Bike Rental Prediction

Shrunket Sanjeev Bahadure

23rd December, 2019

Contents

1 Introduction

2 Methodology

3 Conclusion

Chapter 1

Introduction

1.1 Problem Statement

The objective of this case is to predict daily bike rental count based on the environmental and seasonal settings.

1.2 Data

The task of this project is to build a regression model to predict daily bike rental count from the provided dataset. Below is the sample data that we are using for the prediction.

Table 1.1 Bike Rental Sample Data (Column 1-7) (Rows 1-8)

instant	dteday	season	yr	mnth	holiday	weekday	workingday
1	01-01-2011	1	0	1	0	6	0
2	02-01-2011	1	0	1	0	0	0
3	03-01-2011	1	0	1	0	1	1
4	04-01-2011	1	0	1	0	2	1
5	05-01-2011	1	0	1	0	3	1
6	06-01-2011	1	0	1	0	4	1
7	07-01-2011	1	0	1	0	5	1

Table 1.1 Bike Rental Sample Data (Column 1-7) (Rows 9-16)

weathersit	temp	atemp	hum	windspeed	casual	registered	cnt
2	0.344167	0.363625	0.805833	0.160446	331	654	985
2	0.363478	0.353739	0.696087	0.248539	131	670	801
1	0.196364	0.189405	0.437273	0.248309	120	1229	1349
1	0.2	0.212122	0.590435	0.160296	108	1454	1562
1	0.226957	0.22927	0.436957	0.1869	82	1518	1600
1	0.204348	0.233209	0.518261	0.089565	88	1518	1606
2	0.196522	0.208839	0.498696	0.168726	148	1362	1510

From the above data it is clear that we have 16 attributes as follows:

Number of attributes:

instant: Record index

dteday: Date

season: Season (1:springer, 2:summer, 3:fall, 4:winter)

yr: Year (0: 2011, 1:2012)

mnth: Month (1 to 12)

hr: Hour (0 to 23)

holiday: weather day is holiday or not (extracted from Holiday Schedule)

weekday: Day of the week

workingday: If day is neither weekend nor holiday is 1, otherwise is 0.

weathersit: (extracted from Freemeteo) 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: Normalized temperature in Celsius. The values are derived via (t-t_min)/(t_max-t_min), t_min=-8, t_max=+39 (only in hourly scale)

atemp: Normalized feeling temperature in Celsius. The values are derived via (t-t_min)/(t_maxt_min), t_min=-16, t_max=+50 (only in hourly scale)

hum: Normalized humidity. The values are divided to 100 (max)

windspeed: Normalized wind speed. The values are divided to 67 (max)

casual: count of casual users.

registered: count of registered users

cnt: count of total rental bikes including both casual and registered

We will use the above attributes to predict the target variable ***cnt***

1.3 Hypothesis Generation

In this section we will write the entire possible hypothesis which will affect our target variable ***cnt***.

1. Rental count will be more on weekdays as compared to weekends.
2. Rental count will be more on peak hours as compared to non-peak hours.
3. Rental count may also depend on day of the month.
4. Rental count depends on season and weather situations.
5. Rental count depends on temperature (rental count will decrease at low and high temperature, while rental count will increase at normal temperature)
6. Rental count will decrease due to high humidity and increase with low humidity.
7. Rental count will be less for high wind speed and more for low wind speed.

Chapter 2

Methodology

2.1 Pre Processing

In this section we will explore and understand the available data. And we will process the raw data before designing an optimum model for the given problem.

2.2.1 Exploratory Analysis

First we will explore the data distribution of each variable from the given data to get valuable insights and underlying patterns such as skewness, frequencies, center and outliers. Most analysis like regression, require the data to be normally distributed.

We will also analyse the summary of all the variables. This will help us to design an approach for the model development.

Now we will visualize behaviour of independent variables with the target variable. And we will visualize data distribution.

Fig 1.1

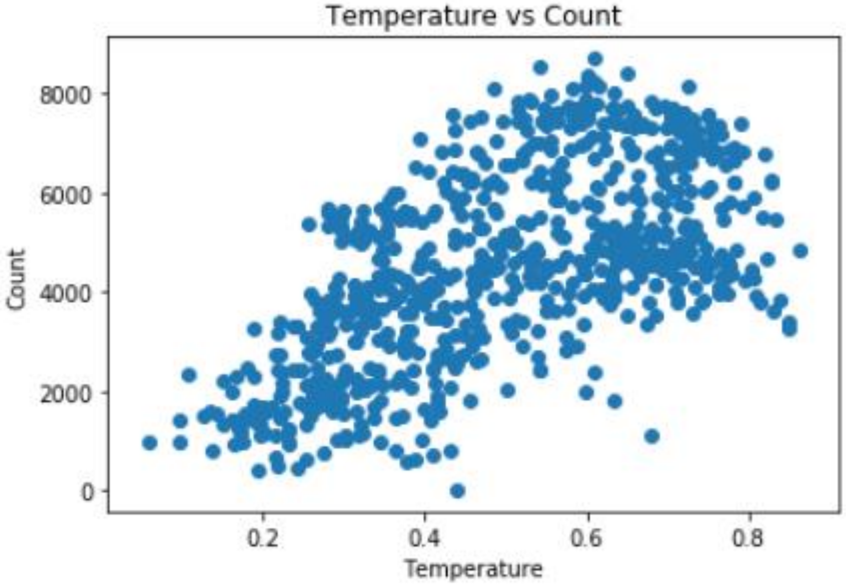


Fig 1.1 indicates that as rental count increases so does the temperature increases upto certain point and after that we observe a reverse trend.

