Bike Sharing Demand Prediction

Presented By

Tanay Tupe

Contents

* **Business domain**
* **Project Problem Description**
* **Dataset Summery**
* **Understanding Each Variable**
* **Modules in project**
    1. Importing necessary libraries, Loading and understanding data
    2. Exploratory Data Analysis
    3. Preparation for prediction model
    4. Implementing regression models
* **Conclusion**

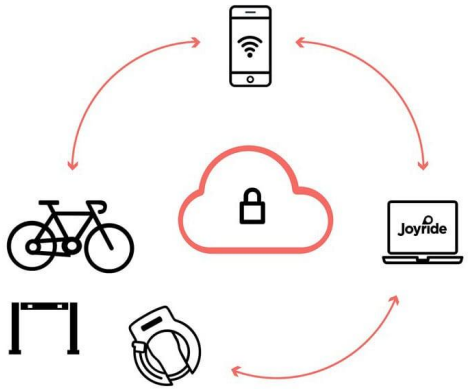
About “Bike Sharing” business domain

A bike rental or bike hire business rents out bicycles 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 a vehicle, typically travellers and particularly tourists.

Project Problem Description

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.

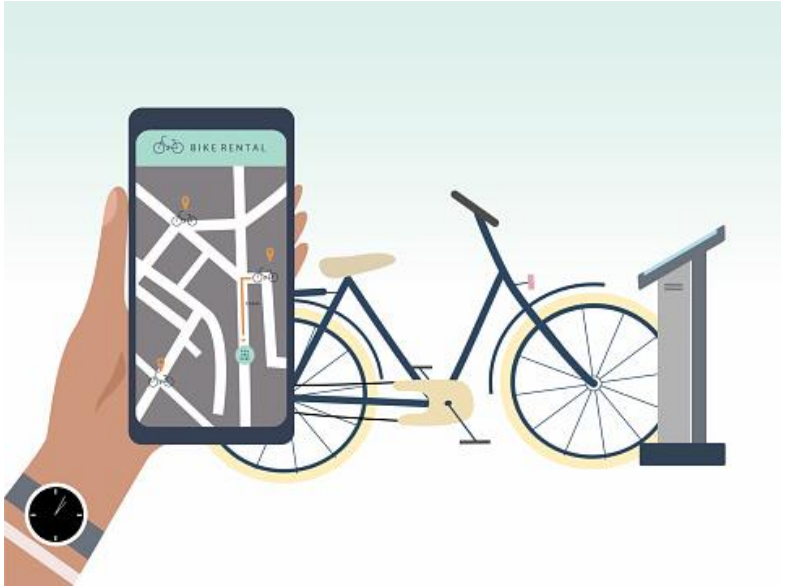
Eventually, providing the city with a stable supply of rental bikes becomes a major concern. The crucial part is the prediction of bike count required at each hour for the stable supply of rental bikes.



Dataset Summery

The dataset we have used here is “Seoul Bike Sharing dataset”.This dataset in it’s raw form has around “8760” observations and “14” columns and it’s a mix between categorical and numeric values.

Also, we have "0" null values and no duplicated observations throughout our dataset. Though its highly unlikely to come across such datasets, here it works in our favour.



Understanding Each Variable

Let’s Get Familiar With Each feature Present In The DataSet.

1. **Date** : Date recorded for a particular observation. (Date-Time)

2. **Rent Bike Count** : Hourly bikes rented record. (continuous) **(Target variable)**

3. **Hour** : 24-Hours record for each day. values from “00:00” to “23:00”. (Discrete)

4. **Temperature** : Temperature recorded hourly in Celsius. (continuous)

5. **Humidity** : Water content in atmosphere in percentage. (Discrete)

6. **Wind speed** : Velocity of wind recorded in meter per second. (continuous)

7. **Visibility** : The degree of clearness of the atmosphere. (continuous)

Understanding Each Variable (Continued)

8. **Dew point temperature** : Temperature below which the water vapour will condense into liquid water. Recorded hourly. (continuous)

9. **Solar Radiation** : Electromagnetic radiation emitted by the sun. In watt per square metre. (continuous)

10. **Rainfall** : Hourly record of water poured in “mm”. (continuous)

11. **Snowfall** : Hourly record of snow poured in “cm”.(continuous)

12. **Seasons** : Each of the four divisions of the year. (Categorical)

13. **Holiday** : Record of working day and non-working day (Categorical)

14. **Functioning Day** : NoFunc(Non Functional Hours), Fun(Functional hours). (Categorical)

Module 1 Importing necessary libraries, Loading and understanding data.

