Prediction of Bike Rental Count in R

Sagar B.

2nd September 2019

Contents

Chapter 1

Introduction

1.1 Instructions to run the code file

- Extract the ‘Bike_Rental_Prediction.zip’ file.
- There will be two folders “Python” and “R”.
- Make sure R studio is installed in your PC.
- From the “R” folder open “Bike_Rental.R”
- R Studio will open the above file.

1.2 Problem Statement

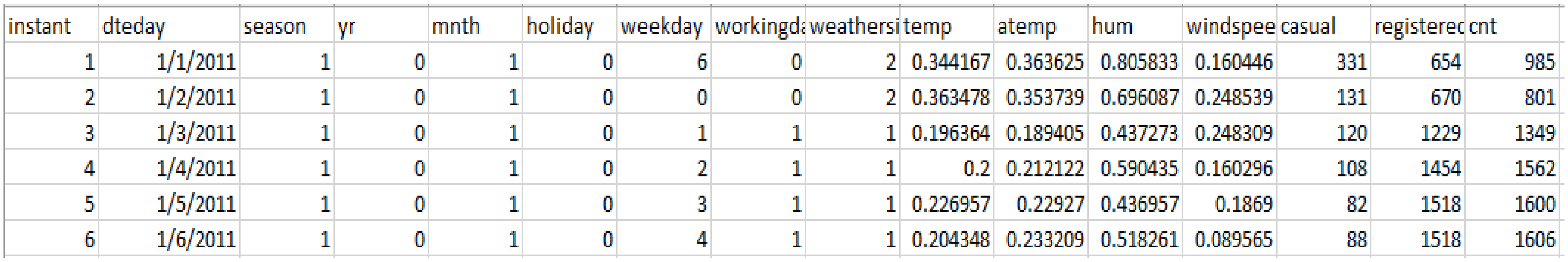
The objective of this Case is to Predication of bike rental count on daily basis based on the environmental and seasonal settings.

1.3 Dataset

Note that only one dataset with 731 observations was provided and there was no separate test dataset. The dimensions of the dataset were 731 x 16. The features in the dataset were as follows:

- Instant – It is the index number of the observations.
- Dteday – Date when the observation was recorded.
- Season – 1 : spring , 2 : summer, 3 : fall, 4 : winter
- Yr – 0 : 2011 , 1 : 2012
- Mnth – Months in a year from 1-12
- Holiday – 1 : Holiday , 0 : Not a Holiday
- Weekday – Day of the week from 0-6
- Workingday – 1 : Neither a holiday nor weekend , 0 : Holiday
- Weathersit – 1: Clear, Few clouds, Partly cloudy, Partly cloudy
  2: Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist
  3: Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds
  4: Heavy Rain + Ice Pallets + Thunderstorm + Mist, Snow + Fog
- Temp – Temperature on that day (Normalized)
- Atemp – Feeling temperature (Normalized)
- Hum – Humidity (Normalized)
- Windspeed – windspeed (Normalized)
- Casual – Number of casual users who rented bike .

- Registered – Number of registered users who rented bike.
- Cnt – Total count Casual + Registered.



instant	dteday	season	yr	mnth	holiday	weekday	workingda	weathersi	temp	atemp	hum	windspee	casual	registerec	cnt
1	1/1/2011	1	0	1	0	6	0	2	0.344167	0.363625	0.805833	0.160446	331	654	985
2	1/2/2011	1	0	1	0	0	0	2	0.363478	0.353739	0.696087	0.248539	131	670	801
3	1/3/2011	1	0	1	0	1	1	1	0.196364	0.189405	0.437273	0.248309	120	1229	1349
4	1/4/2011	1	0	1	0	2	1	1	0.2	0.212122	0.590435	0.160296	108	1454	1562
5	1/5/2011	1	0	1	0	3	1	1	0.226957	0.22927	0.436957	0.1869	82	1518	1600
6	1/6/2011	1	0	1	0	4	1	1	0.204348	0.233209	0.518261	0.089565	88	1518	1606

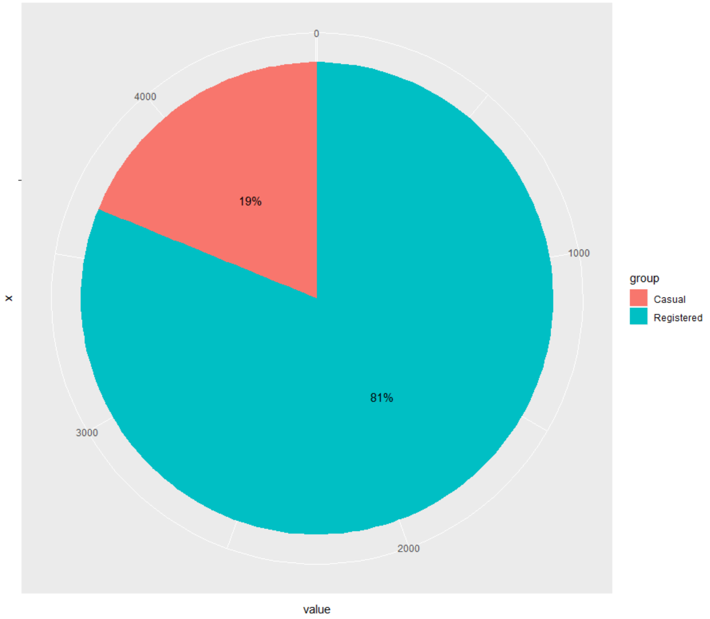
Above is a sample of the dataset on which we will be working on. The table shows first 6 rows with all its features.

