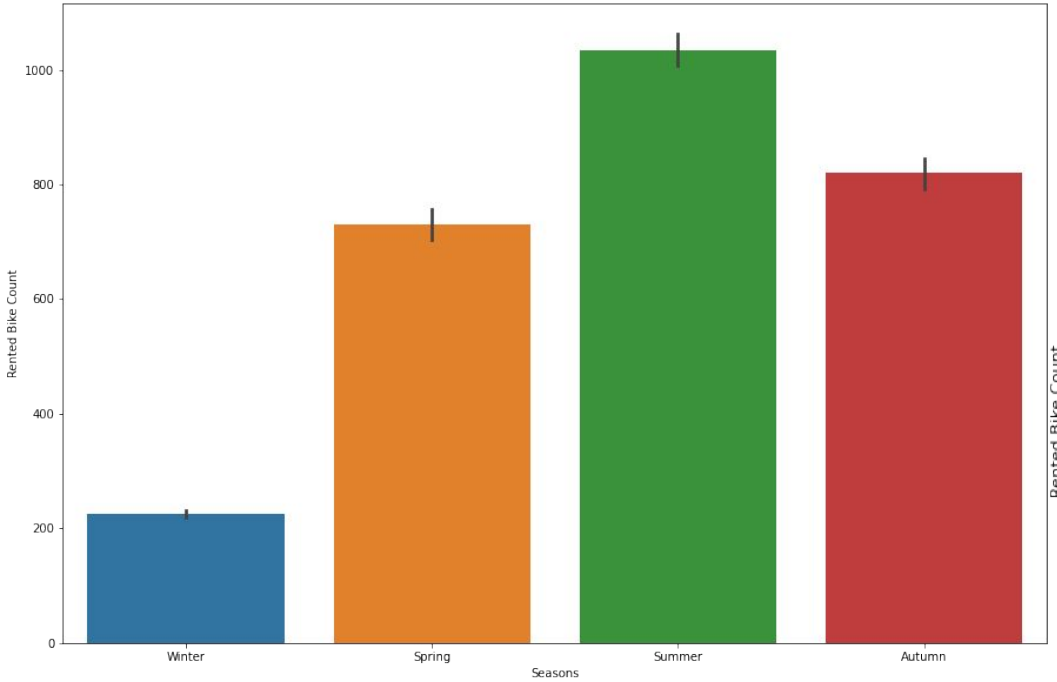


Exploratory Data Analysis

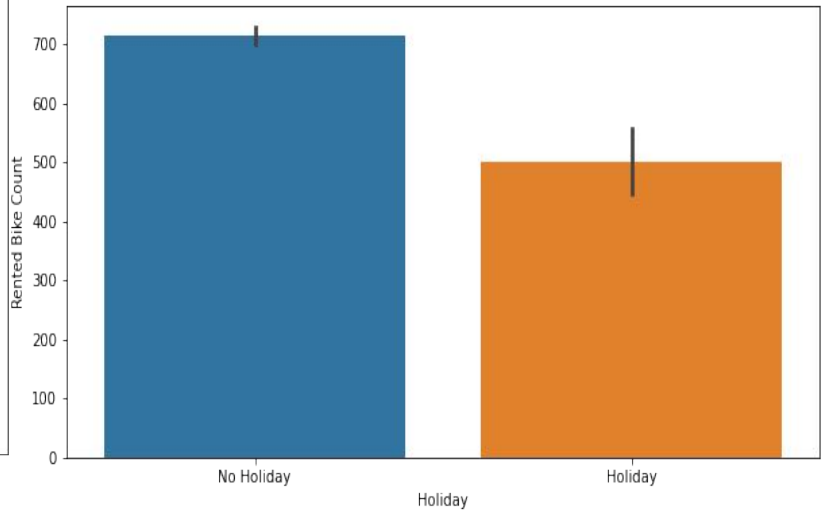


Exploratory Data Analysis

Bike counts vs Seasons

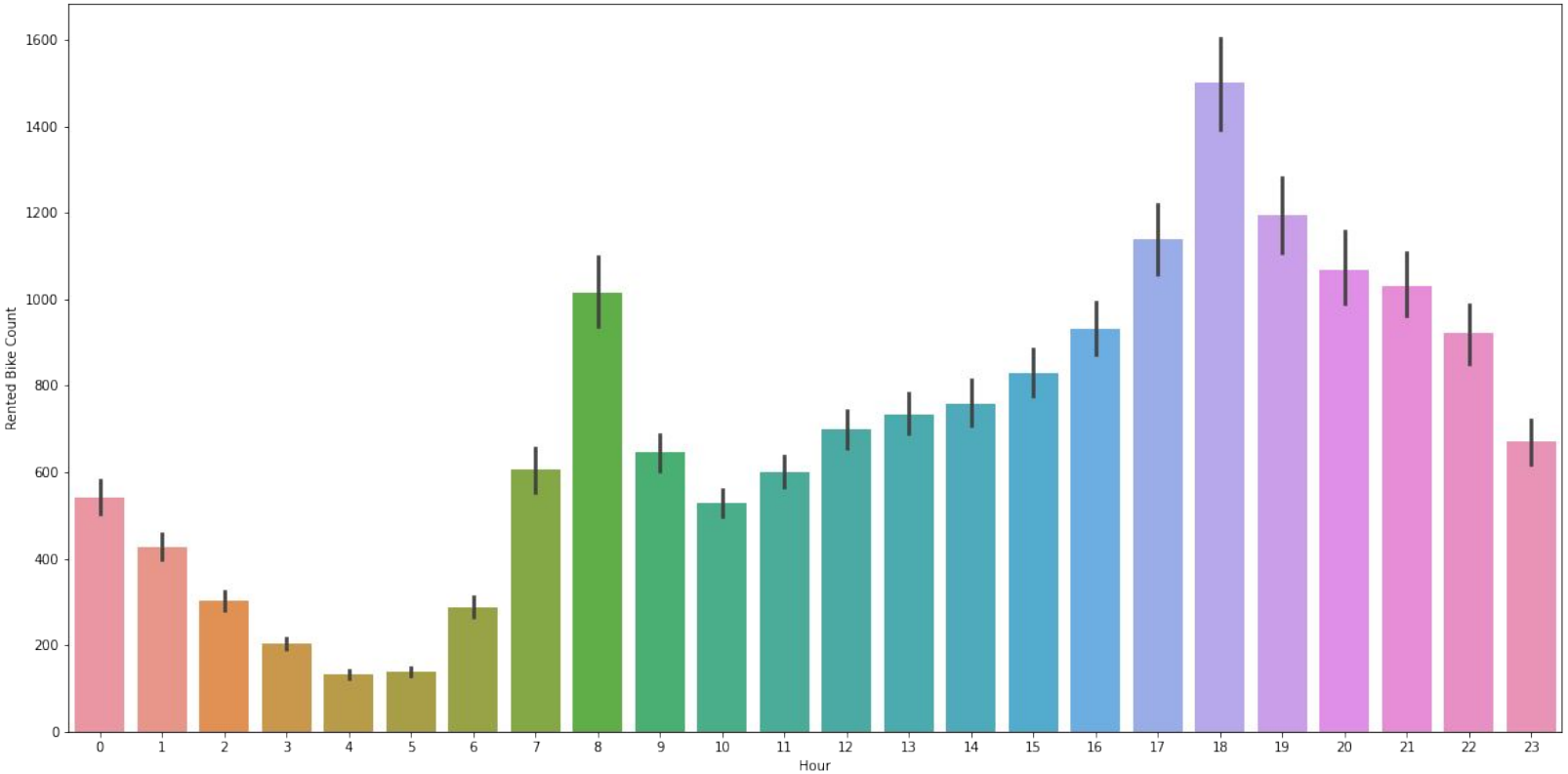


Bike counts vs Functionality day



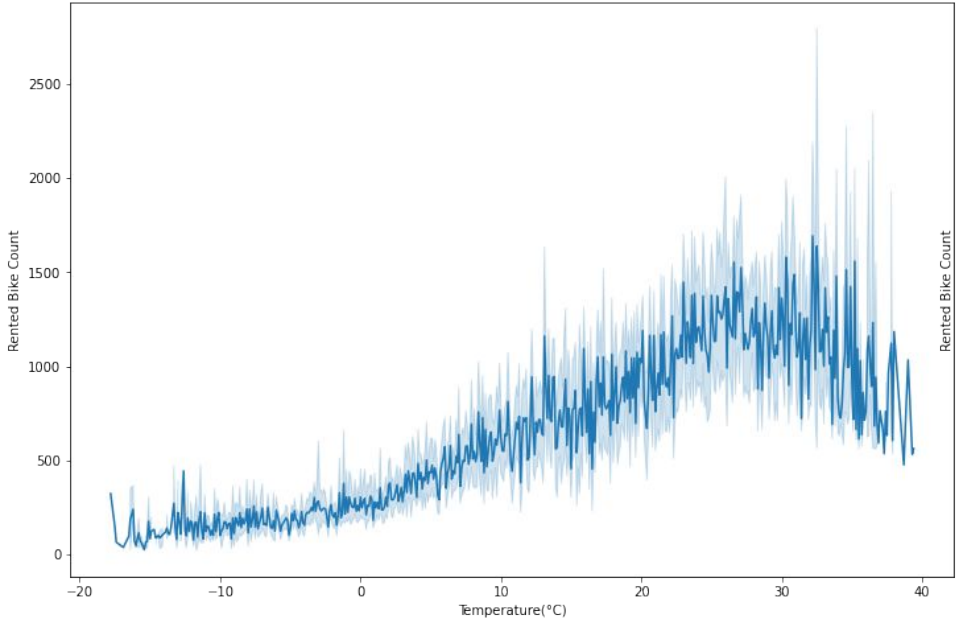