Before anything, we'll be needing all relevant libraries as required in our analysis. Few universal libraries like Numpy, Pandas, Matplotlib, Seaborn are must needed, and are imported even before anything, but we also will be needing other libraries as we go through our analysis.

```
# Importing all necessary libraries.
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.preprocessing import MinMaxScaler, StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import Lasso, Ridge, LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error

from scipy.stats import zscore

from statsmodels.stats.outliers_influence import variance_inflation_factor

import warnings
warnings.filterwarnings('ignore')
```

Module 1 Importing necessary libraries, Loading and understanding data.

As this analysis was performed on “Google Colab” we first need to mount the notebook to our drive, where we have our “Seoul Bike Sharing Demand Prediction” dataset in “CSV” format.

Checks were performed on dataset’s Size, Shape, general information on dataset with info() and describe() function. Where we observed number of rows(8760), features (14), different data type of our features, the count of their null values, memory(958.2+ KB) the dataset is occupying.

Also, we have a general overview of numerical features and we can already make rough estimations over it. Like,

* Our target variable appears to have slight skewness.
* Main factors affecting could be "hour of the day", "Rainfall", "Temperature" etc.
* Median of each columns.

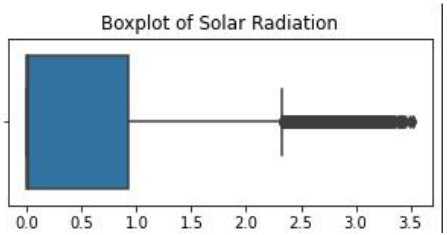
Module 2 Exploratory Data Analysis.

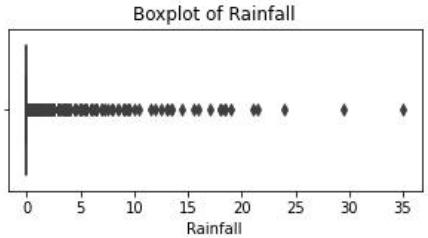
We began our EDA with univariate analysis. Where checks have been performed for type of distribution, skewness and outliers.

Considering outliers, we encountered outliers in few important features like “Solar Radiation” and “rainfall” even our target/dependent variable.

We can clearly see that some variables have a lot of outliers. Still, we will be keeping them in our analysis and check how models fits.

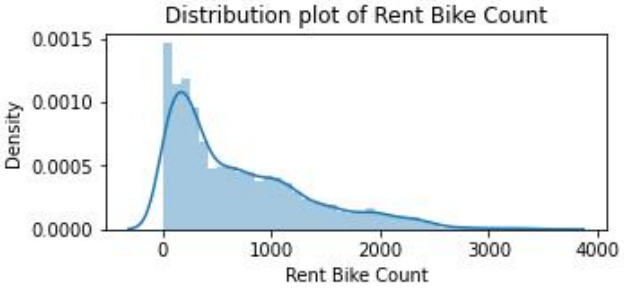
Reason we are considering outliers is because few important variables like "Rainfall", "Snowfall" have a ton of them, and deleting them may result in information loss from dataset.

