# Bike renting

Sankepally vikram reddy

12/7/2018

# Contents:

Chapter 5:

Machine learning models

- Linear Regression
- Random forest
- Elastic net
- Lasso regression
- Ridge regression

Chapter 6:

Model results

- Model results with feature selection in R and python
  - RMSE
  - R-squared
- Model results with PCA in R and python
  - RMSE
  - R-squared
- Comparing results of both the model
- Selecting the best fitted model

Chapter 7:

Code

- R
- Python

# Chapter 1:

# Introduction

### Problem statement

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

### Data

The details of data attributes in the dataset are as follows -
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 fromHoliday Schedule) weekday: Day of the week workingday: If day is neither weekend nor holiday is 1, otherwise is 0.
weathersit: (extracted fromFreemeteo)
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

# Chapter 2:

## Exploratory data analysis

### Recoding factor variables

In order to view the distribution of classes in factor variables
We have to recode them back to their original form

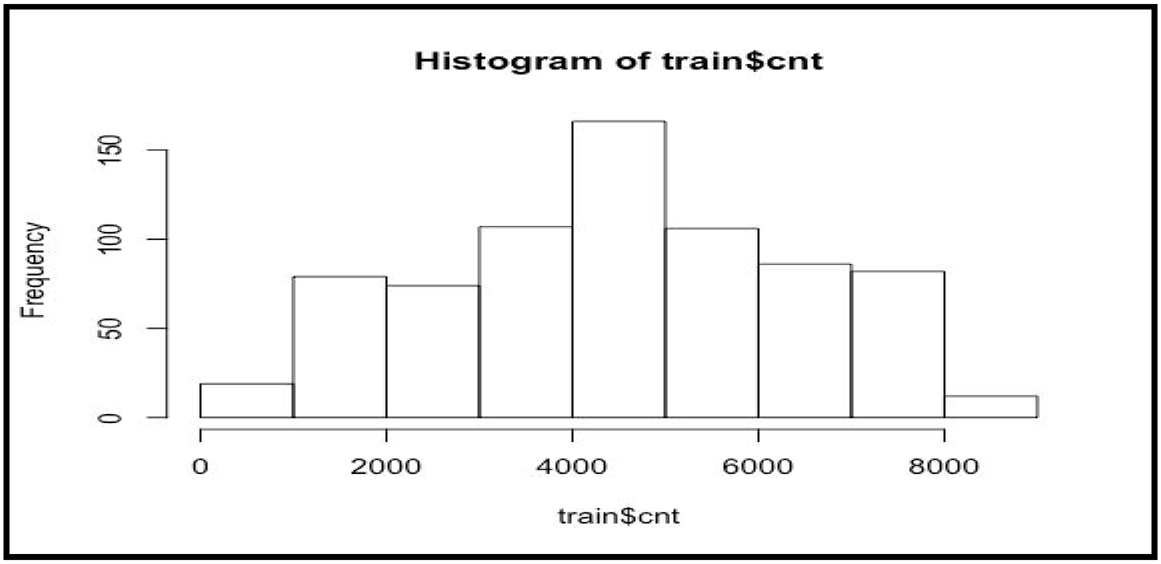
Features to be recoded

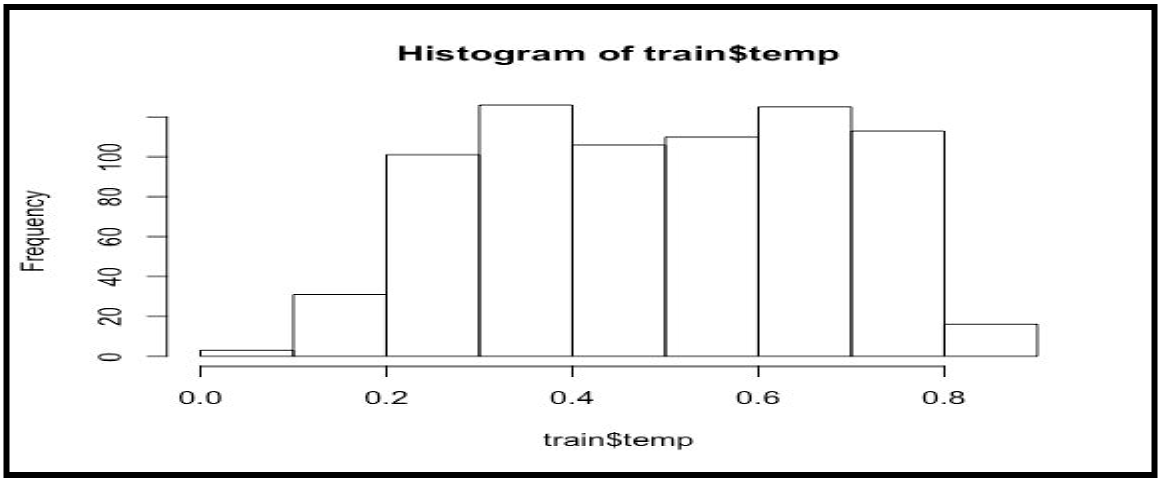
- Season
- Weekday
- Working day
- Yr
- Weathersit
- Holiday
- Month

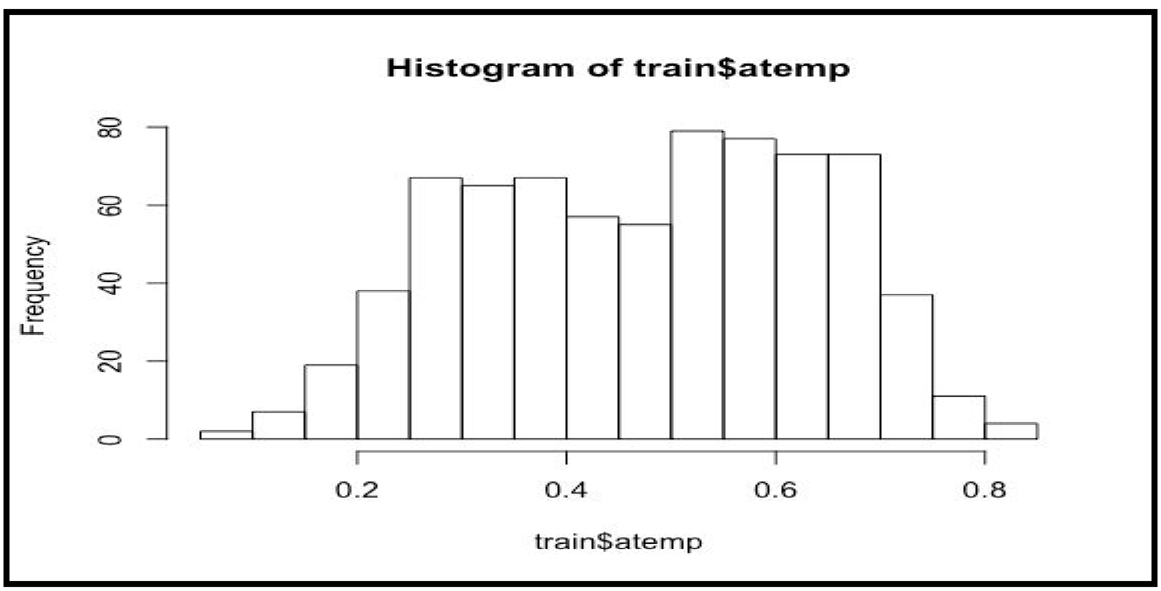
| | instant | dteday | season | yr | mnth | holiday | weekday | workingday | weathersit | temp |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 2011-01-01 | Spring | 2011 | jan | No | sat | No | Normal | 0.344167 |
| 2 | 2 | 2011-01-02 | Spring | 2011 | jan | No | sun | No | Normal | 0.363478 |
| 3 | 3 | 2011-01-03 | Spring | 2011 | jan | No | mom | Yes | Good | 0.196364 |
| 4 | 4 | 2011-01-04 | Spring | 2011 | jan | No | tue | Yes | Good | 0.200000 |
| 5 | 5 | 2011-01-05 | Spring | 2011 | jan | No | wed | Yes | Good | 0.226957 |
| 6 | 6 | 2011-01-06 | Spring | 2011 | jan | No | thu | Yes | Good | 0.204348 |