Chapter 2

Data Cleaning and Exploratory Data Analysis

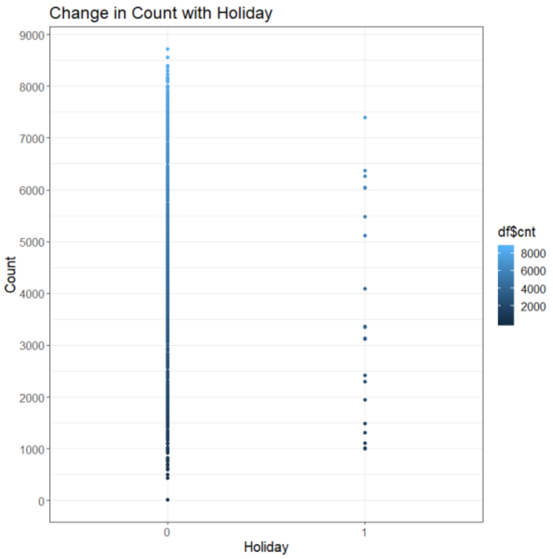
- First, we started off by importing all the necessary libraries in the R environment.
- Then, the training dataset 'day.csv' was loaded in the environment.
- Variable 'instant' was removed, as it was just an index of all the observations.
- Also, date was extracted from 'dteday' and the variable was removed.
- The dataset was checked for missing values. There were no missing values in the dataset.

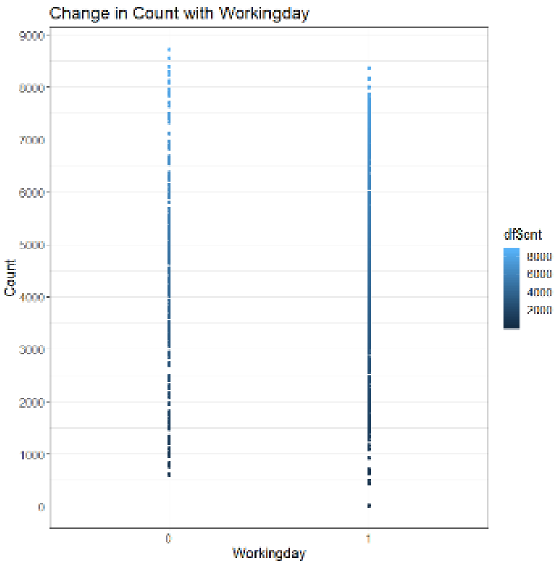
2.1 Pie Chart for comparing Casual and Registered users



- Then, we did some exploratory data analysis. First, we checked the percentage of casual and registered users which account for the total rental count. The following pie chart was obtained. We can see that casual users account for only 19% of the total count whereas registered users account for 81%.

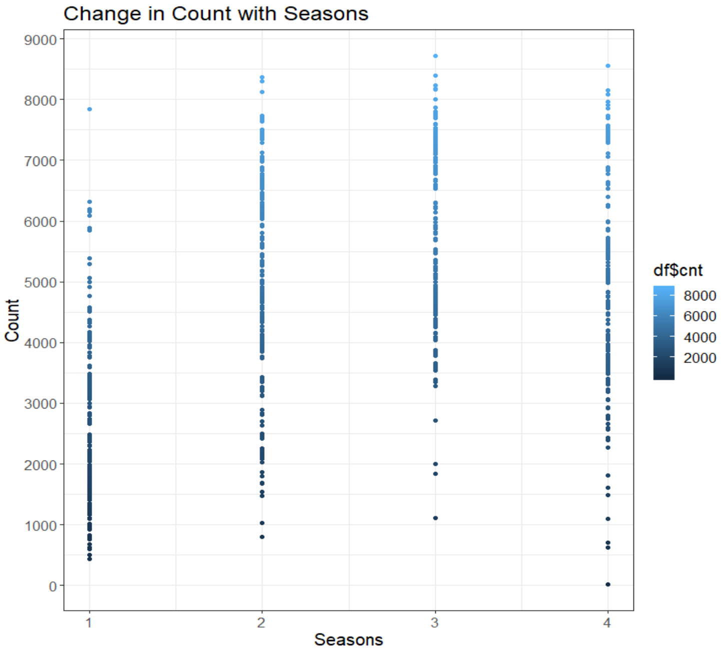
2.2 Effect of holiday and workingday on rental count