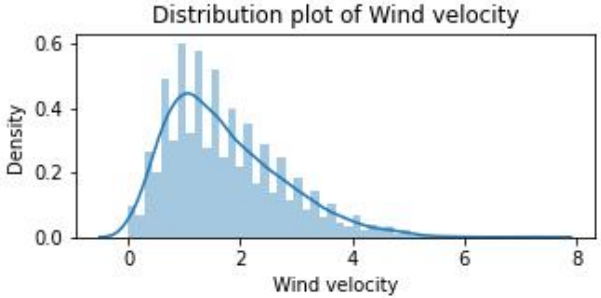


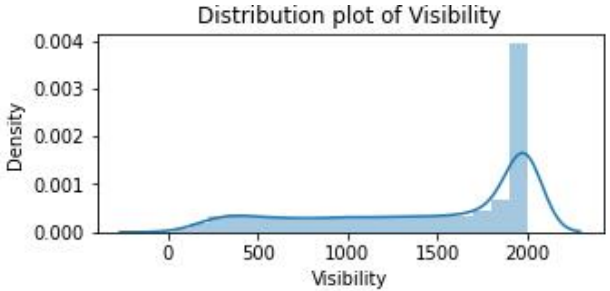


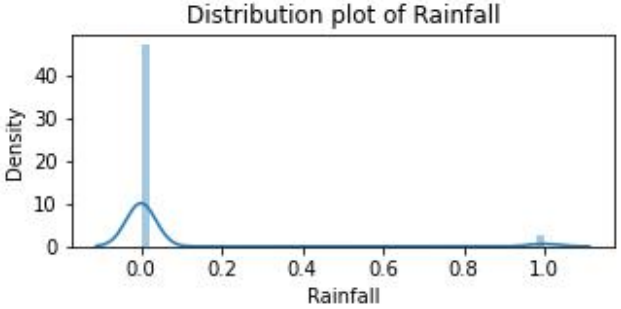
Module 2 Exploratory Data Analysis.

Analyzing distributions of variables.







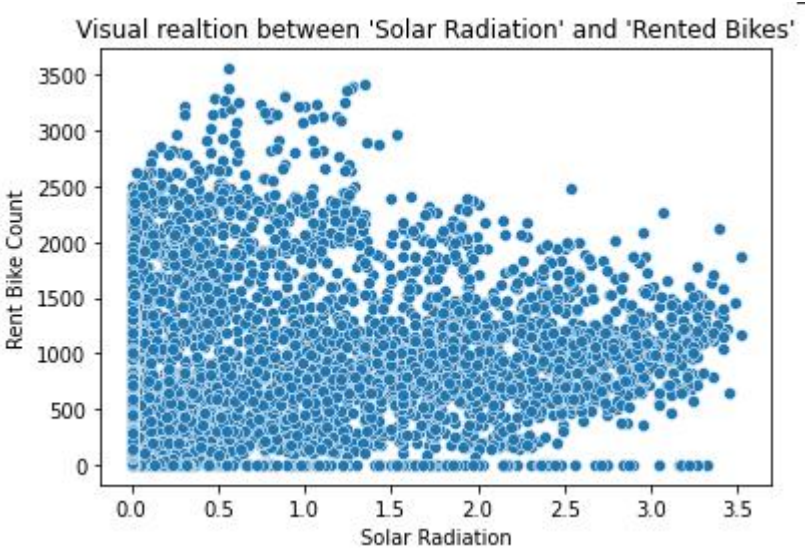


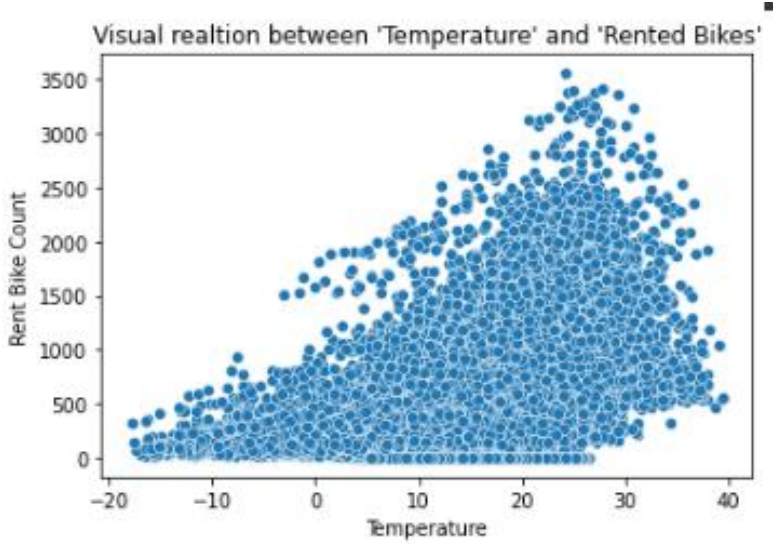
Here we can witness most of the distribution are not normal, skewness can be found in every other variable.