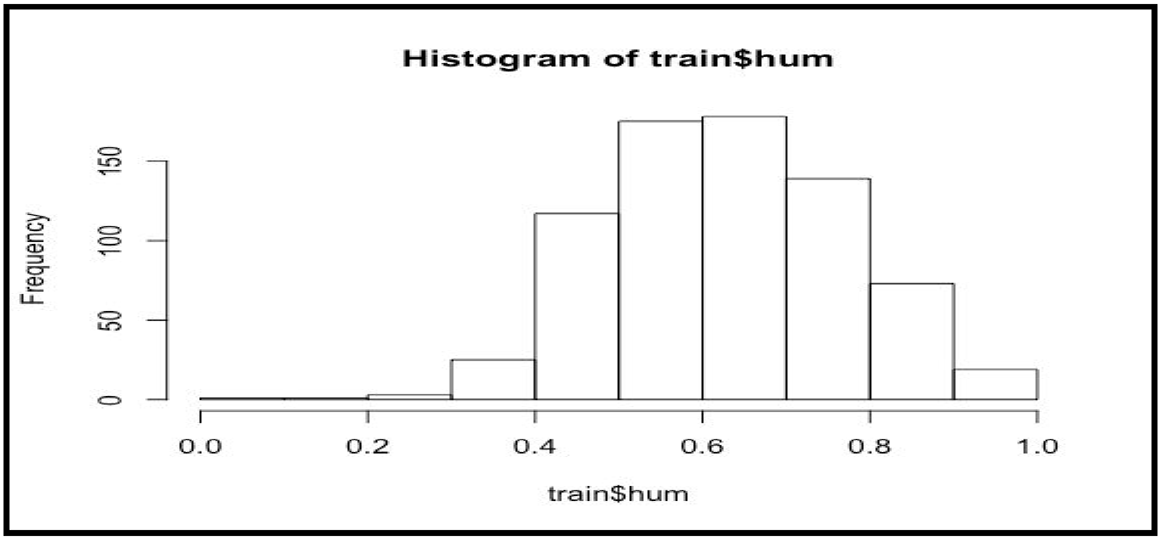
| | atemp | hum | windspeed | casual | registered | cnt |
|---|---|---|---|---|---|---|
| 1 | 0.363625 | 0.805833 | 0.1604460 | 331 | 654 | 985 |
| 2 | 0.353739 | 0.696087 | 0.2485390 | 131 | 670 | 801 |
| 3 | 0.189405 | 0.437273 | 0.2483090 | 120 | 1229 | 1349 |
| 4 | 0.212122 | 0.590435 | 0.1602960 | 108 | 1454 | 1562 |
| 5 | 0.229270 | 0.436957 | 0.1869000 | 82 | 1518 | 1600 |
| 6 | 0.233209 | 0.518261 | 0.0895652 | 88 | 1518 | 1606 |

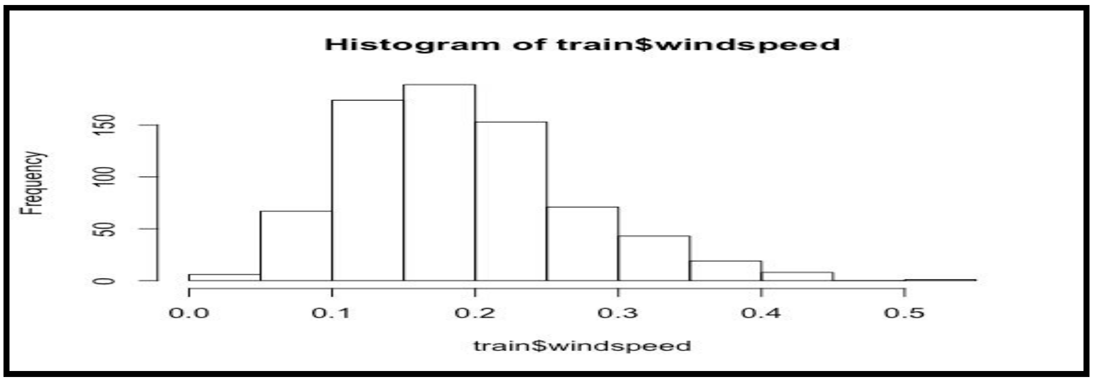
**Data exploration:**



Histogram of train$cnt
Frequency
0
50
100
150
0
2000
4000
6000
8000
train$cnt

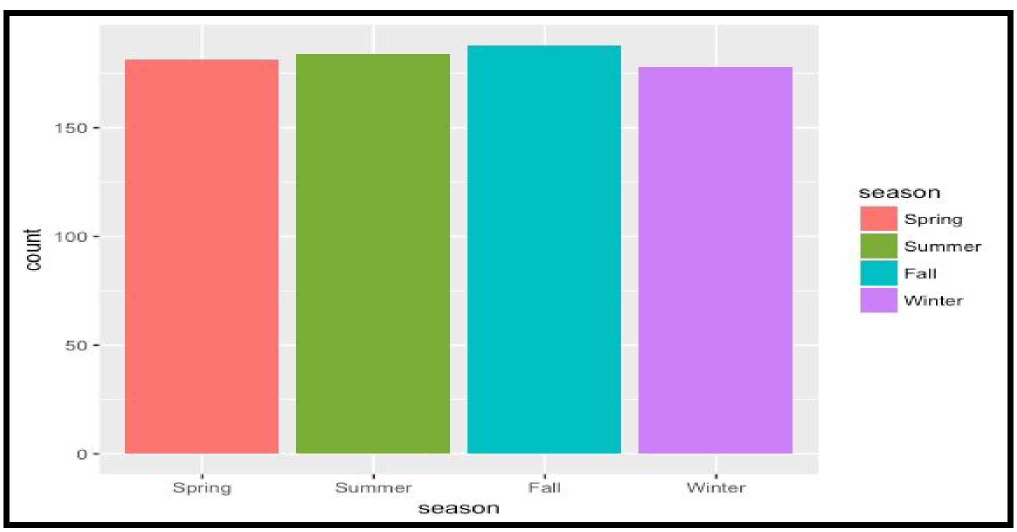

Histogram of train$temp
Frequency
0
20
40
60
80
100
0.0
0.2
0.4
0.6
0.8
train$temp