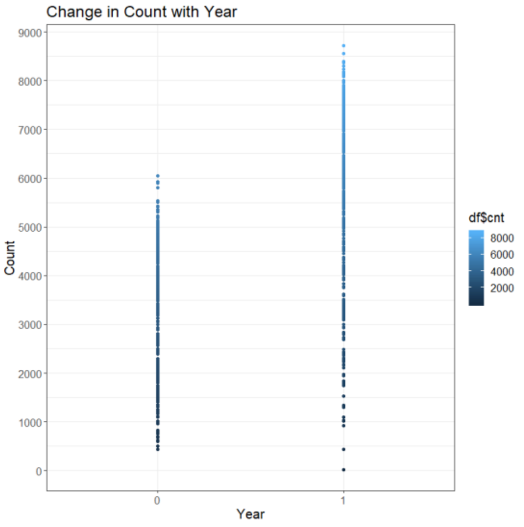
- On the left, we see how holiday affect the count and, on the right, we see how workingday affect the count. From both graphs we can infer that count as well as demand is more on working day than on holidays.

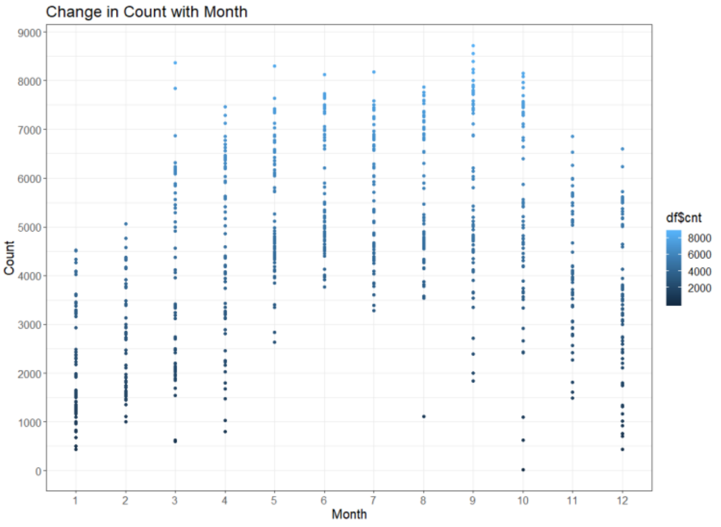
2.3 Effect of season on rental count



- Now, we observe how count varies with seasons. It is seen that in spring season the count is least and for other three seasons the count seems similar.

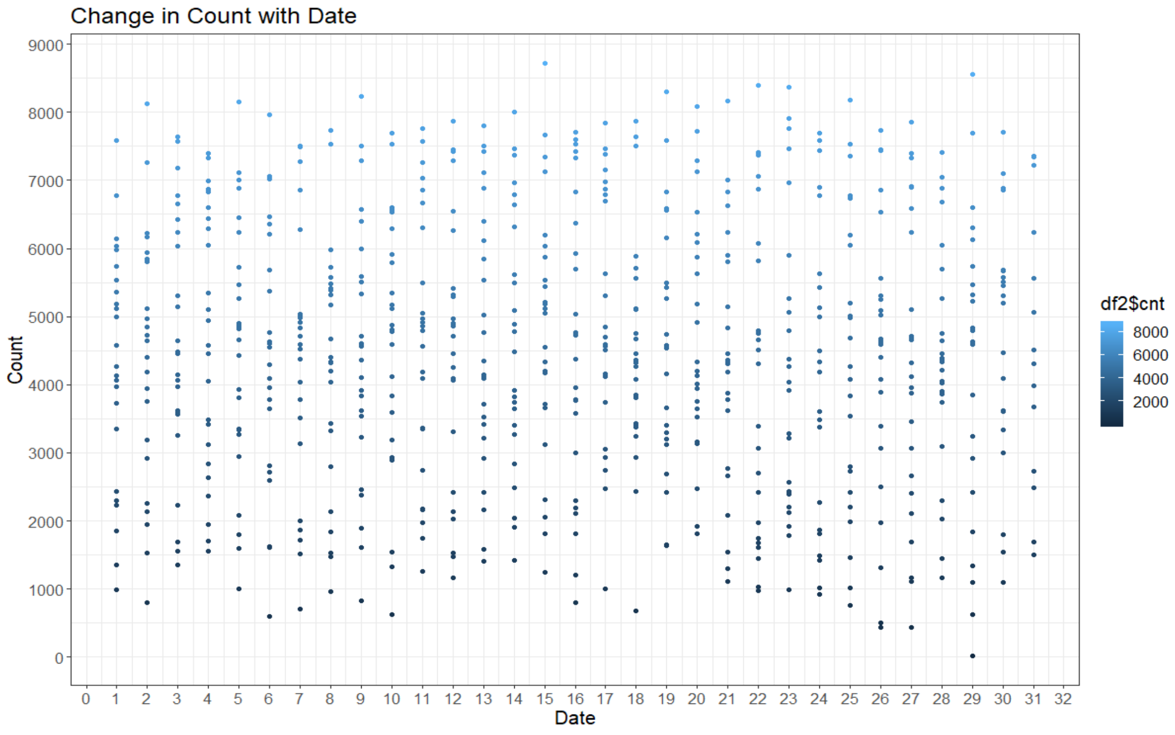
2.4 Effect of year and month on rental count