Initial guess here is " Tree based models will work better than linear regression "

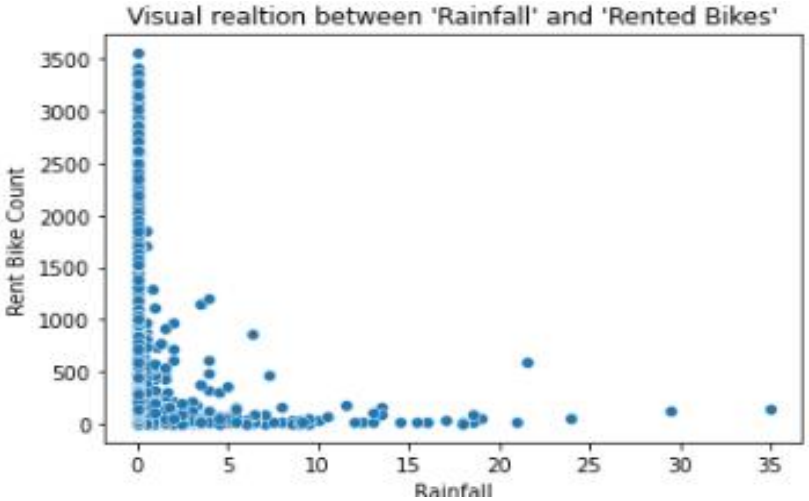
Module 2 Exploratory Data Analysis.

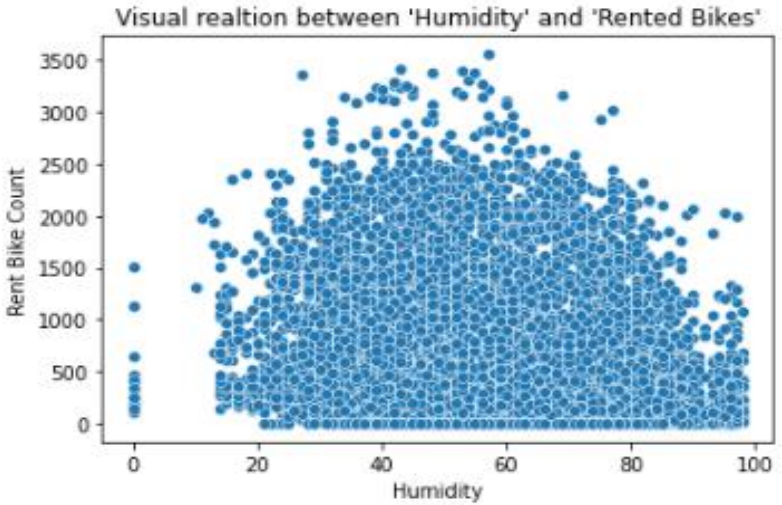
Bivariate analysis. Checking for linear relationship between dependent and independent variables.





Module 2 Exploratory Data Analysis.





In this, and slide before we can again observe that, there ain't enough linear relation between dependent and independent variables.

As thought Linear Regression will be an issue on this dataset. But again, our main focus will be on Tree based models for this analysis.

Module 2 Exploratory Data Analysis.

Check for multicollinearity using “Variance Inflation Factor”.