Fig 1.2

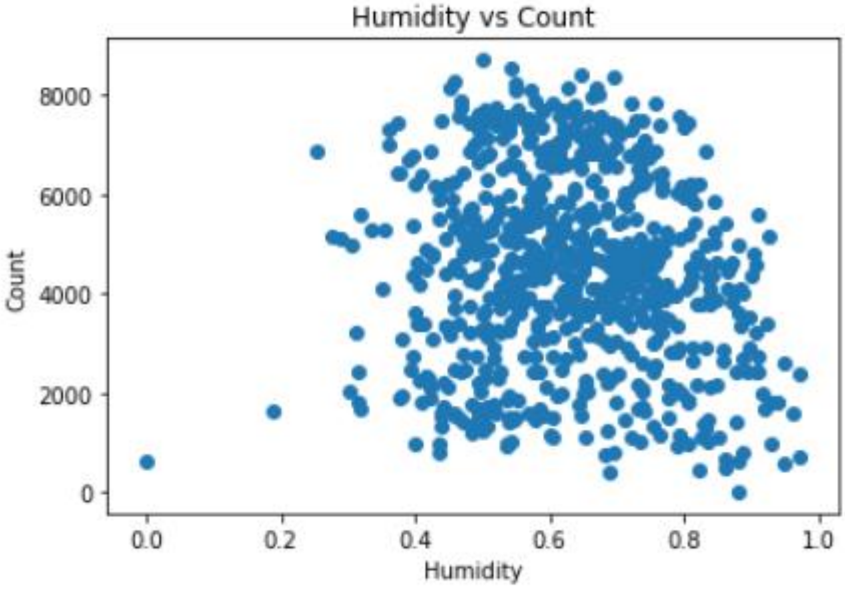


Fig 1.2 indicates that as the humidity increases, rental count decreases and vice versa.

Fig 1.3

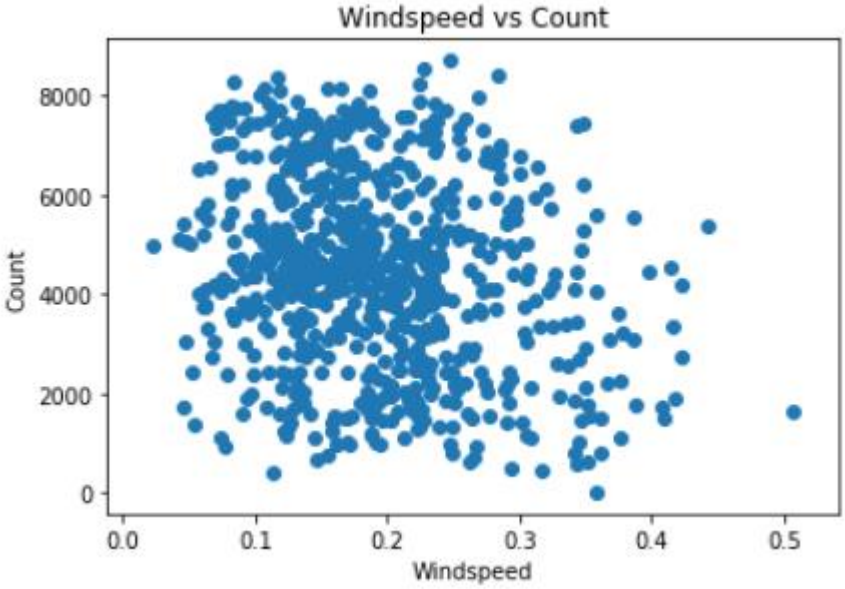


Fig 1.3 indicates that as the wind speed increases, rental count decreases and vice versa.