Exploratory Data Analysis

Rented Bike counts across different hours in a day

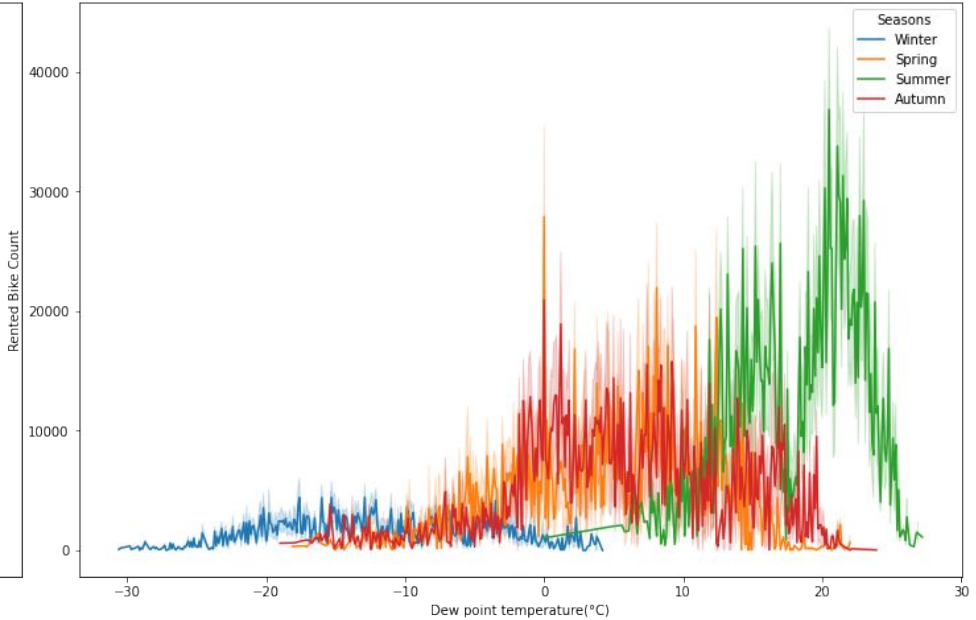


Exploratory Data Analysis

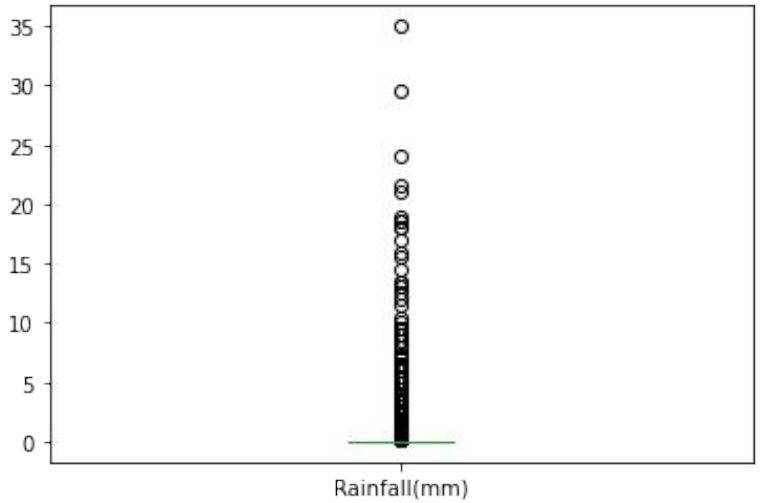
Bike counts vs Temperature

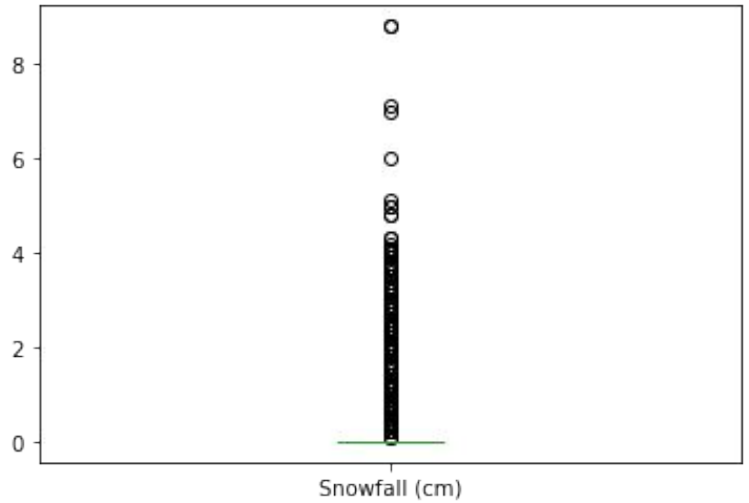


Bike counts vs Dew point temperature