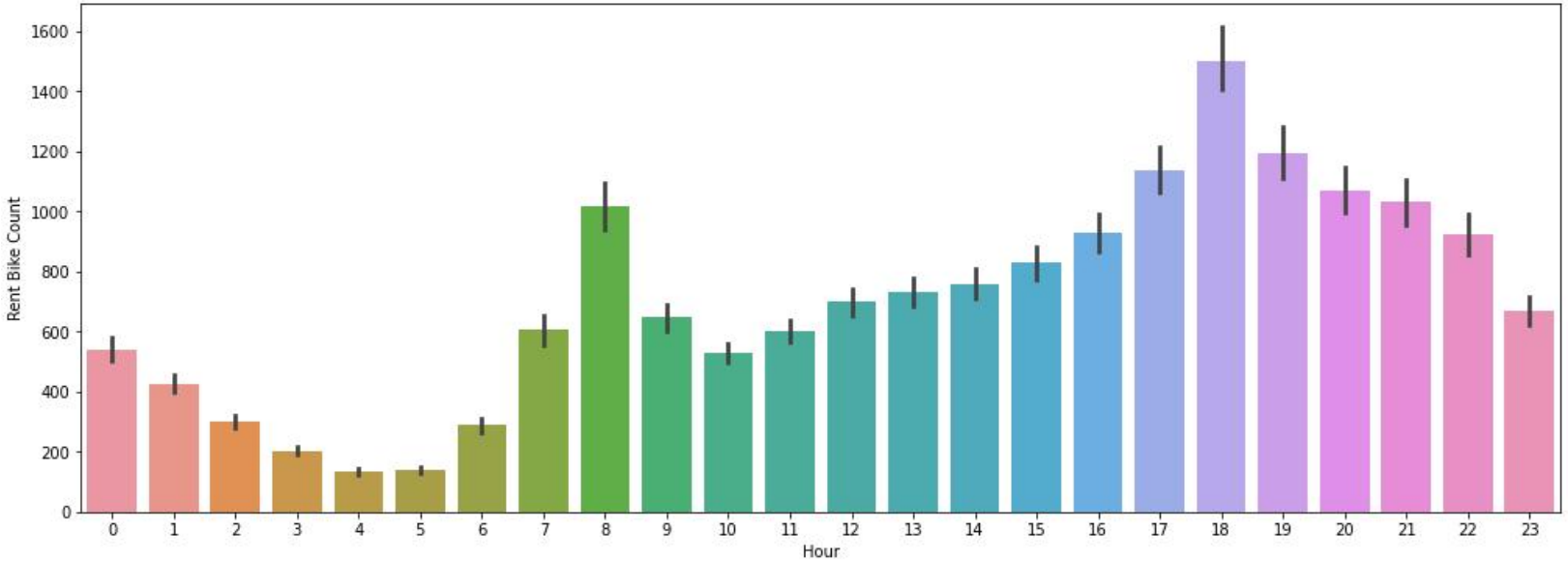
Here we can clearly see "Temperature" and "Dew.P Temperature" have a little high "VIF".

"VIF" can be a problem considering Linear Regression, but Tree based models will not be affected with multicollinearity.

variables	VIF
Hour	4.418242
Temperature	33.385256
Humidity	5.371996
Wind velocity	4.805364
Visibility	9.085977
Dew.P Temp	17.126199
Solar Radiation	2.881590
Rainfall	1.081567
Snowfall	1.120833

Module 2 Exploratory Data Analysis.