Histogram of train$atemp
Frequency
0
20
40
60
80
0.2
0.4
0.6
0.8
train$atemp

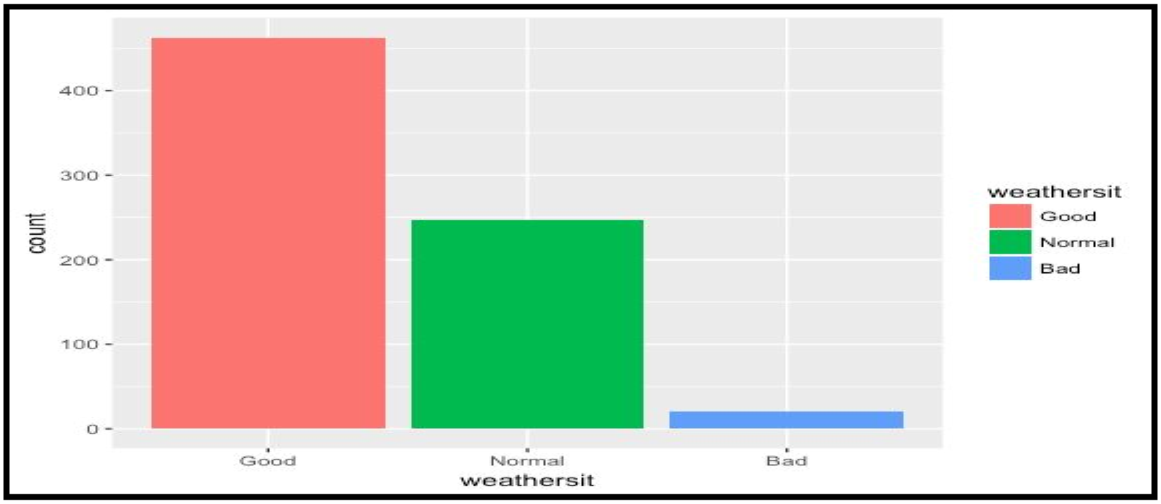


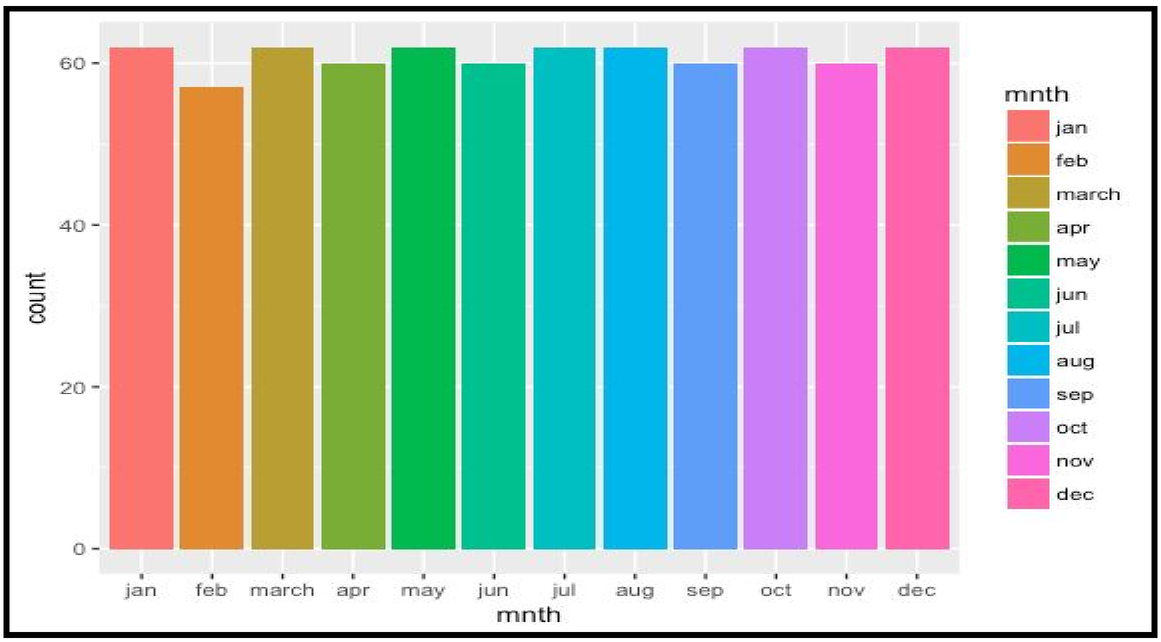

## Distribution of factor variables:
## Season



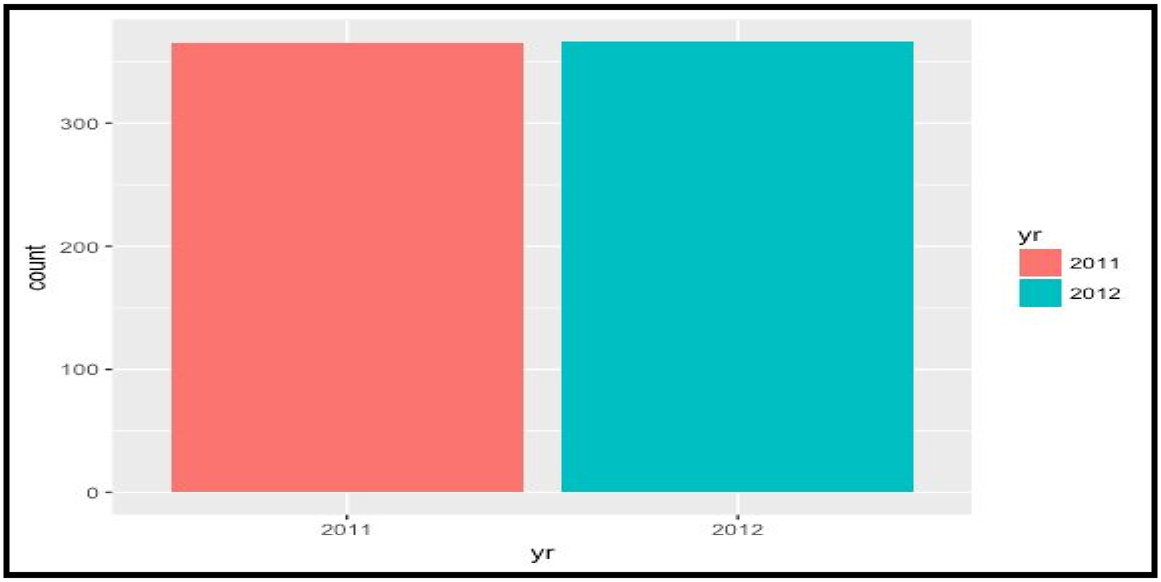
**Weather**



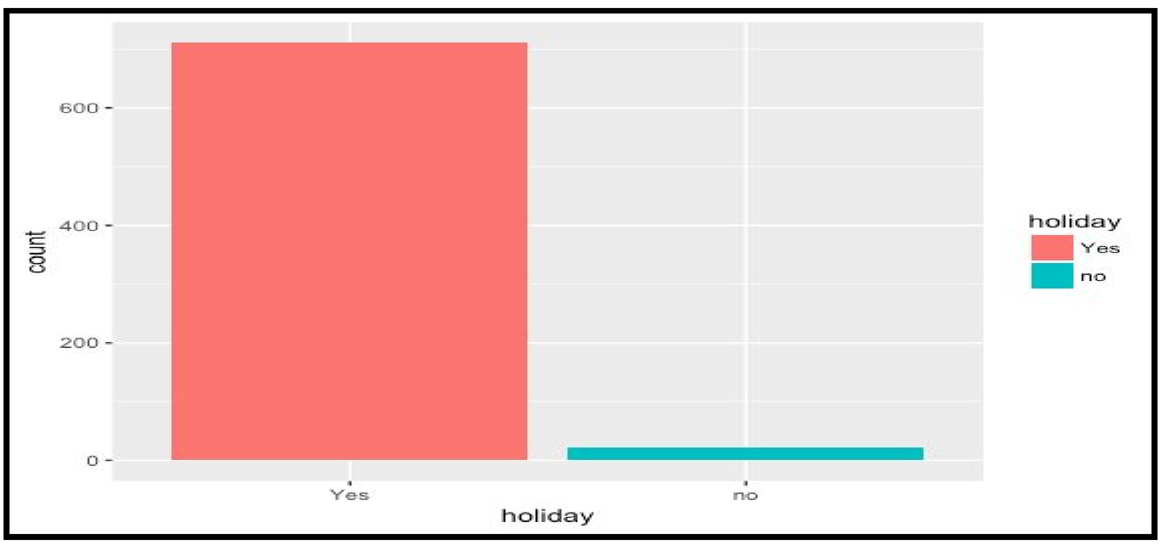