Fig 1.4

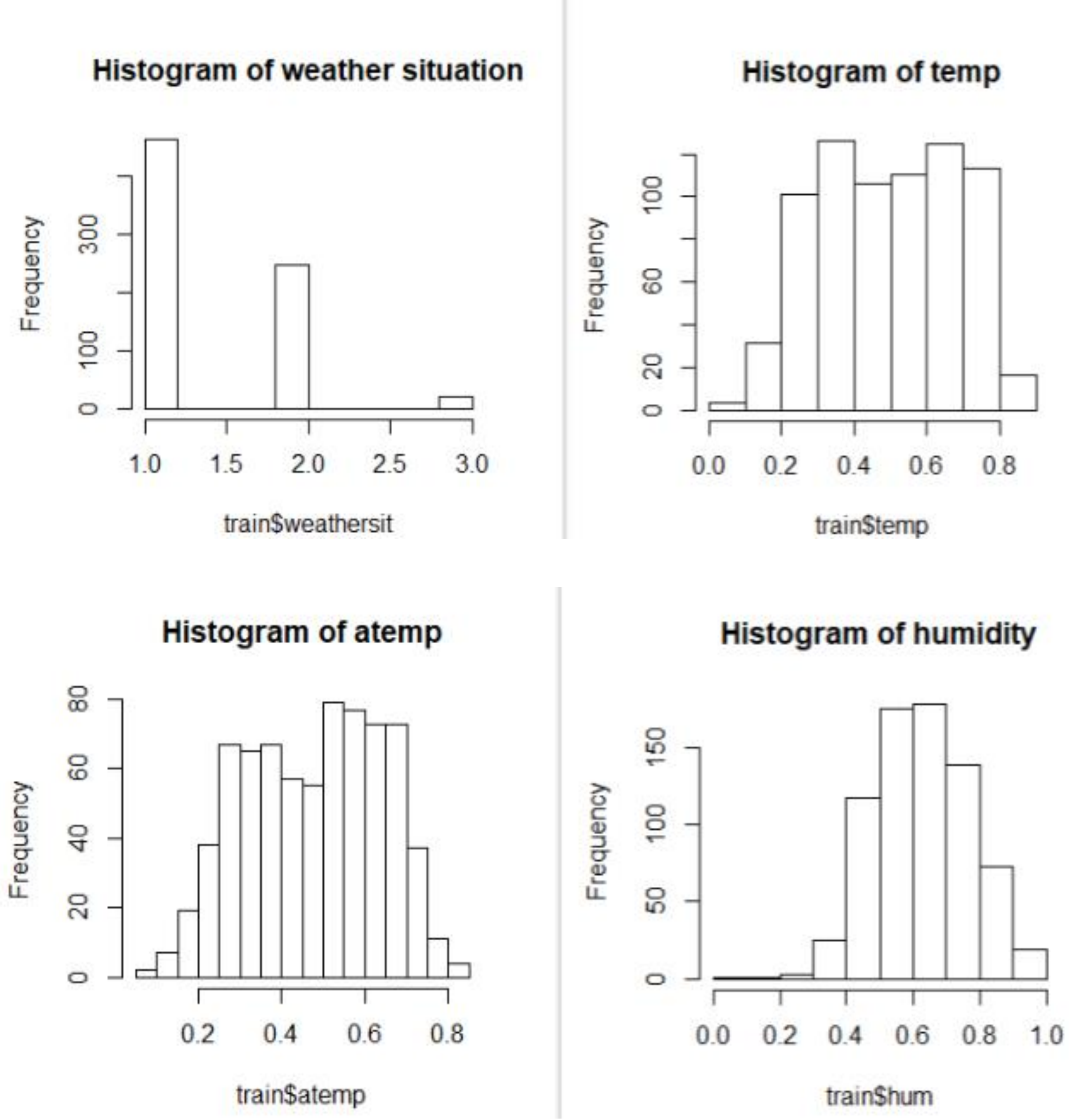
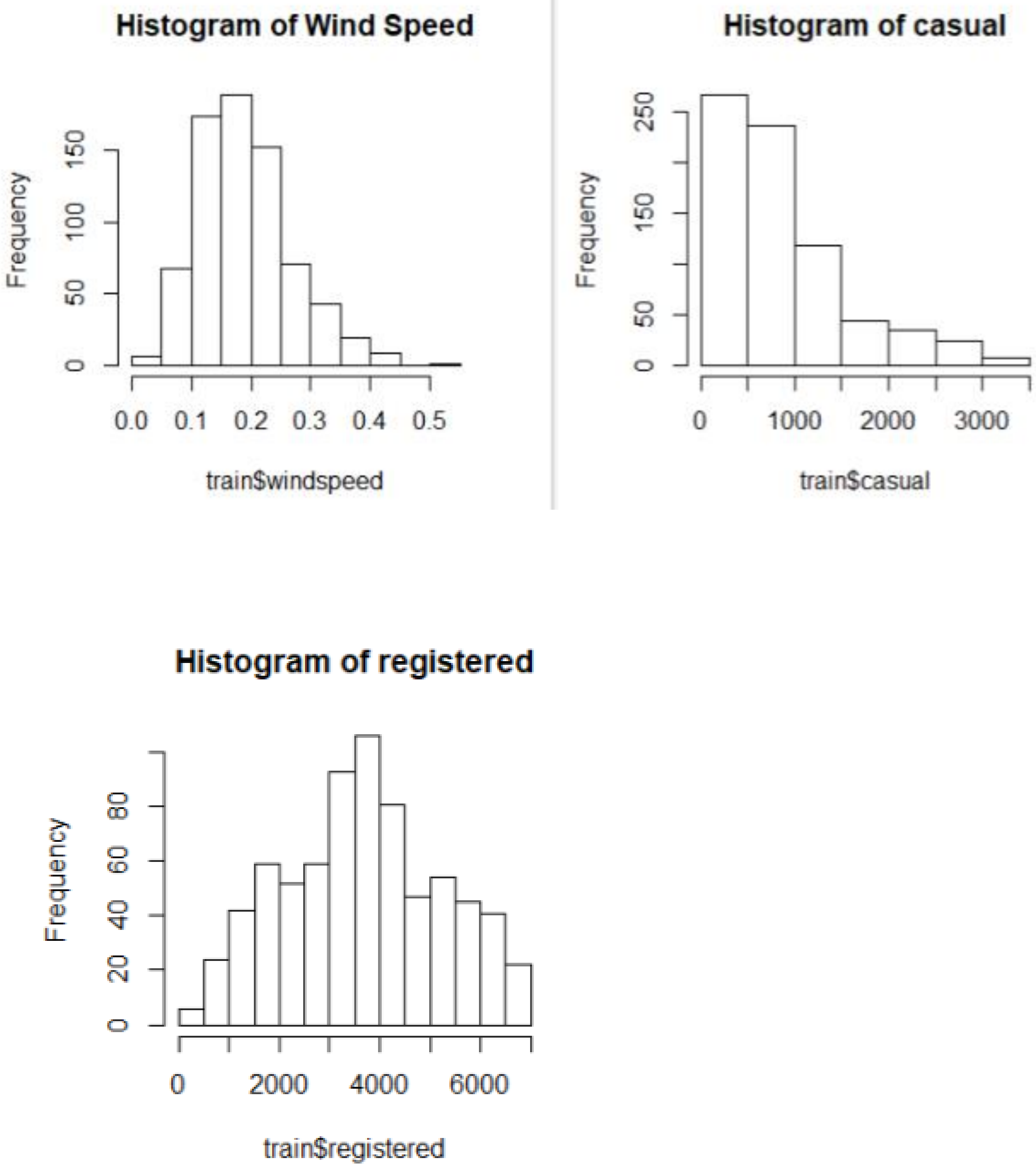