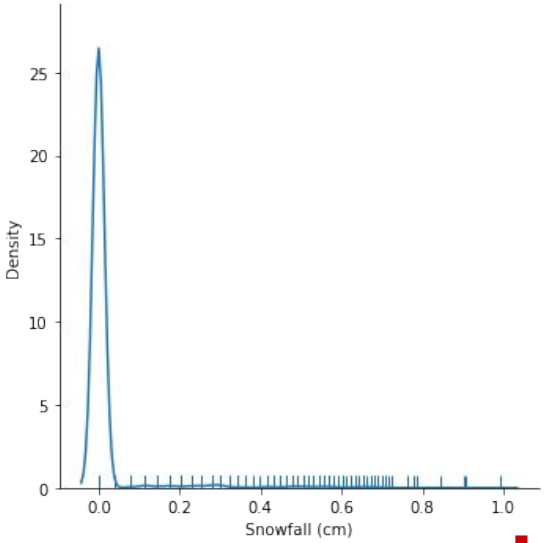


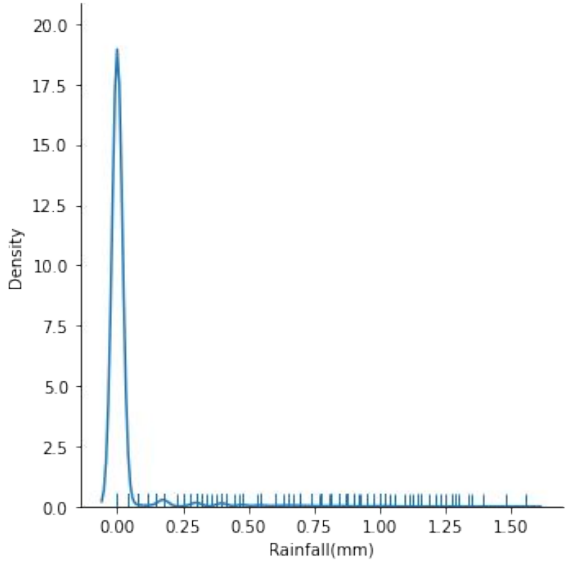
Outliers detection and treatment






```
[ ]  #Outlier treatment
     dataset['Rainfall(mm)']=np.log10(dataset['Rainfall(mm)']+1)
     dataset['Snowfall (cm)']=np.log10(dataset['Snowfall (cm)']+1)
```