**Month**



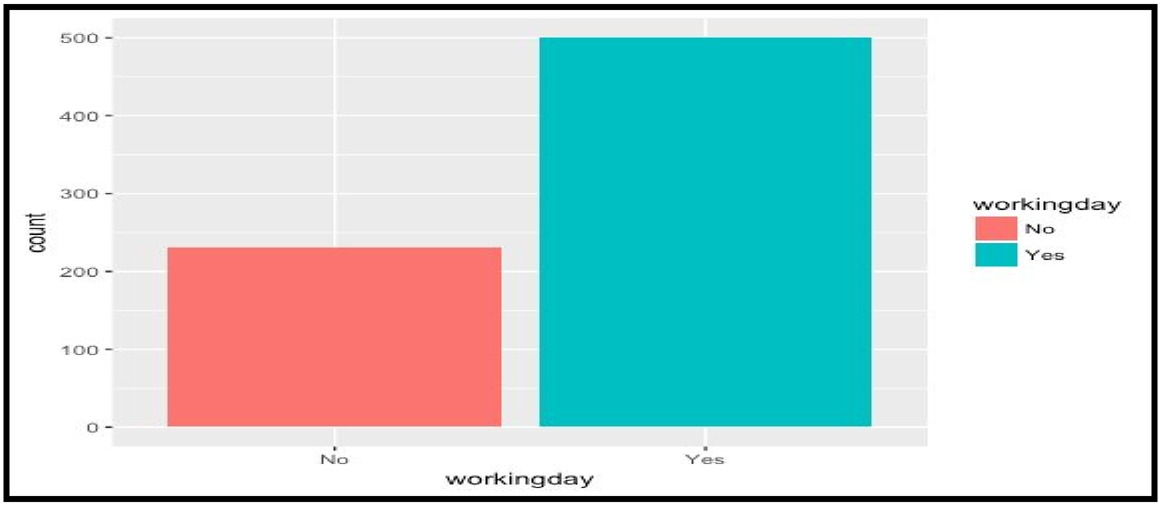
**Year**



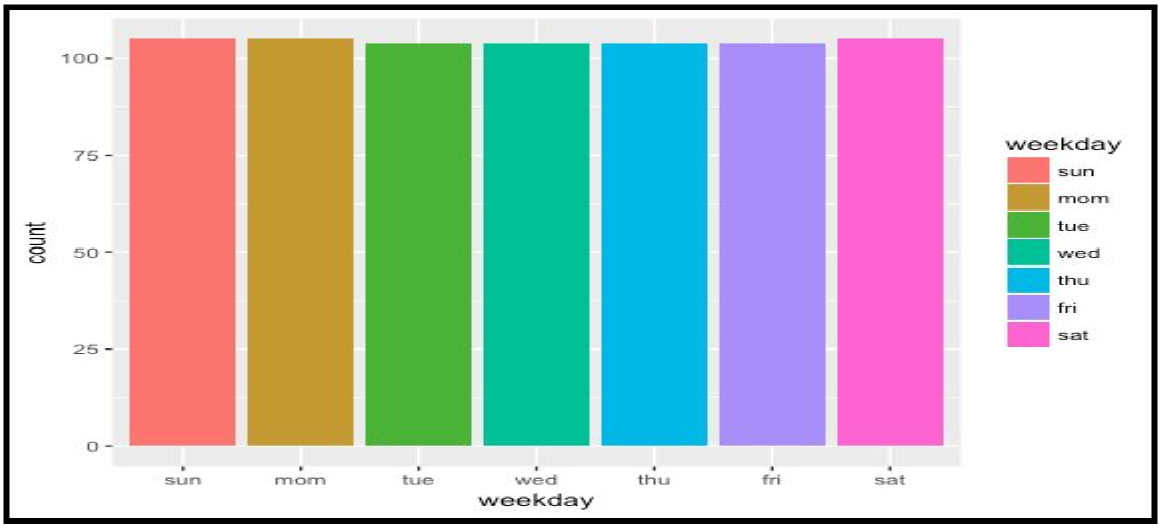

**Holiday**



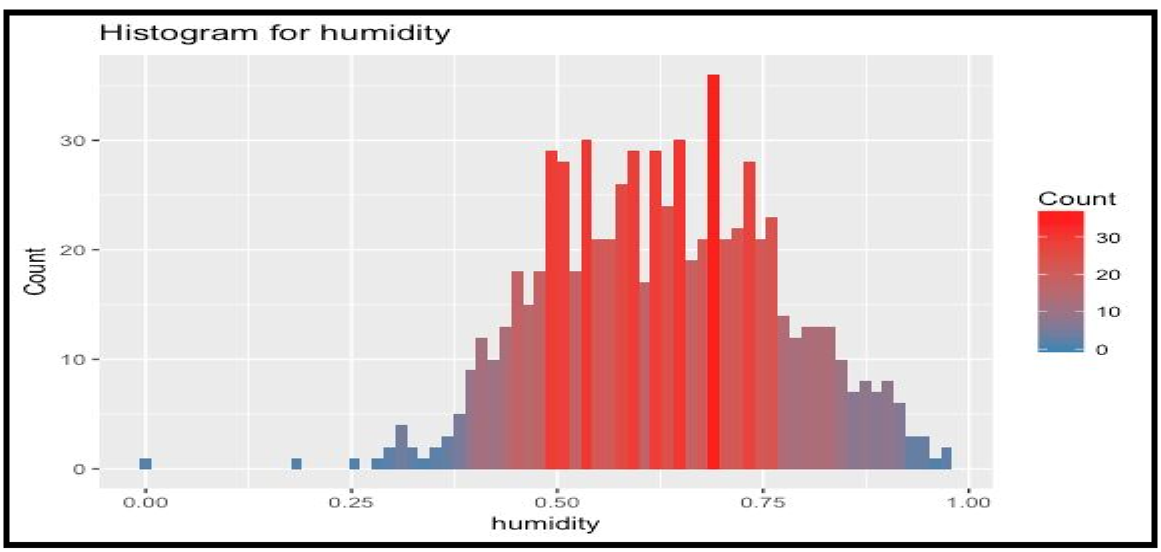
**Working day**



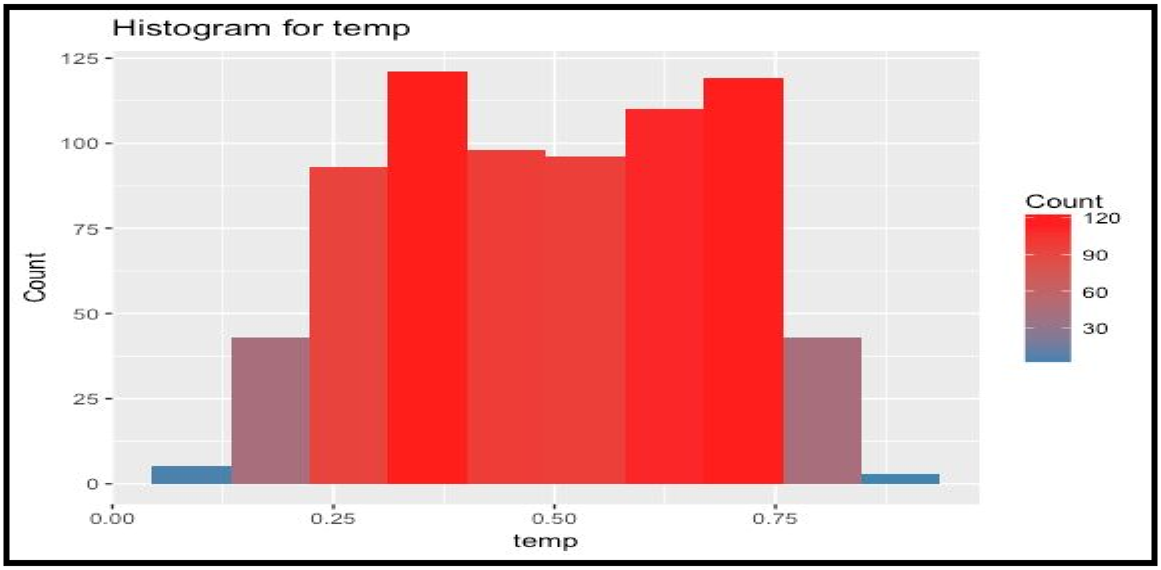
500
400
300
200
100
0
count
No
Yes
workingday