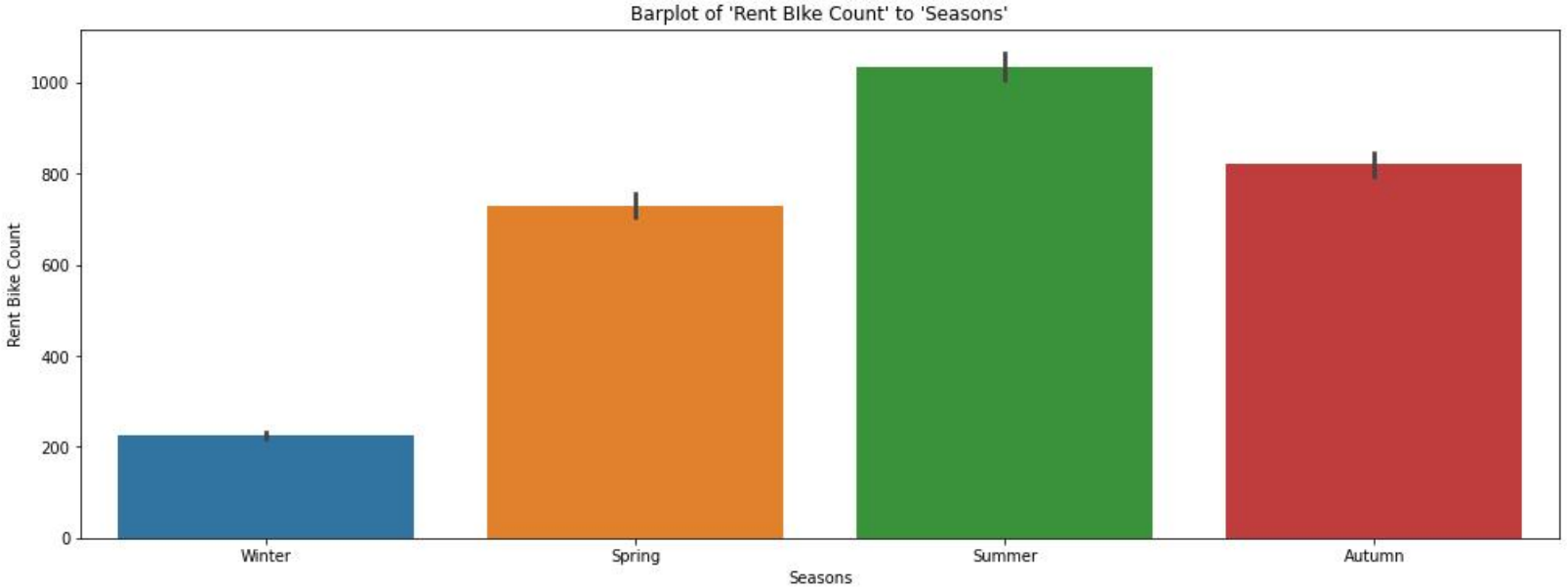
Barplot of Bike rented per hour.



As we can see above, Bikes are mostly rented on "8:00 AM" and "5:00 PM" to "10:00 PM".

Module 2 Exploratory Data Analysis.

Barplot of Bike rented per hour.



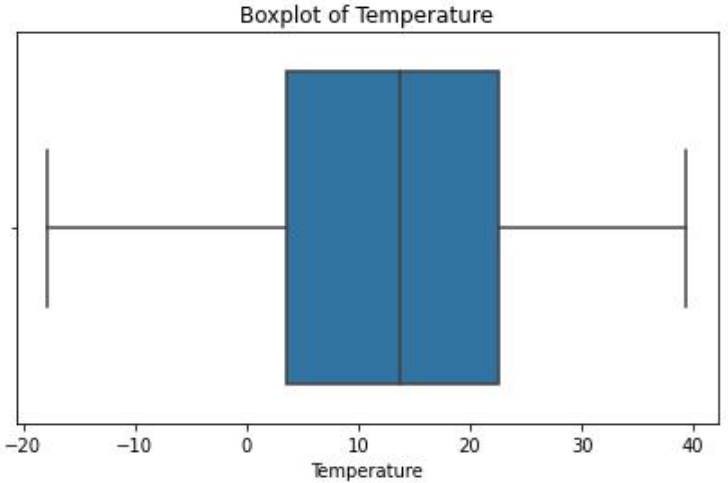
People mostly rent bikes on "Spring", "Summer" and "Autumn" as temperature is favourable during this period of year.

Module 2 Exploratory Data Analysis.

Checking for temp range and average minimum and maximum temperature.

Temperature in winters drops down as low as **“-19 Degree Celsius”**, with mean temp in winters being **“-2.5 Degrees”**.
This explains why number of bike rented are low in this particular season.

Whereas, temperature in rest of the seasons are quite favourable with mean temp in summer recorded as **“26.5 Degrees”** .



Module 3 Preparation for prediction model

Vectorizing continuous variables.
Solar Radiation : Continuous feature categorized to “0” and “1”. Split point on mean of data.
Rainfall : Continuous feature categorized to “0” and “1”. Split point on mean of data.
Snowfall : Continuous feature categorized to “0” and “1”. Split point on mean of data.