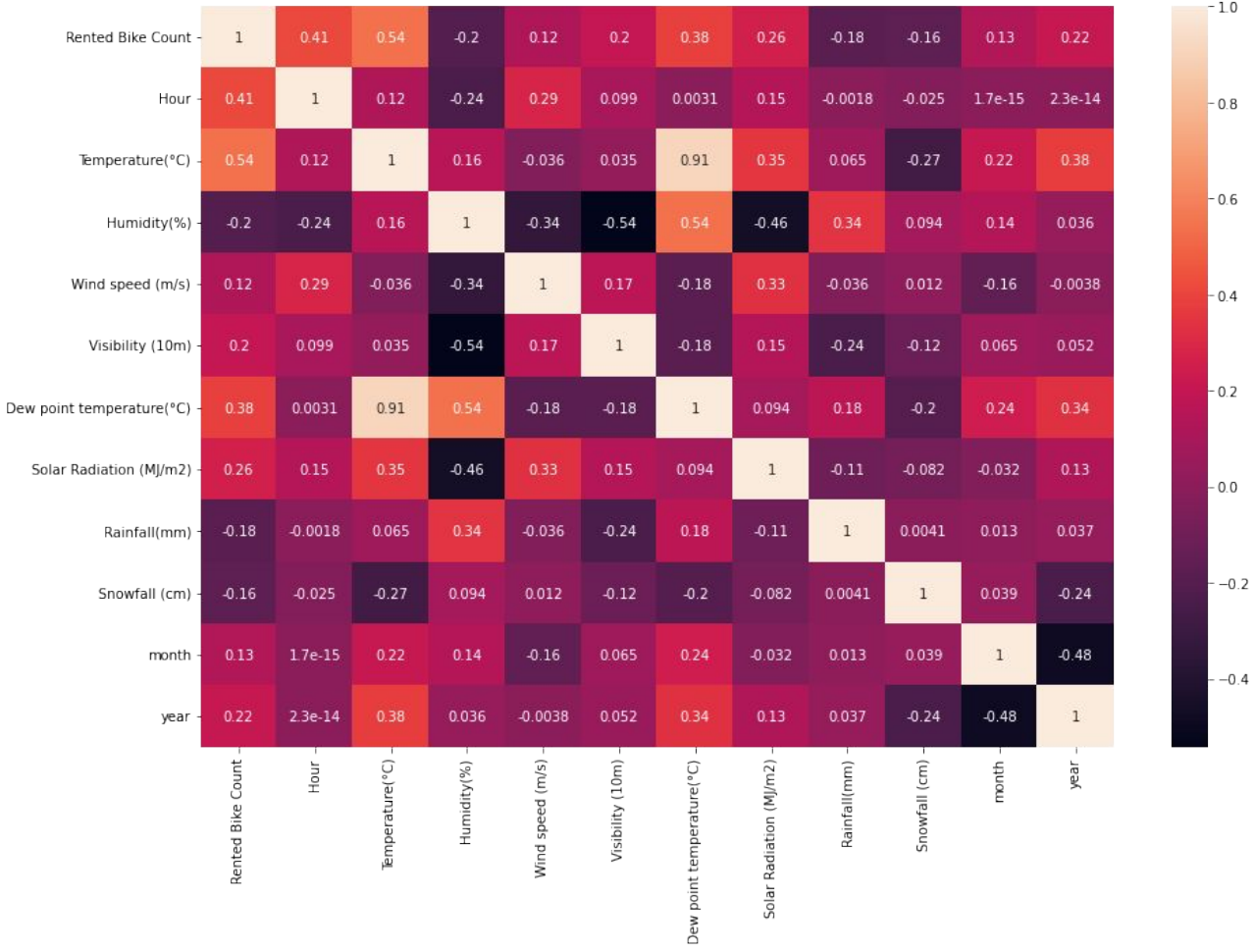


Label Encoding

```
[ ]  newdf['Holiday']=newdf['Holiday'].apply(lambda x : 0 if x=='No Holiday' else 1)
     newdf['Functioning Day']= newdf['Functioning Day'].apply(lambda x : 0 if x=='No' else 1)
```

```
[ ]  #Label encoding for seasons
     from sklearn.preprocessing import LabelEncoder
     label = LabelEncoder()
     newdf['Seasons'] = label.fit_transform(newdf['Seasons'])
```

Correlation Heatmap



Machine Learning Modelling

1. Linear Regression

Linear regression is a **linear model**, e.g. a model that assumes a linear relationship between the input variables (x) and the single output variable (y).

Different techniques can be used to prepare or train the linear regression equation from data, the most common of which is called **Ordinary Least Squares**. It is common to therefore refer to a model prepared this way as Ordinary Least Squares Linear Regression or just Least Squares Regression.

```
[ ]  #Linear Regression Model
     reg = LinearRegression().fit(X_train,y_train)
```

```
[ ]  reg.coef_,reg.intercept_
```

```
(array([[ 2.73622205e+01,  2.94535590e+01, -6.50493831e+00,
          2.05158809e+01,  2.52599859e-02, -8.15362074e+01,
         -8.55960989e+02,  1.55176327e+02, -9.74538663e+01,
         -1.20902651e+02,  9.40765121e+02]]), array([-372.1197669]))
```

```
[ ]  y_pred_train = reg.predict(X_train)