workingday
No
Yes

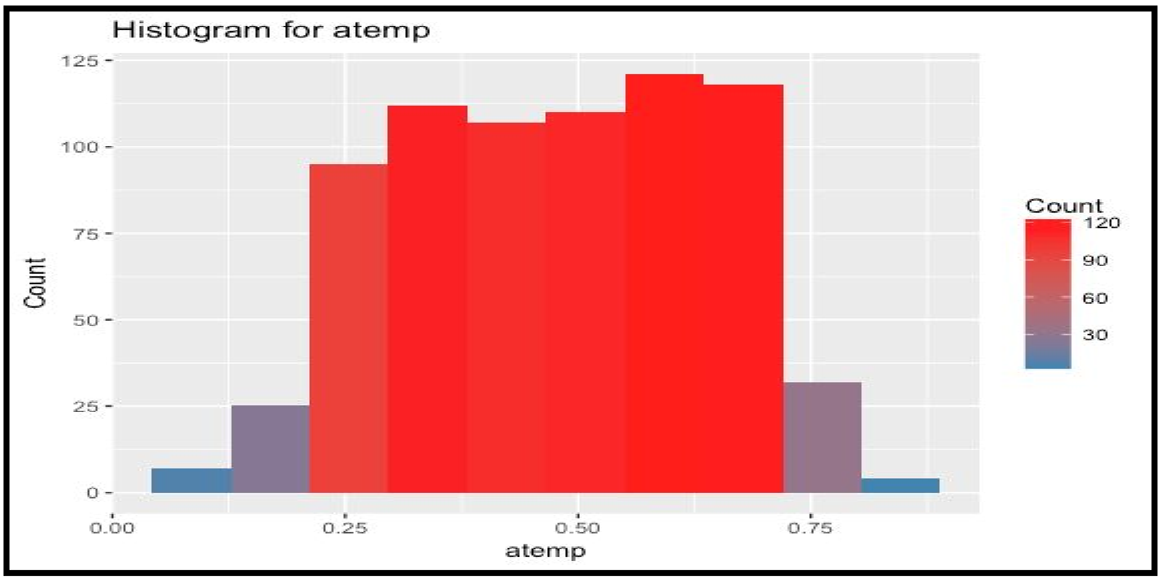
100
75
50
25
0
count
sun
mom
tue
wed
thu
fri
sat
weekday
weekday
sun
mom
tue
wed
thu
fri
sat

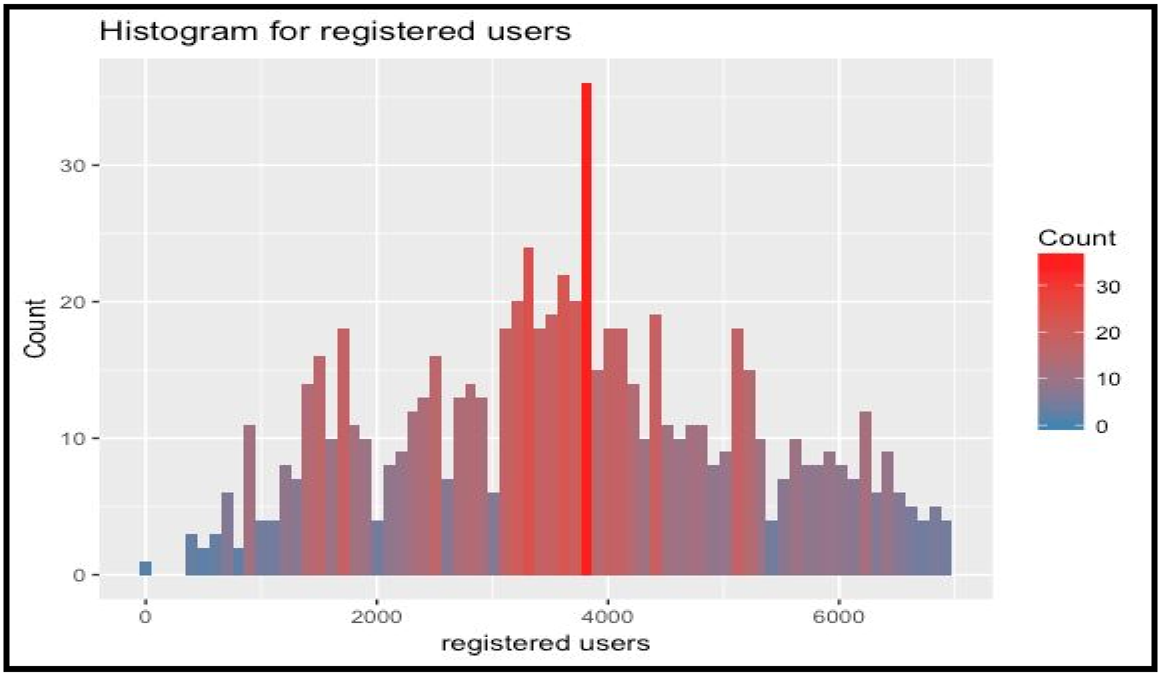
Histogram for humidity
30
20
10
0
Count
0.00
0.25
0.50
0.75
1.00
humidity
Count
30
20
10
0