Histogram of weather situation
Frequency
0
100
300
1.0
1.5
2.0
2.5
3.0
train$weathersit
Histogram of temp
Frequency
0
20
60
100
0.0
0.2
0.4
0.6
0.8
train$temp
Histogram of atemp
Frequency
0
20
40
60
80
0.2
0.4
0.6
0.8
train$atemp
Histogram of humidity
Frequency
0
50
100
150
0.0
0.2
0.4
0.6
0.8
1.0
train$hum

Fig 1.5



Histogram of Wind Speed

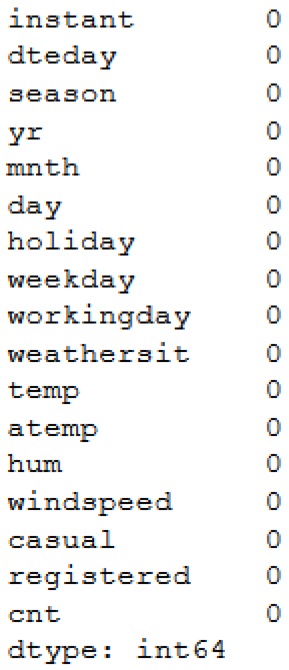
Histogram of casual

Histogram of registered

2.2.2 Missing Value Analysis

Missing values in the dataset are caused due to insufficient data or human error. Missing values in the dataset gives us misleading results. Therefore it is very important to treat these missing values.

In this section we will look for missing values in each variable. Let's check total number of missing values for each variable in the table below:



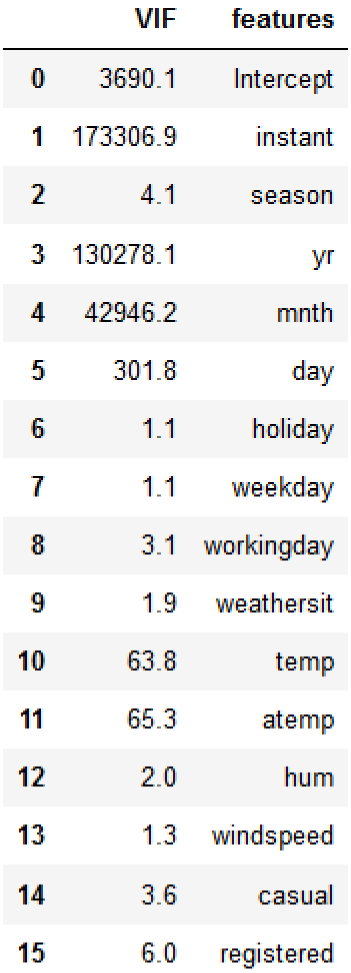
```
instant       0
dteday        0
season        0
yr            0
mnth          0
day           0
holiday       0
weekday       0
workingday    0
weathersit    0
temp          0
atemp         0
hum           0
windspeed     0
casual        0
registered    0
cnt           0
dtype: int64
```

In the table above it is clear that we don't have any missing values in the dataset.