```

```
[ ]  y_pred = reg.predict(X_test)
```

Machine Learning Modelling

2. Polynomial Regression

- Polynomial Regression is a regression algorithm that models the relationship between a dependent(y) and independent variable(x) as nth degree polynomial.
- It is also called the special case of Multiple Linear Regression in ML. Because we add some polynomial terms to the Multiple Linear regression equation to convert it into Polynomial Regression,
- It is a linear model with some modification in order to increase the accuracy.
- The dataset used in Polynomial regression for training is of non-linear nature.
- It makes use of a linear regression model to fit the complicated and non-linear functions and datasets.
- Hence, "In Polynomial regression, the original features are converted into Polynomial features of required degree (2,3,..,n) and then modelled using a linear model."

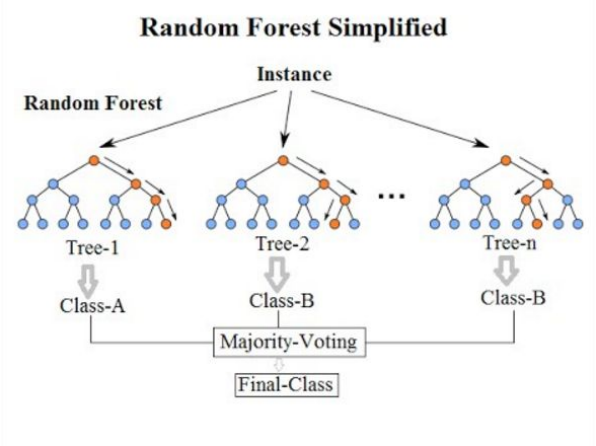
```
[ ] from sklearn.preprocessing import PolynomialFeatures
    poly_regressor = PolynomialFeatures(degree = 2)
    X_poly = poly_regressor.fit_transform(X_train)
    X_poly_test = poly_regressor.fit_transform(X_test)
    lin_reg = LinearRegression()
    lin_reg.fit(X_poly, y_train)

    LinearRegression()
```

```
#Predicting training and testing dataset
y_pred_poly_train = lin_reg.predict(X_poly)
y_pred_poly_test = lin_reg.predict(X_poly_test)
```