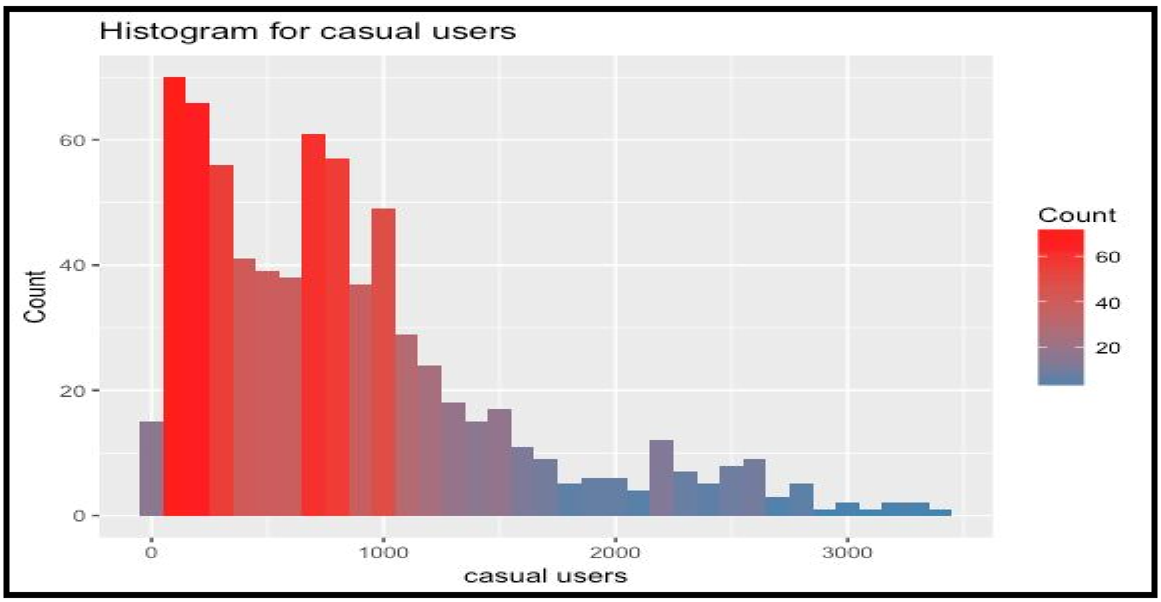


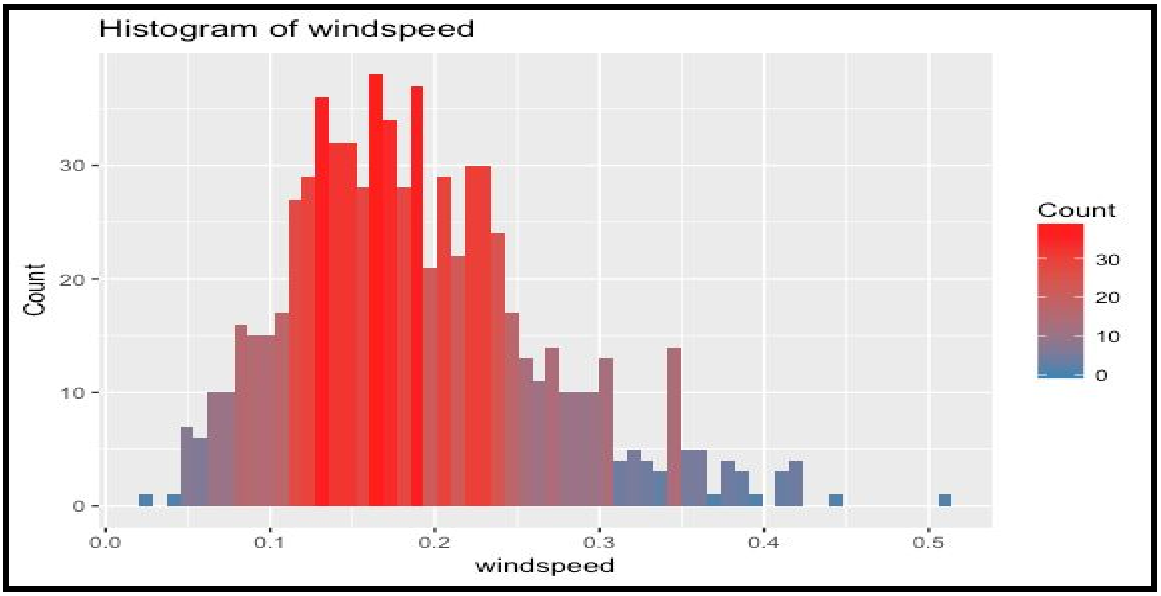
Histogram for temp
125
100
75
50
25
0
Count
0.00
0.25
0.50
0.75
temp
Count
120
90
60
30

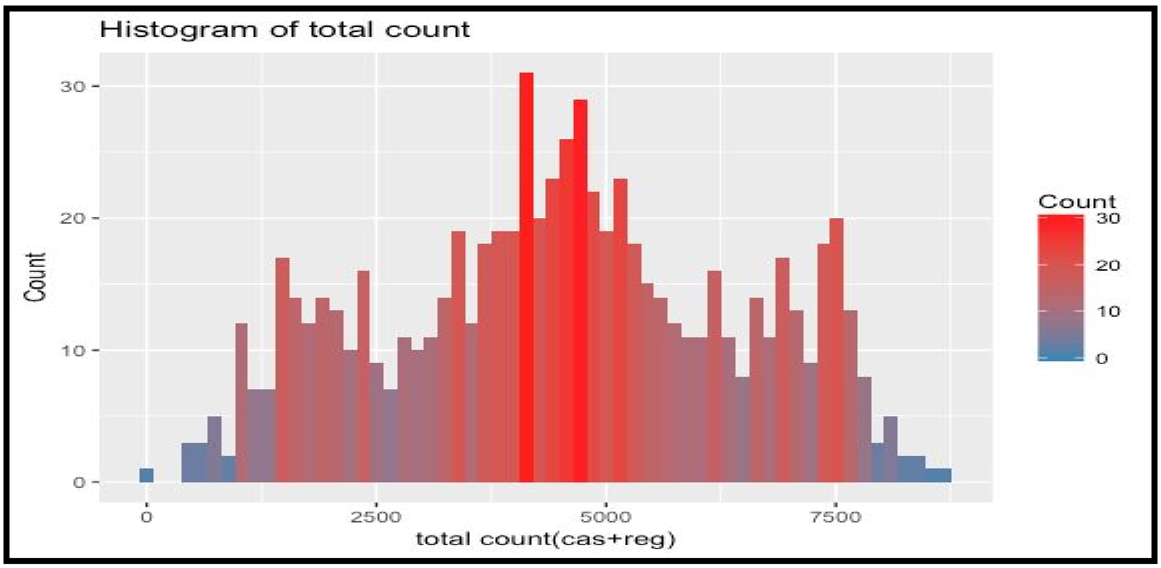



Histogram for atemp
125
100
75
50
25
0
Count
0.00
0.25
0.50
0.75
atemp
Count
120
90
60
30