Vectorizing categorical variables.
Holiday : Mapping “No Holiday” to “0” and “Holiday” to “1”.
Functioning Day : Mapping “No” to “0” and “Yes” to “1”.

Finally Vectorizing using Dummies.
Seasons : Divided into “Seasons_Autumn”, “Seasons_Spring”, “Seasons_Summer”, “Seasons_Winter”

Dividing data into independent and dependent set and Splitting data into Train and Test sets.

Module 4 Implementing regression models

Implementing Linear Regressor.

The reason we are passing data to a Linear regressor is just to prove that these type of regression models won't actually fit very closely to target variable in dataset like these. As it was evident enough during analysis. Still, let's pass data to regressor and verify using evaluation metrics. We will use Cross-Validation for Linear regressor. And, instead of normal Linear Regression, we'll be using Regularized Regression. (Lasso Regressor)

Best Alfa here was: 0.1
Train RMSE for lasso regressor is: 423.3
Test RMSE for lasso regressor is: 431.6
Train R-sq for lasso regressor is: 0.56
Test R-sq for lasso regressor is: 0.55

Module 4 Implementing regression models

Implementing Decision Tree Regressor.

As decision tree regressor was our initial guess, let's see how this regressor performs. Decision tree is a flowchart-like structure and splits the data on Entropy or Information gain, which is a complete different approach unlike mapping a function in Linear regression. Let's pass data to regressor and verify using evaluation metrics.

Train RMSE for Decision Tree regressor is: 0.0
Test RMSE for Decision Tree regressor is: 320.18
Train R-sq for Decision Tree regressor is: 1.0
Test R-sq for Decision Tree regressor is: 0.75

Module 4 Implementing regression models

Implementing Random Forest.

As we saw in previous slide, Decision Tree did a better job then Linear regression. But using single tree approach was not very beneficial. Reason being, the Decision Tree over-fitted to data way too much on training set. We can play and experiment with different hyper parameters or we could consider Random Forest to deal with overfitting.

Train RMSE for Random Forest regressor is: 84.04
Test RMSE for Random Forest regressor is: 235.67
Train R-sq for Random Forest regressor is: 0.98
Test R-sq for Random Forest regressor is: 0.86