Change in Count with Year

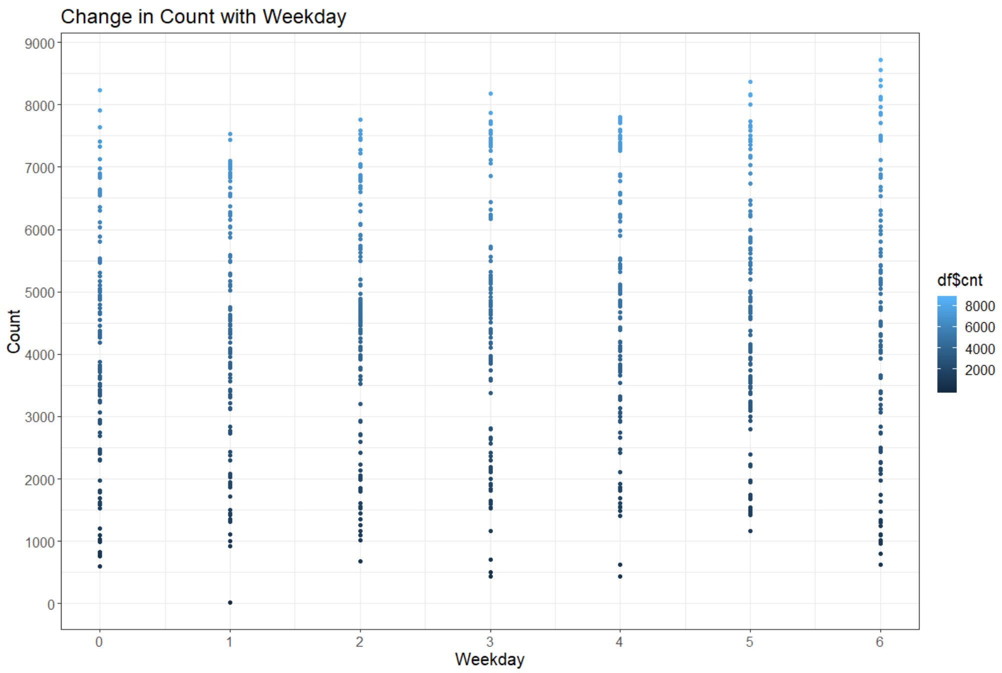

Change in Count with Month

- The rental count is higher in the year 2012 and is also quite high in months May to September.

2.5 Effect of date and weekday on rental count

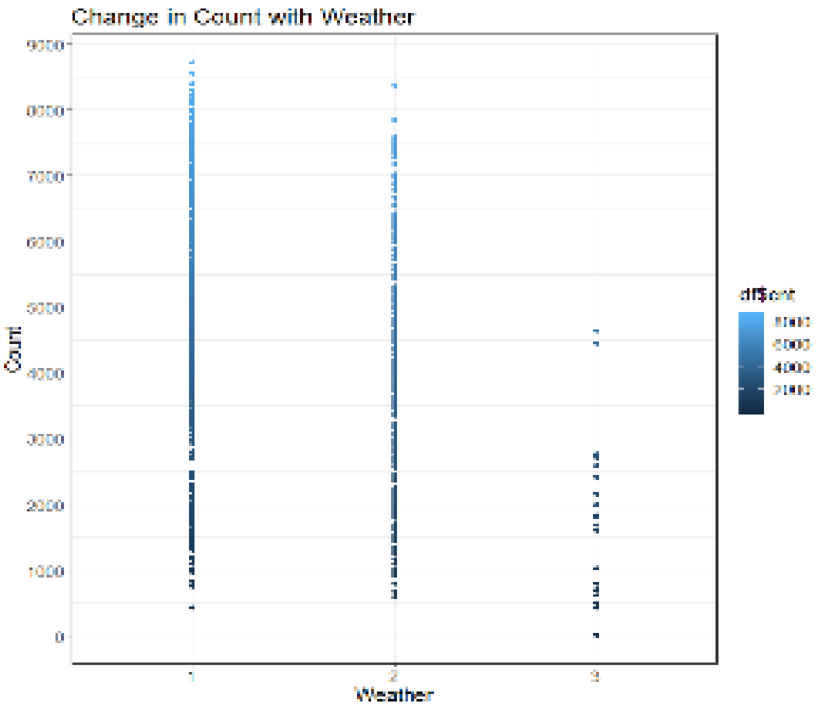

Change in Count with Date

- There is not much datewise variation in the count variable.