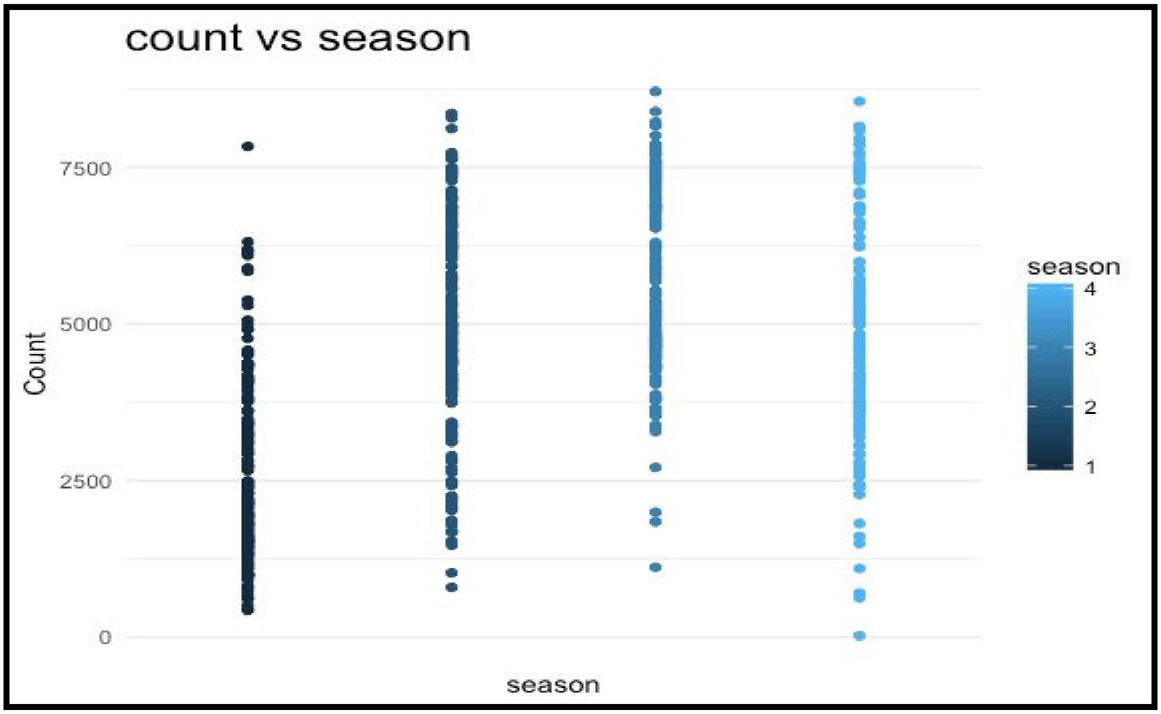

Histogram for registered users
Count
registered users
0
2000
4000
6000
0
10
20
30
Count
30
20
10
0

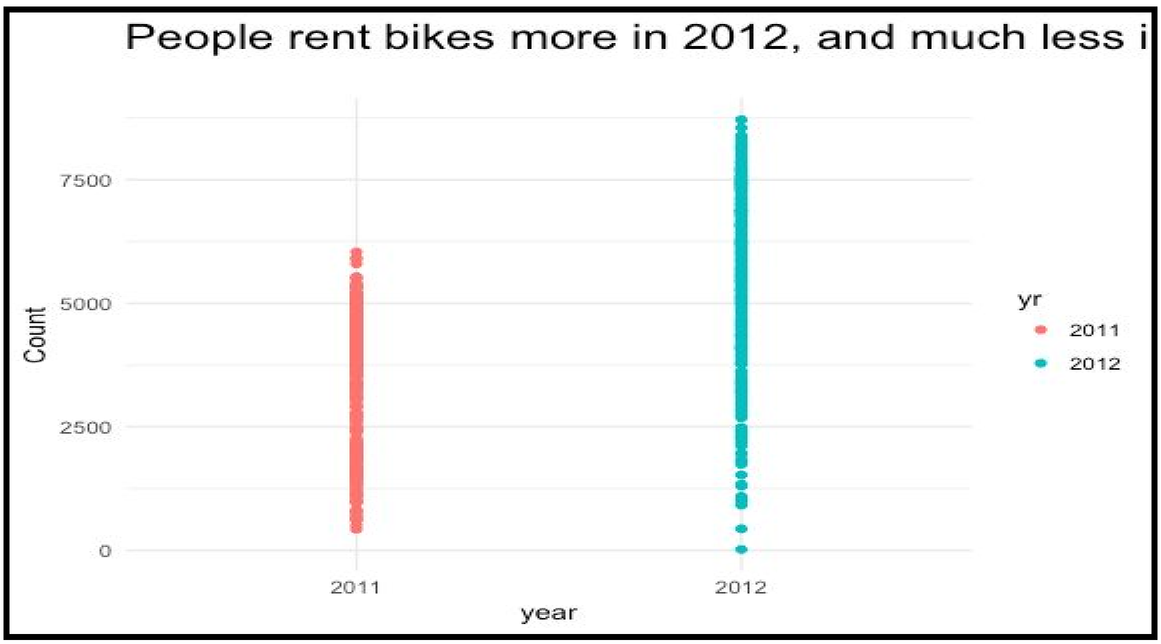

Histogram for casual users
Count
casual users
0
1000
2000
3000
0
20
40
60
Count
60
40
20