2.2.5 Feature Selection

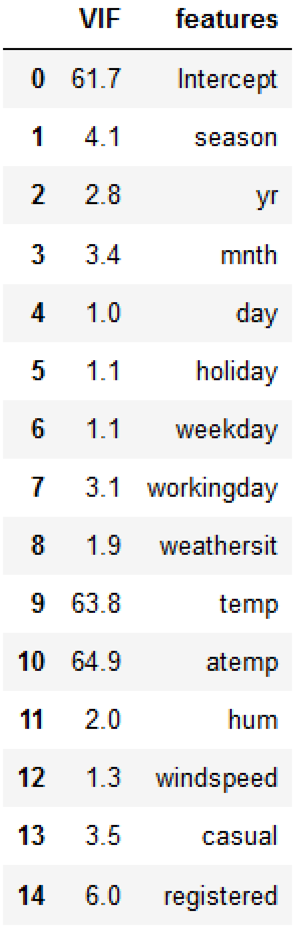
Collinear variables don’t add any significant information in describing the target variable. In return it increases the complexity of the model. Hence it is important to remove highly collinear variables from the dataset before training the model.

To check co-linearity between the variables we will calculate Variance Inflation Factor (VIF) for each variable and we will check how a single variable contributes collinearity to the data as a whole. Threshold value of VIF is 5. We will select variables with VIF value below 5.



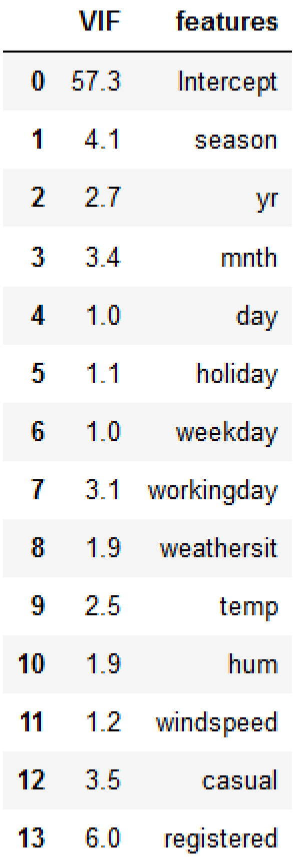
	VIF	features
0	3690.1	Intercept
1	173306.9	instant
2	4.1	season
3	130278.1	yr
4	42946.2	mnth
5	301.8	day
6	1.1	holiday
7	1.1	weekday
8	3.1	workingday
9	1.9	weathersit
10	63.8	temp
11	65.3	atemp
12	2.0	hum
13	1.3	windspeed
14	3.6	casual
15	6.0	registered

As *instant* variable’s VIF value is the highest among other variables. Therefore we will remove this variable first. We will again calculate the VIF values of the remaining variable.



	VIF	features
0	61.7	Intercept
1	4.1	season
2	2.8	yr
3	3.4	mnth
4	1.0	day
5	1.1	holiday
6	1.1	weekday
7	3.1	workingday
8	1.9	weathersit
9	63.8	temp
10	64.9	atemp
11	2.0	hum
12	1.3	windspeed
13	3.5	casual
14	6.0	registered

atemp variable is with the highest value of VIF. We will remove this variable from the dataset. We will again check the VIF of remaining variables.



	VIF	features
0	57.3	Intercept
1	4.1	season
2	2.7	yr
3	3.4	mnth
4	1.0	day
5	1.1	holiday
6	1.0	weekday
7	3.1	workingday
8	1.9	weathersit
9	2.5	temp
10	1.9	hum
11	1.2	windspeed
12	3.5	casual
13	6.0	registered

casual and *registered* variables are the subsets of target variable *cnt*. These variables describe too much about target variables so we will remove these variables too. So we have selected variables with VIF value less than the threshold value ie 5 for further model development.

2.2.4 Feature Engineering

The purpose of feature engineering is to derive more variables from the existing variables which will be more helpful in describing the target variable and eventually improve the performance of the machine learning model.

As per our hypothesis, we can derive days from the date variable (as we already have month and year variable). This derived variable adds significant value in describing our target variable. After this we will remove date variable from our dataset.