Machine Learning Modelling

3.Random Forest Regressor

- Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML.
- It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model.
- Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset.
- Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output.
- The greater number of trees in the random forest leads to higher accuracy and prevents the problem of overfitting.




```
rand = RandomForestRegressor(n_estimators=100, random_state = 0)
rand.fit(X_train, y_train)
```

```
RandomForestRegressor(random_state=0)
```

```
#prdicting on training dataset
y_reg_pred = rand.predict(X_train)
```

```
#predicting on test dataset
y_reg_pred_test = rand.predict(X_test)
```

Machine Learning Modelling

Hyper-parameter Tuning

Hyperparameter Tuning on Random forest model

```
[ ] #importing gridsearch
    from sklearn.model_selection import GridSearchCV
```

```
[ ] #Assigning the parameters
    parameters = {'criterion':['squared_error','absolute_error','poisson'], 'max_features': ['auto','sqrt','log2']}
```

```
[ ] #Performing the gridsearch using the parameters with cross valudation number as 5
    grid = GridSearchCV(rand,parameters,cv=5,scoring = 'neg_mean_squared_error')
    grid.fit(X_train,y_train)
```

```
GridSearchCV(cv=5, estimator=RandomForestRegressor(random_state=0),
             param_grid={'criterion': ['squared_error', 'absolute_error',
                                       'poisson'],
                         'max_features': ['auto', 'sqrt', 'log2']},
             scoring='neg_mean_squared_error')
```

```
[ ] grid.best_params_
```

```
{'criterion': 'absolute_error', 'max_features': 'sqrt'}
```

```
    #We will now use best parameters on our model
    rand2 = RandomForestRegressor(n_estimators = 100, random_state = 0, criterion = 'absolute_error', max_features ='sqrt', max_depth =15)
```

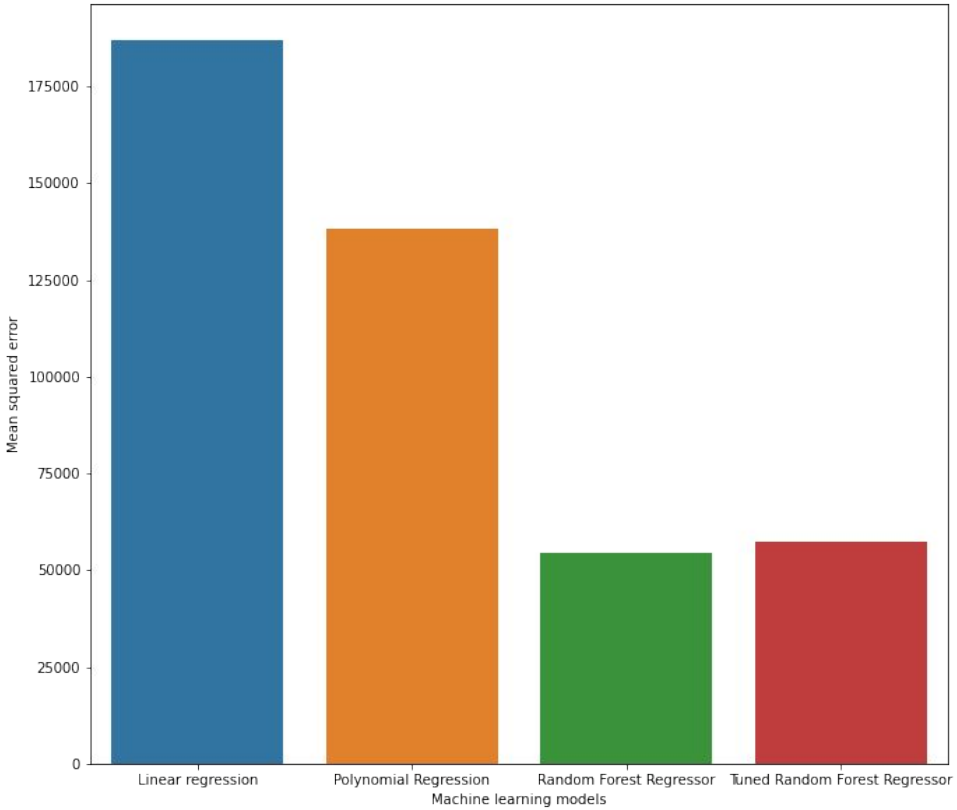
```
[ ] rand2.fit(X_train,y_train)
```

```
RandomForestRegressor(criterion='absolute_error', max_depth=15,
                      max_features='sqrt', random_state=0)
```

```
[ ] y_rand = rand2.predict(X_train)
    y_rand_test = rand2.predict(X_test)
```