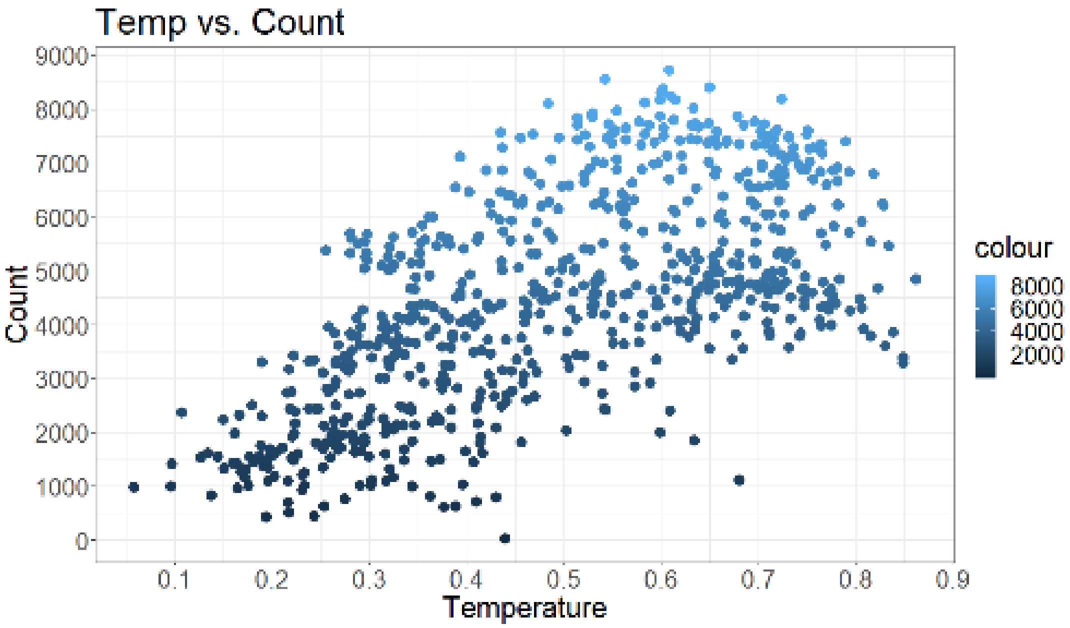
- The rental counts don’t have sharp variations throughout the week. Although the count is high on Sunday (6).

2.6 Effect of weather on rental count

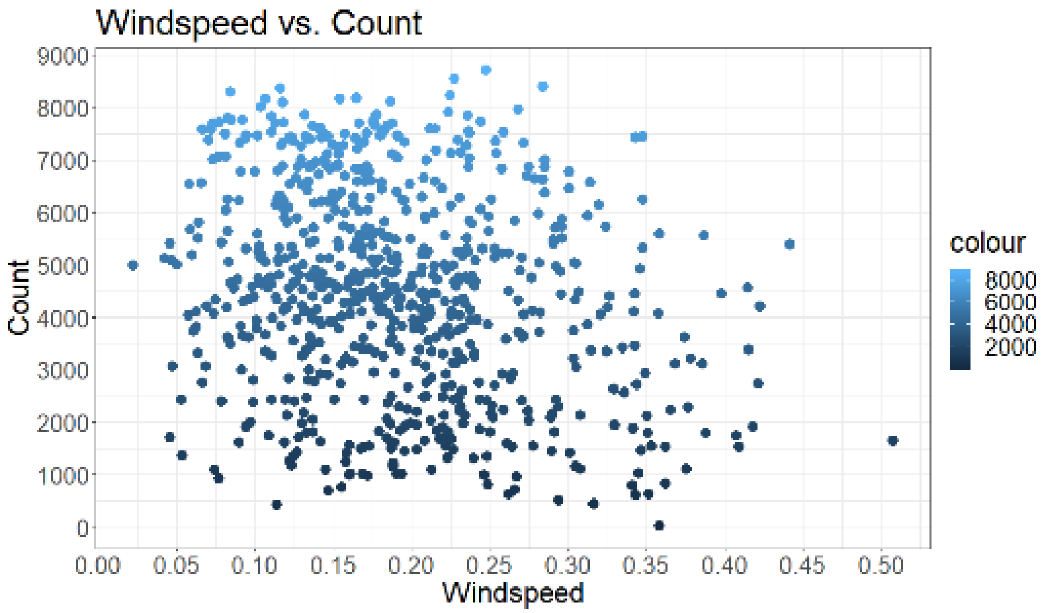


- Here, in the plot of weather vs. count, we can observe that when weather is clear, the count is maximum and it decreases as the weather worsens.