Transformed dataset is as shown below:

Table 1.2 Bike Rental Sample Data (Column 1-8) (Rows 1-5)

instant	day	season	yr	mnth	holiday	weekday	workingday
1	1	1	0	1	0	6	0
2	2	1	0	1	0	0	0
3	3	1	0	1	0	1	1
4	4	1	0	1	0	2	1
5	5	1	0	1	0	3	1

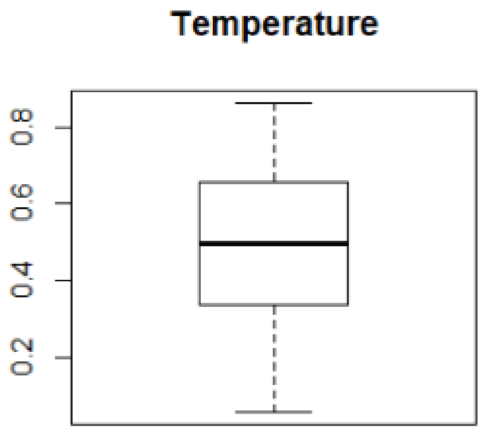
Table 1.2 Bike Rental Sample Data (Column 9-16) (Rows 1-5)

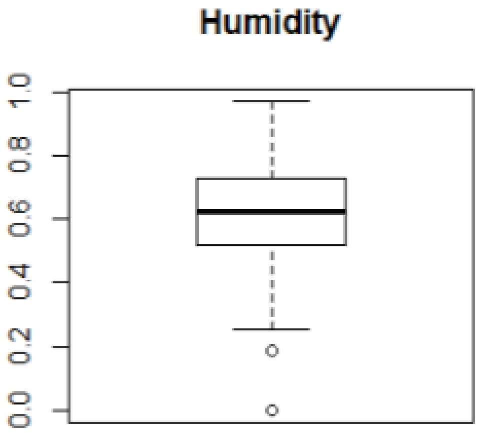
weathersit	temp	atemp	hum	windspeed	casual	registered	cnt
2	0.344167	0.363625	0.805833	0.160446	331	654	985
2	0.363478	0.353739	0.696087	0.248539	131	670	801
1	0.196364	0.189405	0.437273	0.248309	120	1229	1349
1	0.2	0.212122	0.590435	0.160296	108	1454	1562
1	0.226957	0.22927	0.436957	0.1869	82	1518	1600

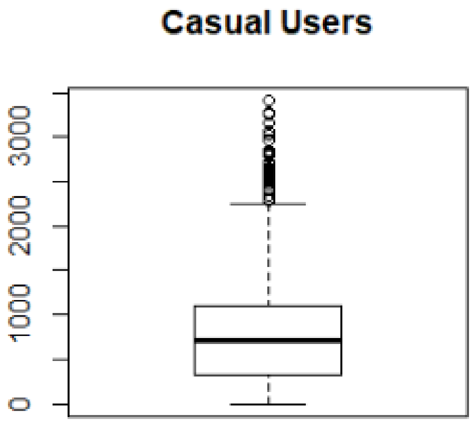
2.2.3 Outlier Analysis

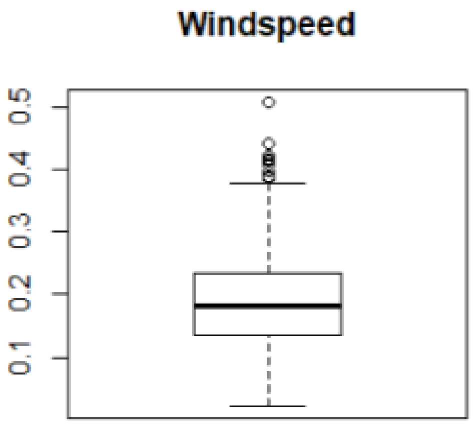
The outlier in an observation is an observation point that is distant from other observations. Outliers can either be a mistake or just variance. This can significantly impact our model performance. Therefore it is important to treat the outliers either by removing it or using imputation.

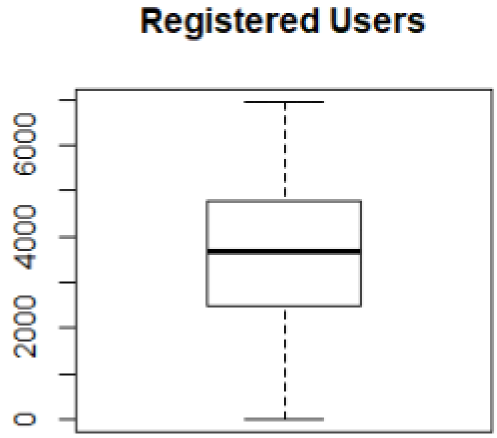
We have decided to drop the outlier observations from the dataset. To identify outliers we will use boxplot method.











Number of Outliers in *hum* variable: 2
Number of Outliers in *windspeed* variable: 12
Total number of outliers removed:14

2.2 Model Development

2.2.1 K-Fold Cross Validation

Before selecting an algorithm for development, we must test the performance of each of the algorithm used for regression. We will start testing the performance from simple algorithm to the complex one.

To do this we will use K-Fold Cross Validation technique. So we will randomly split the training data into K folds with each observation has equal chance of getting selected. Then fit the model using the K — 1 (K minus 1) folds and validate the model using the remaining Kth fold. Repeat the process K times. We will apply all this steps for each algorithm and get the error score. Based on better error score we will choose the algorithm for further development.