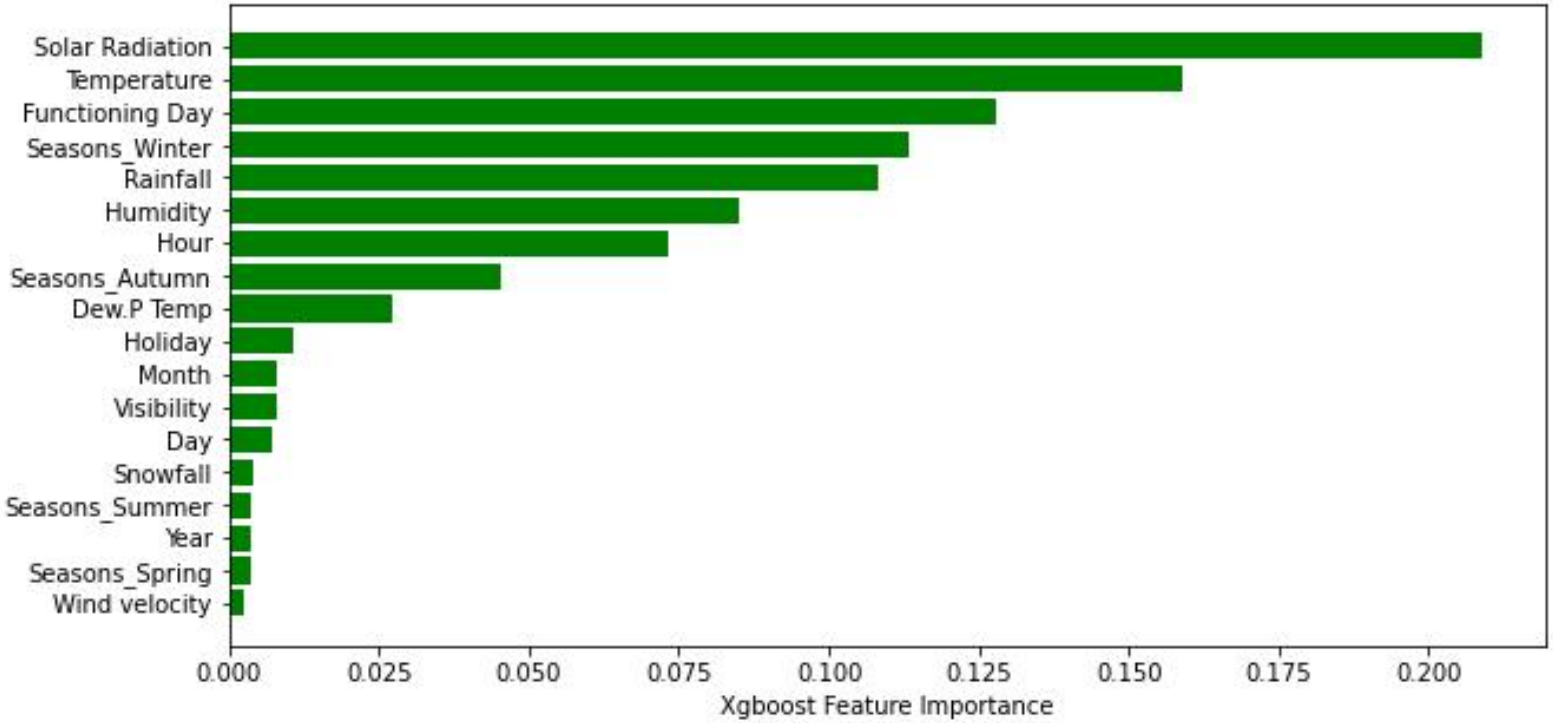
Module 4 Implementing regression models

Implementing Extreme-Gradient Boosting.

XGBoosting is an implementation of gradient boosting trees algorithm. We are trying to close the variance gap by using another variation of tree based Ensemble technique. Let's check the metrics and compare all the metrics from previous algorithms.

Train RMSE for XGB regressor is: 245.24
Test RMSE for XGB regressor is: 260.53
Train R-sq for XGB regressor is: 0.85
Test R-sq for XGB regressor is: 0.83

Feature Importance.



Conclusion

Considering VIF, distribution plots and scatterplots we concluded that, Linear regerssion won't fit this dataset well enough. Implementing Linear regression could cause high variance and could not be best model selection for this analysis.

When plotting a barplot of target variable with respect to time, we found most bike were rented on "08:00" A.M and between "05:00 PM" to "10:00 PM". Most people who rents bike might be working professional and office staff with no personal transport, as the peak rent time matches with office timings.

We also observed a significant drop in rentals during winter seasons. Temperature averages to "-2 Deg Celsius" in winters with records as low as "-17 Deg Celsius". Whereas in summers temperatures are quite favourable averaging to "26.5 Deg Celsius".

We first intentionally implemented Linear regressor just to prove that Linear models won't actually fit very closely to target variable in dataset like these. As it was evident enough during analysis, considering VIF, feature distributions and their Scatterplots of independent to dependent variable.

Conclusion

Decision tree regressor was our initial guess, which is a complete different approach unlike mapping a function in Linear regression. It did a better job then Linear regression. But using single tree approach was not very beneficial. Because Decision Tree over-fitted to data way too much on training set, resulting in high variance.

To overcome this variance we could have experimented with hyper parameters or go with Ensemble techniques, and chose to go with Random forest models. But the model still had a hint of variance.

To close this variance gap even more and generalize our model we used XGBoost regressor. where we finally had promising metrics which were generalized and had minimal variance. Here model had lowest variance and best R-square value for train and test set.

Most important feature according to XGBoost model are “Solar Radiation”, “Temperature”, “Functioning Day” , “Season (winter)” and “Rainfall”.