Regression Evaluation Metrics

Mean Squared Error



$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

MSE = mean squared error

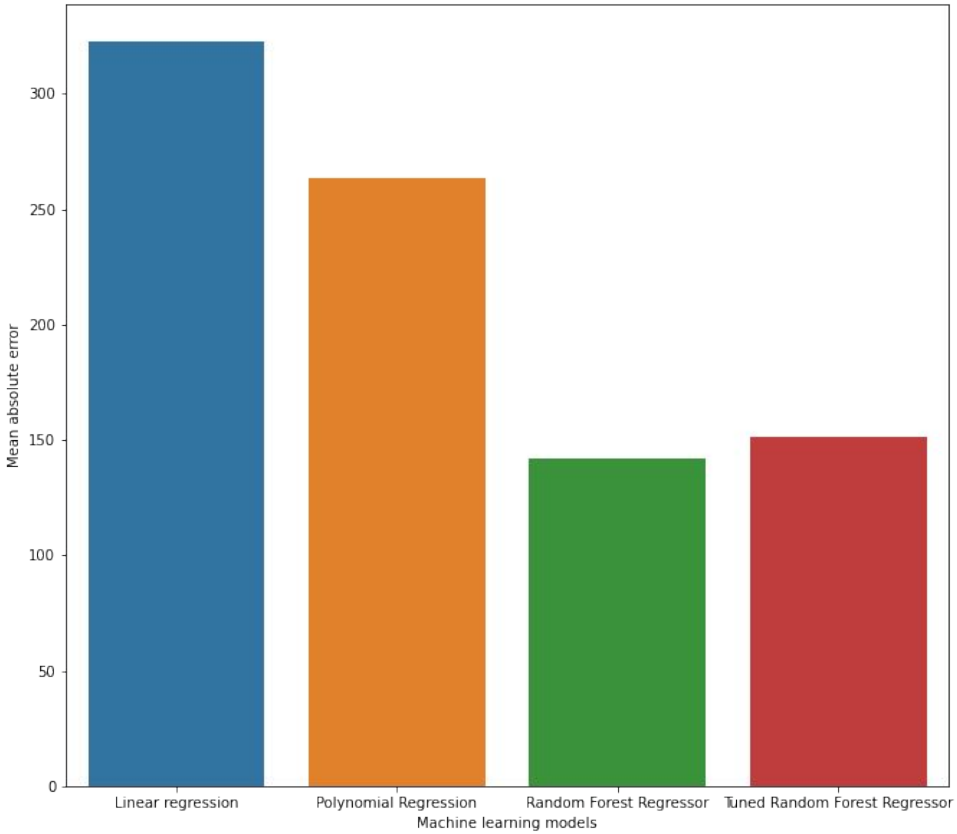
n = number of data points

Y_i = observed values

$\hat{Y}_i$ = predicted values

Regression Evaluation Metrics

Mean Absolute Error



$$MAE = \frac{1}{n}\sum \left| y - \hat{y} \right|$$

Divide by the total number of data points

Predicted output value

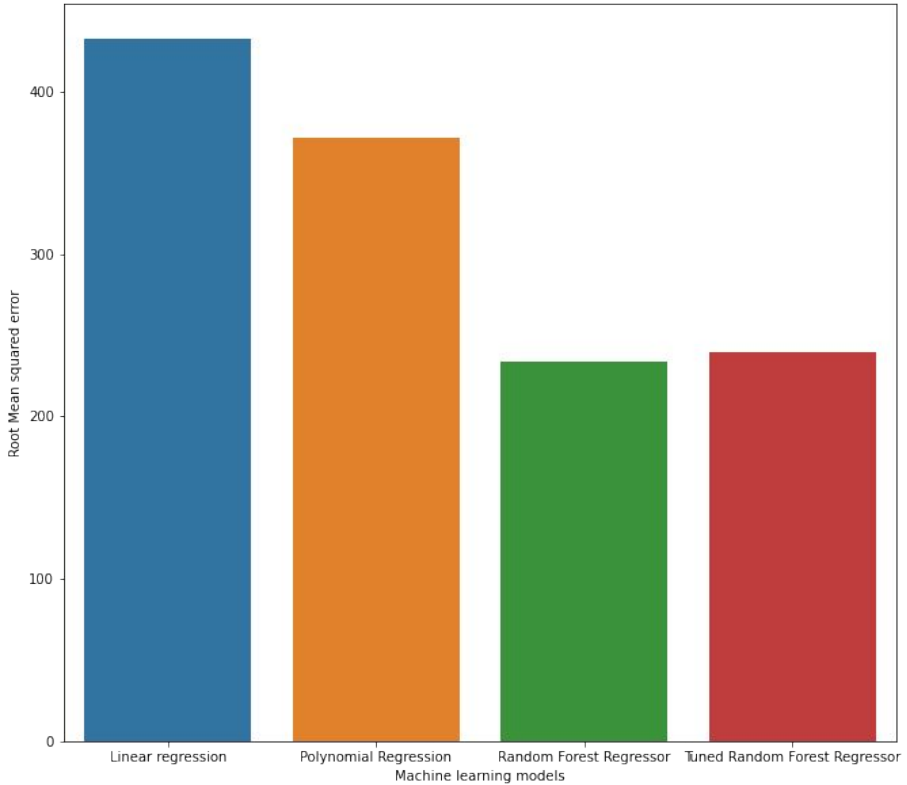
Actual output value

Sum of

The absolute value of the residual

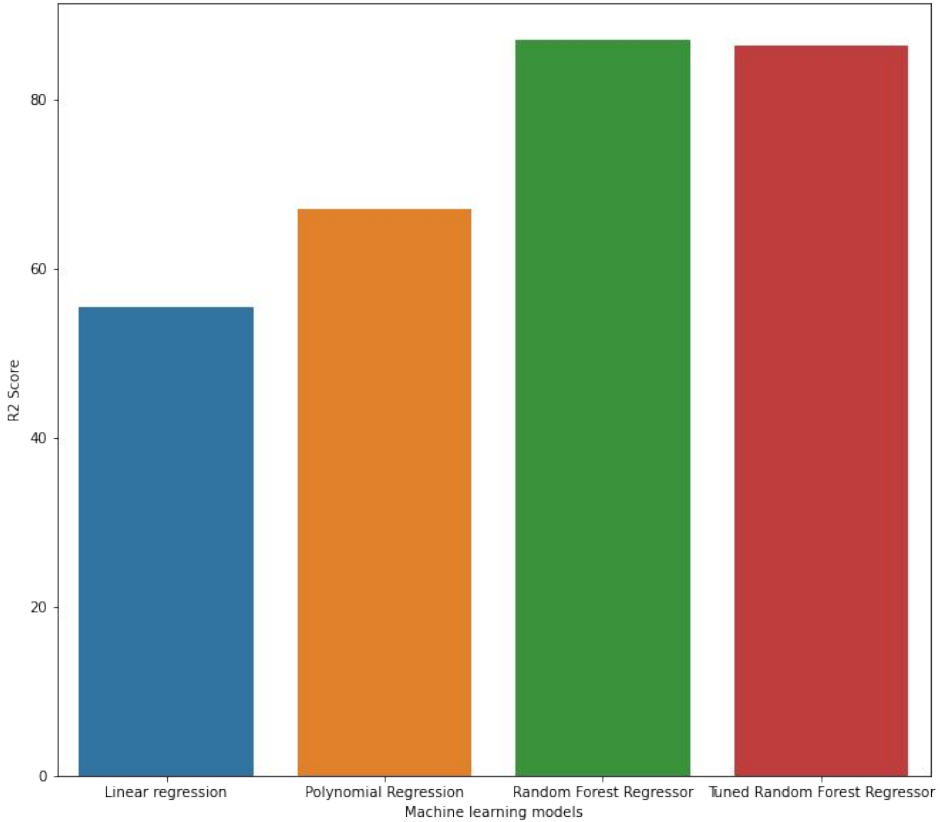
Regression Evaluation Metrics

Root Mean Squared Error



Regression Evaluation Metrics

R2 Scores



$$R^2 = 1 - \frac{SS_{RES}}{SS_{TOT}} = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \overline{y})^2}$$

Conclusion

1. We have used three regression models in our analysis to predict the rented bike count that are Linear Regression, Polynomial Regression and Random Forest Regressor.
2. We have used Hyperparameter tuning on Random forest regressor using GridSearchCV to find the best parameters.
3. Random Forest Regressor is the best model among the three with least errors and highest R2 score of 98.21% for training set and 86.98% for test dataset.
4. The peak time is 6pm where highest number of bookings are rented.
5. Approximately between 10 degree Celsius to 20 degree Celsius the rent booking is peak.
6. Number of bookings are very much higher in Summer season.
7. 'Rainfall' and 'Snowfall' has a huge impact on number of bikes rented.
8. Bike renting is high on Functional days (i.e. No holiday).

References

1). https://pandas.pydata.org/

2). https://matplotlib.org/

3). https://seaborn.pydata.org/

4). Geek for geeks

Thank You