Linear Regression

We are selecting K=3 for each algorithm and record there error scores.

```
LR= train(cnt~.,data=train , method='lm',trControl= trainControl(method= 'cv', number = 3) )
LR
```

Result

Linear Regression: RMSE: 876.639 Rsquared:0.794 MAE: 657.096

KNN

```
KNN= train(cnt~.,data=train , method='knn',trControl= trainControl(method= 'cv', number = 3) )
KNN
```

Result

KNN Algorithm : RMSE: 1549.256 Rsquared:0.369 MAE: 1315.999

Decision tree

```
DT= train(cnt~.,data=train , method='rpart',trControl=
trainControl(method= 'cv', number = 3) )
DT
```

Result

Decision Tree: RMSE: 1104.027 Rsquared:0.675 MAE: 860.168

Random Forest

```
RF= train(cnt~.,data=train , method='rf',trControl=
trainControl(method= 'cv', number = 3) )
RF
```

Result

Random Forest: RMSE: 649.3443 Rsquared:0.8887 MAE: 450.7936

2.2.2 Model Selection

Evaluation Summary:

Linear Regression: RMSE: 876.639 Rsquared:0.794 MAE: 657.096

Decision Tree: RMSE: 1104.027 Rsquared:0.675 MAE: 860.168

KNN Algorithm : RMSE: 1549.256 Rsquared:0.369 MAE: 1315.999

Random Forest: RMSE: 649.3443 Rsquared:0.8887 MAE: 450.7936

We have implemented K-Fold Cross Validation for each algorithm. From the evaluation summary it is clear that Random Forest is better than other algorithms for our dataset with lowest RMSE of 649.344 and highest Rsquared value of 0.8887
Therefore we will use Random Forest for model development.

2.3 Modelling

2.3.1 Hyper-parameter tuning

To improve the model performance we will perform parameter tuning for *mtry* and *ntree* parameters in Random forest algorithm.

Tuning mtry :

After trying value of mtry=(1,2,3,4,5,6,7,8,9,10), we got a better performance for mtry=6

Tuning ntree :

After trying value of ntree=(500,600,700,800,900, 1000, 1500,2000,2500,3000,3500,4000,4500,5000), we got a better performance for For ntree= 1500, MAE= 438.341, RMSE= 622.9952, R-squared= 0.8653500.

Final performance of the Random Forest: MAE= 438.341, RMSE= 622.9952, R-squared= 0.8653

2.3.2 Random Forest

```
set.seed(123)

train.index = createDataPartition(train$cnt, p = .80, list = FALSE)

train_1 = train[ train.index,]

test_1  = train[-train.index,]



RF_model= randomForest(cnt~., train_1, mtry=6, ntree=1500,
importance= TRUE)

RF_predict= predict(RF_model,test_1[,-12])
```

Chapter 3

Conclusion

3.1 Model Evaluation

In this section we will take a look at the error metrics of the model that we developed before and we will understand why it has been used. We will split same dataset (training-80% & testing- 20%) for fitting and predicting the model to measure error metrics.

3.1.1 R-squared (R2)

R-squared value represents how the independent variables describe the target variable. The perfect R-squared value is 1. Closer to the ideal better is the model. We can calculate using formula:

$$R^2 = 1 - \frac{\sum(y_i - \hat{y})^2}{\sum(y_i - \bar{y})^2}$$

Where,

$\hat{y}$ $- predicted\ value\ of\ y$

$\bar{y} - mean\ value\ of\ y$

By using same data for training and testing we achieved R2= 0.912 For our Random Forest model.

To know how close the data are to the fitted regression line we choose R-squared as a statistical measure.

3.1.2 Mean Absolute Error (MAE)

The **Mean Absolute Error** (MAE) is the average of all absolute errors. We can calculate using formula:

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n}|x_i - x|$$

Where, xi = actual values
x = predicted values

For our model we achieved MAE = 420.1074

3.1.3 Mean Absolute Percentage Error (MAPE)

The mean absolute percentage error (MAPE) can be calculated as the average absolute percentage error for predicted values minus actual values divided by actual values. Where A_t is the actual value and F_t is the predicted value, this is given by:

$$\mathrm{M} = \frac{1}{n}\sum_{t=1}^{n}\left|\frac{A_t - F_t}{A_t}\right|$$

For our Random Forest model we got MAPE = 14.948%

3.1.4 Root Mean Squared Error (RMSE)

When we do a square root of summation of the square of distances from the points to the line, we get root mean squared error. The drawback of this error metric is that it is sensitive to outliers.
We can calculate using formula:

$$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y})^2}$$

For our Random Forest model we achieved RMSE= 590.763
As we have removed outliers from our data we can choose RMSE as an error metric for model evaluation.