2.7 Effect of temperature and windspeed on rental count

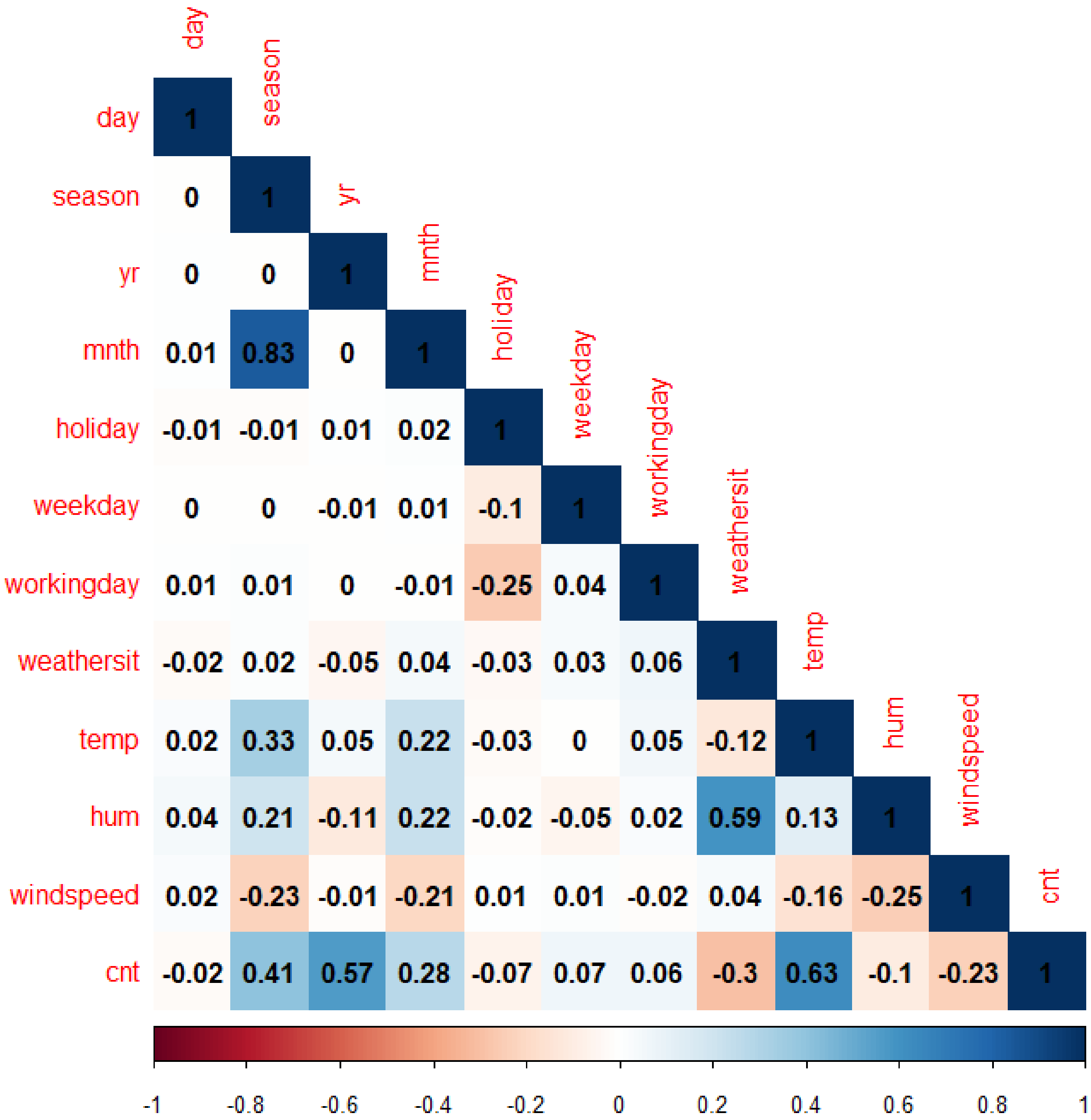


- Temperature and count seem to be linearly related. This means that generally, as the temperature increases, the rental count also increases.