Appendix A R Code

```
rm(list = ls())
getwd()
setwd('C:/Users/Samruddhi/Desktop/Edwisor Project 2')
train= read.csv('day.csv')   #importing dataset

# Libraries Loading
x =
c('tidyverse','lubridate','pedometrics','car','caret','randomForest','ggplot2',
'MLmetrics')
lapply(x, require, character.only = TRUE)
```

Exploratory Analysis

```
head(train)
str(train)
col_names=list(names(train))
for (i in col_names){
  print(summary(train[i]))
}

hist(train$weathersit, main= 'Histogram of wrking day')
hist(train$temp, main= 'Histogram of temp')
hist(train$atemp, main= 'Histogram of atemp')
hist(train$hum, main= 'Histogram of humidity')
hist(train$windspeed, main= 'Histogram of Wind Speed')
hist(train$casual, main= 'Histogram of casual')
hist(train$registered, main= 'Histogram of registered')
```

Feature engineering

```
train$dteday = format(as.Date(train$dteday, format='%d-%m-
%Y'),format= '%d') # extracting days from date column
names(train)[2] = 'day'
train$day= as.numeric(train$day)
head(train$day)
summary(train$day)
```

Missing Value Analysis

```
sum(is.na(train))       #No missing values present in our data set
```

Feature Selection

```
train_cat=train[c(2,3,4,5,6,7,8,9)]        # Subset of categorical variables
train_con= train[c(1,10,11,12,13,14,15,16)]  # Subset of continous
variables

# Variance Inflation Factor Analysis
fit= lm(cnt~., data= train)
vif(fit)                          # calculating VIF values of each variable with
threshold value of 5
train= train[-c(1)]             # removing 'instant' variable from the dataset
head(train)
train= train[-c(10)]            # removing 'atemp' variable
train= train[-c(12,13)]         # removing 'casual' and 'registered' variable
head(train)
```

Outlier Analysis

```
boxplot(train$registered, main='Registered Users')
num_col= c('temp','hum','windspeed') #continous variables

for(i in num_col){
  print(i)
  val = train[,i][train[,i] %in% boxplot.stats(train[,i])$out]
  print(length(val))
  train = train[which(!train[,i] %in% val),]
}
length(train$day)
```

Model Selection

```
set.seed(123)
#Using K-fold cross validation method for model selection
# Linear Regression
LR= train(cnt~.,data=train , method='lm',trControl=
trainControl(method= 'cv', number = 3) )
LR      # Linear Regression: RMSE: 876.639 Rsquared:0.794  MAE:
657.096

# KNN Algorithm
KNN= train(cnt~.,data=train , method='knn',trControl=
trainControl(method= 'cv', number = 3))
KNN     # KNN Algorithm : RMSE: 1549.256  Rsquared:0.369  MAE:
1315.999

# Decision Tree
DT= train(cnt~.,data=train , method='rpart',trControl=
trainControl(method= 'cv', number = 3))
DT      # Decision Tree: RMSE: 1104.027 Rsquared:0.675  MAE: 860.168

# Random Forest
RF= train(cnt~.,data=train , method='rf',trControl=
trainControl(method= 'cv', number = 3))
RF      # Random Forest: RMSE: 649.3443 Rsquared:0.8887  MAE:
450.7936  for mtry:6
```

Hyper-parameter tuning

```
# finding best possible mtry value
RF= train(cnt~.,data=train , method='rf',trControl=
trainControl(method= 'cv', number = 3))
RF
# optimum value of mtry= 6

# finding best possible ntree value
x= train[c(1,2,3,4,5,6,7,8,9,10,11)]  #independent variables
y= train[c(12)]                        # target variable
control = trainControl(method="cv", number=3, search="grid")
tunegrid = expand.grid(.mtry=6)
modellist = list()
```

```
for (ntree in c(500,600,700,800,900,1000, 1500, 2000,
2500,3000,3500,4000,4500,5000)) {
  set.seed(123)
  fit = train(cnt~., data=train, method="rf", tuneGrid=tunegrid,
trControl=control, ntree=ntree)
  key = toString(ntree)
  modellist[[key]] = fit
}
results = resamples(modellist)
summary(results)
# For ntree= 1500, MAE= 438.341, RMSE= 622.9952, R-squared=
0.8653
```

Model Building

```
set.seed(123)

train.index = createDataPartition(train$cnt, p = .80, list = FALSE)

train_1 = train[ train.index,]

test_1  = train[-train.index,]



RF_model= randomForest(cnt~., train_1, mtry=6, ntree=1500,
importance= TRUE)

RF_predict= predict(RF_model,test_1[,-12])
```

Model Evaluation

```
mae = MAE(RF_predict,test_1[,12])              # MAE = 420.1074

mape = MAPE(RF_predict,test_1[,12])*100       # MAPE = 14.948 %

rmse=RMSE(RF_predict,test_1[,12])              # RMSE = 590.763

r2= R2(RF_predict,test_1[,12])                # R-squared = 0.912
```

References

1. https://edwisor.com/
2. https://www.analyticsvidhya.com/
3. https://www.kaggle.com/
4. https://medium.com/
5. https://towardsdatascience.com/
6. https://en.wikipedia.org/