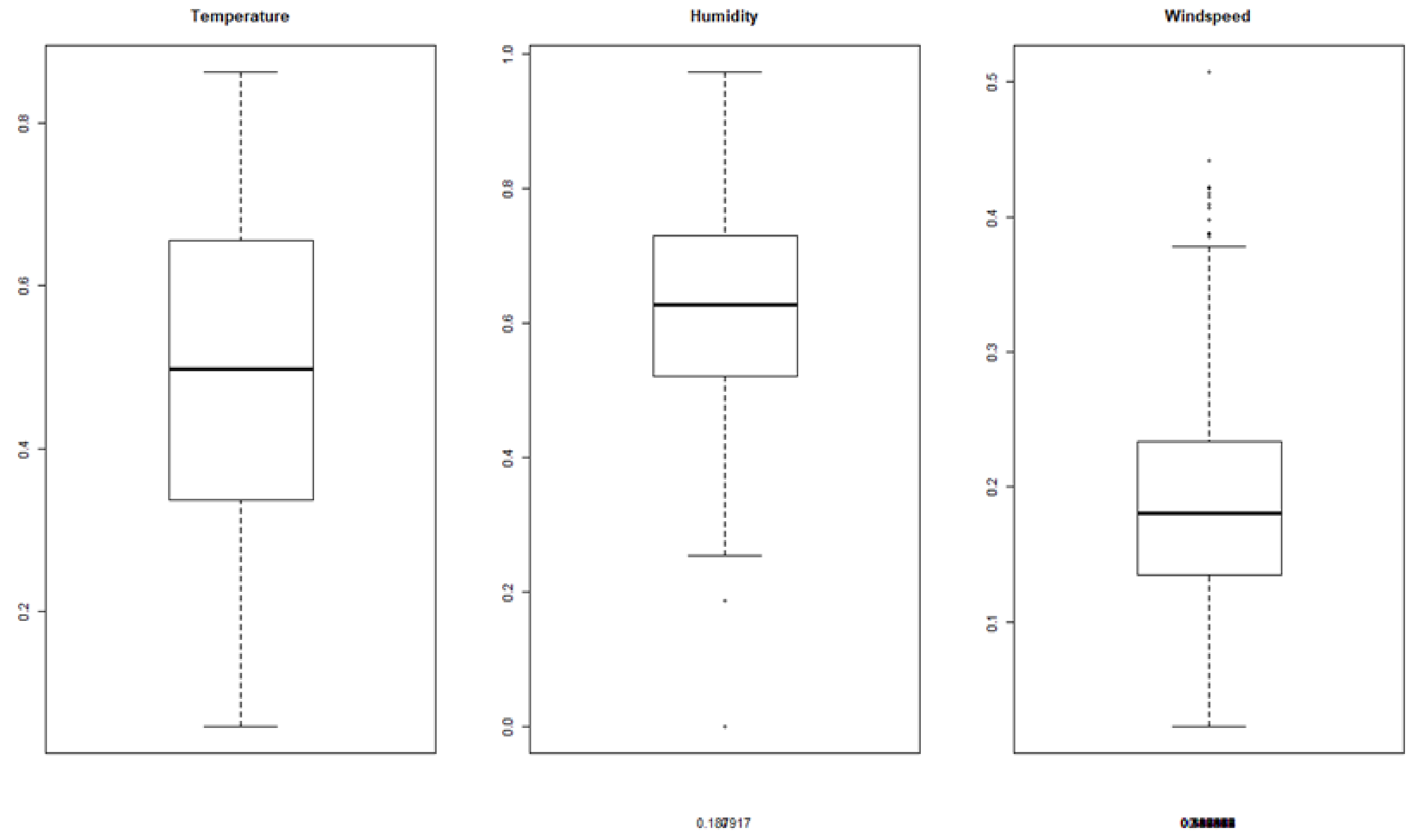
- There isn't any strong relation but we can see that there is a fairly inverse linear relation. As windspeed increases, the count decreases.

2.8 Correlation plot



- Now, since this is a regression problem and we have categorical variables like season and weathersit with multiple categories, there is a chance that our algorithm may take these categories as numeric values. To avoid this, we create dummy variables for each category.

2.9 Outlier Analysis



- Then we perform some outlier analysis and it is seen that ‘windspeed’ and ‘hum’ variables have outliers. Thus, we replace the outlier values with NAs and then impute them with Random Forest imputer in ‘missForest’ library.

Chapter 3

Modelling and Rental Count Predictions

- First, we create a function 'errors' to calculate the error metrics.
- Then, we split our train data into train (80%) and validation data (20%).
- Now, we move on to modelling phase. Note that basic multiple linear regression, lasso and ridge were also implemented, but these were predicting negative count values. So, these models were not considered for further analysis.

  Note: Even though many error metrics were calculated, the focus was to minimize the RMSLE value. We have chosen RMSLE for this project because the count values are in the range of thousands and thus other error metrics will be either in the range of hundreds or thousands. If we use log-based error metric like RMSLE, we can get a better scale to interpret the performance of our model.

3.1 Implementation of models

i. **Random Forest Regression: -**
   This is ensemble-based regression model in which several decision trees are generated where each of them has equal vote to decide what the prediction value of the target variable should be.
   Here, Random Forest gave RMSLE of 0.2197 for train data and 0.2639 for validation data.

ii. **AdaBoost Regression: -**
   Similar to random forest but generates several stumps rather than trees and feeds the error from previous stump to the next stump. Thus, as consequent stumps are influenced by the errors of previous stumps, all stumps will not get equal vote on the final value of the prediction.
   Here, AdaBoost gave RMSLE of 0.1362 on train data and 0.2326 on validation data.

iii. **KNN Regression: -**
   This algorithm predicts the target variable based on the patterns in the training data and decides the prediction value based on the K – nearest neighbors.
   KNN gave RMSLE of 0.2961 on train data and of 0.344 on validation data.

iv. **Light Gradient Boost Regression: -**
   LGBM has trees of fixed length and are shorter than the trees in decision tree and random forest. It is also based on ensemble method like random forest, but the difference is that all the trees in LGBM do not have equal say on the output. The

say (or vote) of each tree depends upon the error of previous tree and how that current tree reduces the error.

In the model, most of the default parameters were set and we implemented our model.

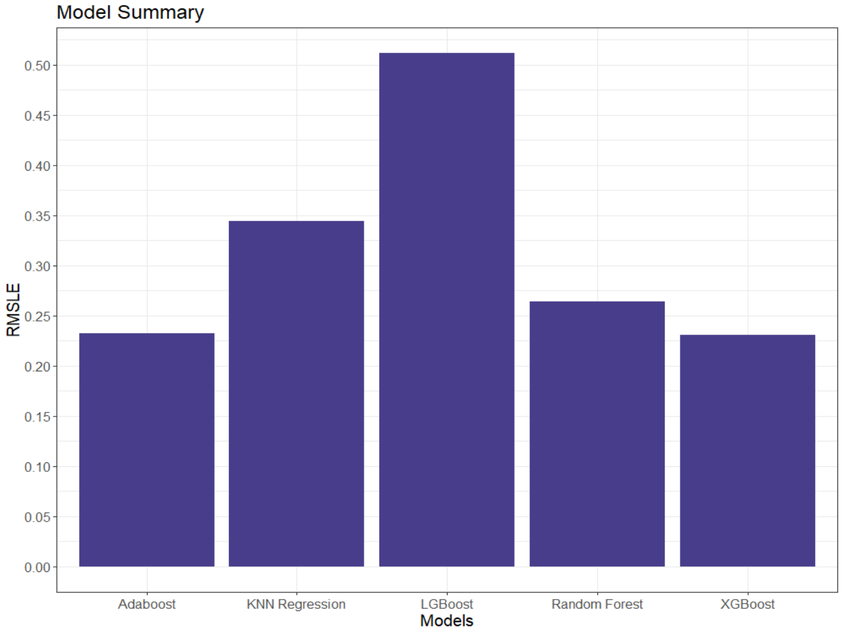
Here, LGBM gave RMSLE of 0.482 on training data and of 0.5115 on validation data.

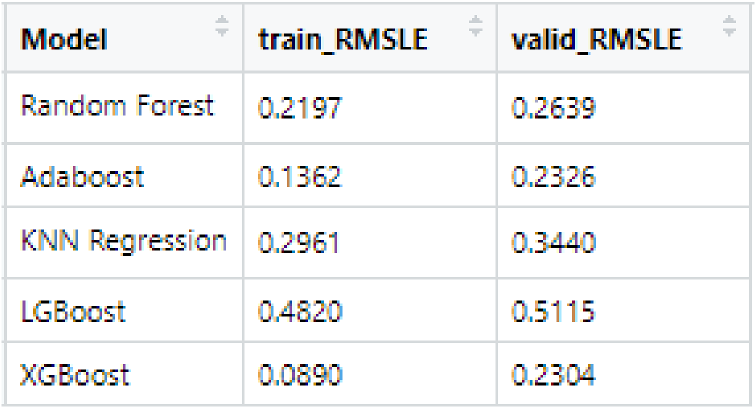
v. **Xtreme Gradient Boost Regression: -**

This model is similar to LGBM. In fact, LGBM improves upon XGBoost.

Here, XGBoost gave RMSLE of 0.089 on training data and 0.2304 on validation data.

3.2 Performance Summary of models





Model	train_RMSLE	valid_RMSLE
Random Forest	0.2197	0.2639
Adaboost	0.1362	0.2326
KNN Regression	0.2961	0.3440
LGBoost	0.4820	0.5115
XGBoost	0.0890	0.2304

3.3 Hyperparameter Tuning

- Out of all the models we have implemented, Adaboost and XGBoost performed the best with RMSLE of 0.2326 and 0.2304 respectively for validation data with variance of 0.044 and 0.141. Since XGBoost is overfitting to the train data despite tuning manually, we consider Adaboost and perform hyperparameter tuning on it to see whether the results improve.
- After hyperparameter tuning, we obtained RMSLE of 0.205 on train data and 0.2139 on validation data.
- The optimal parameters were found to be:

```
gbm (cnt~.,data=train,
        distribution = "gaussian",
        interaction.depth=3,
        bag.fraction=0.9,
        n.trees = ntree)
```

- Thus, we use adaboost regressor to predict our sample test dateset.

3.4 Prediction on Sample Dataset

- We create a sample dataset with all the variables that are present in a validation data (without target variable) as there was no test data that was provided separately. We then use our trained adaboost model to predict the rental count value for each observation.
- The file was saved as ‘Final_Predictions_R.csv’ and is included in the folder.

Chapter 4

Conclusion

- The aim of this project was to predict the bike rental count value based on various factors such as season, weather, temperature, windspeed, holiday/ working day etc.
- For this, we first gained insights from the data about the features and how they were related to our target variable ‘count’.
- From above EDA, we gained following key insights: -
  - More registered users rent bikes as compared to casual users.
  - The rental count is more on working days rather than on holidays.
  - The rental count was high in the year 2012.
  - The count was high from May to September.
  - As weather gets bad, the rental count decreases.
  - As temperature goes down, the rental count decreases.
- Then we trained several models and out of them, Adaboost proved to be the best for our case as it gave least RMSLE of 0.2139 on validation data after hyperparameter tuning.
- A sample dataset with random observations was created and the count value was predicted for each case using our trained Adaboost model.