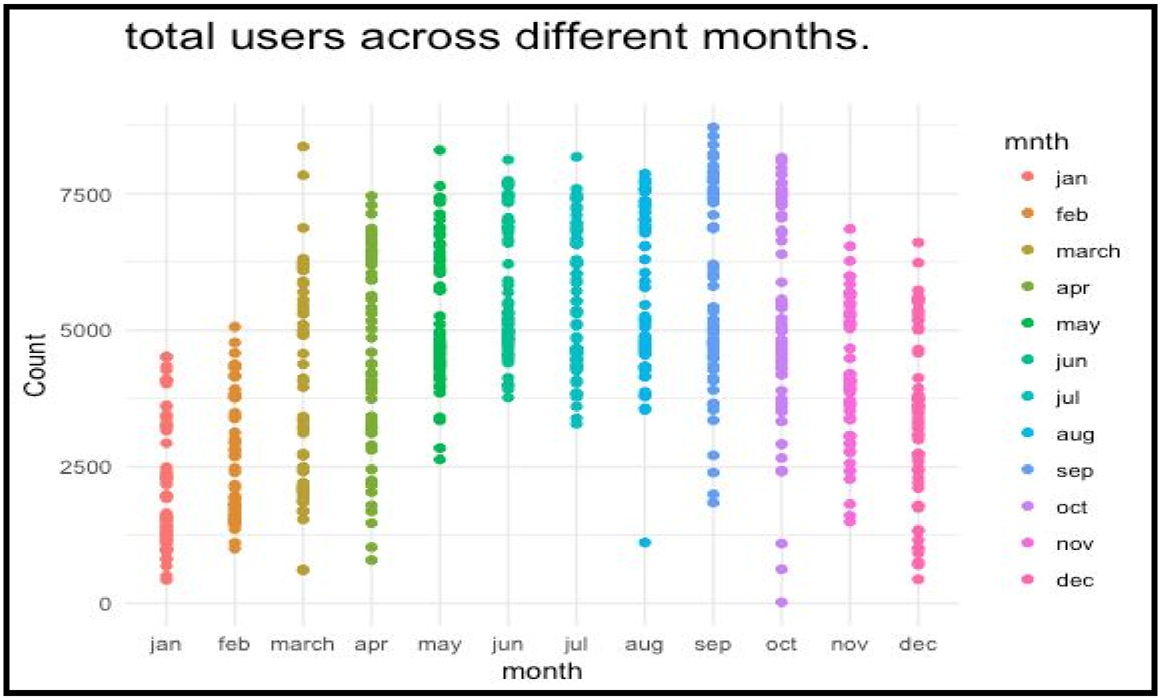


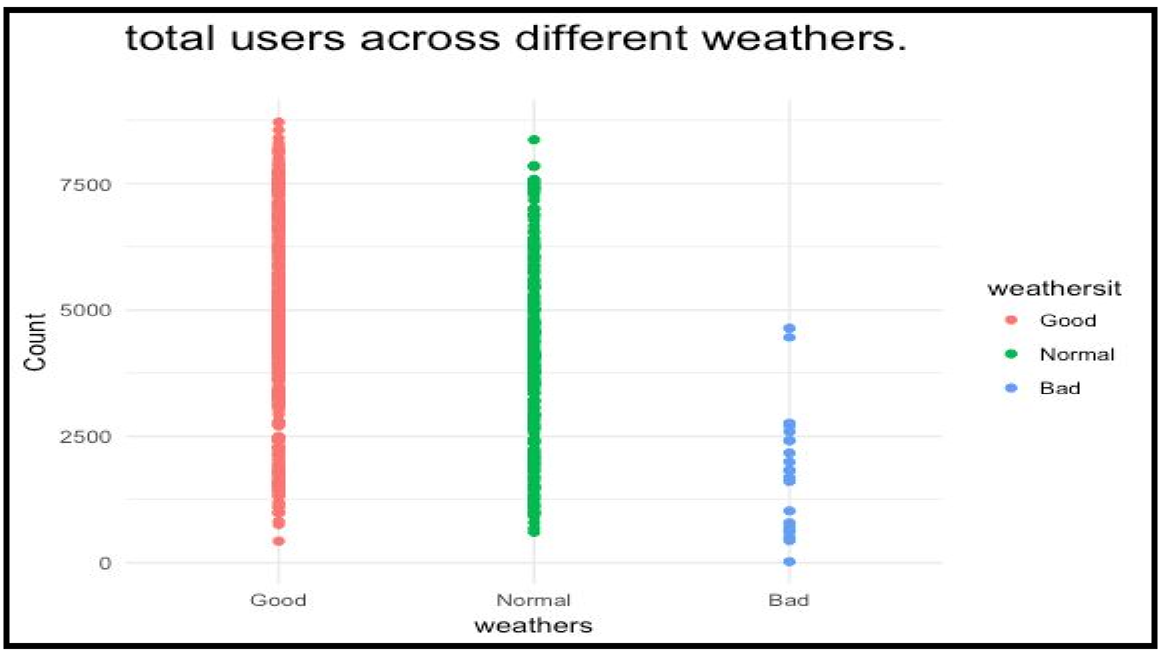



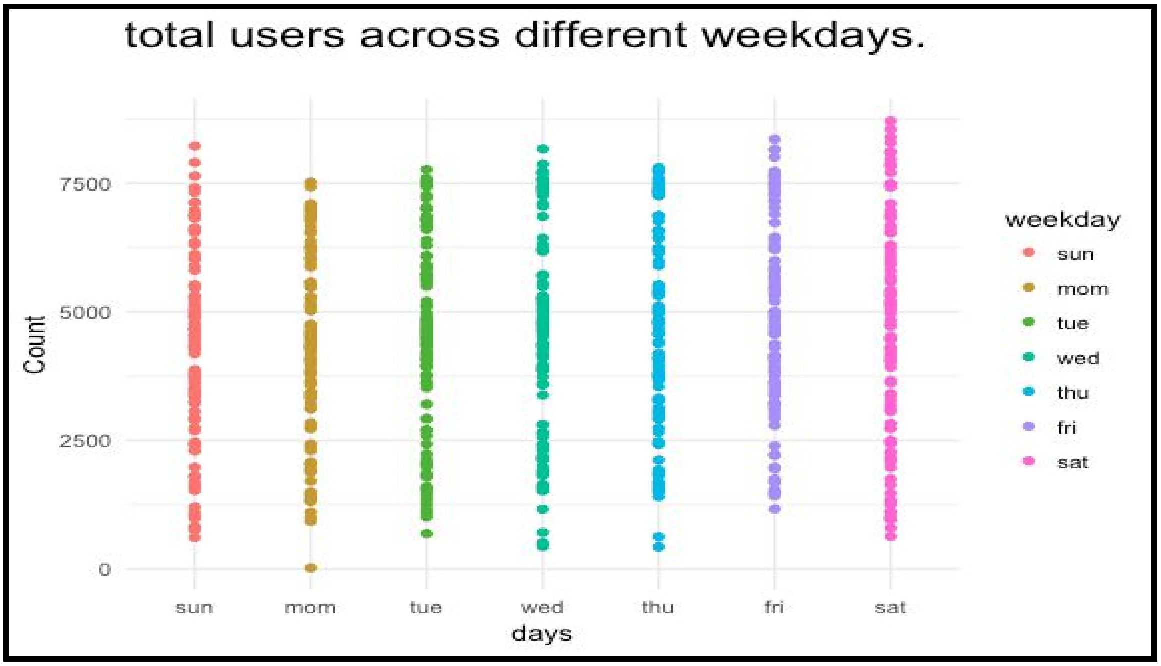

**Distribution of categorical variables across total count:**

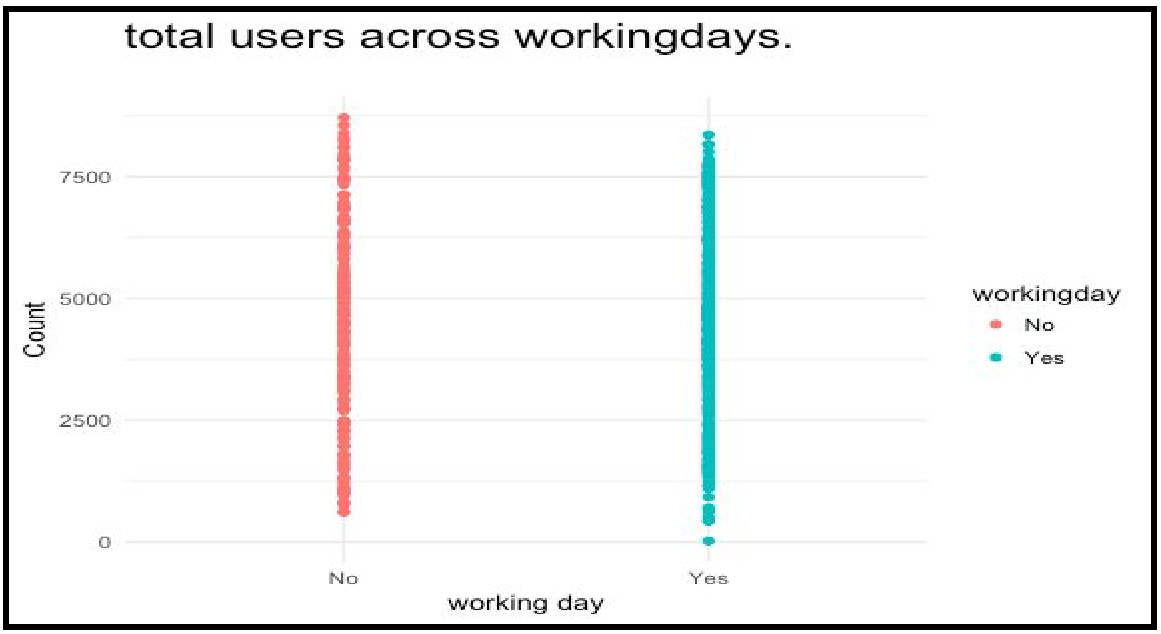

count vs season
7500
5000
2500
0
Count
season
season
4
3
2
1

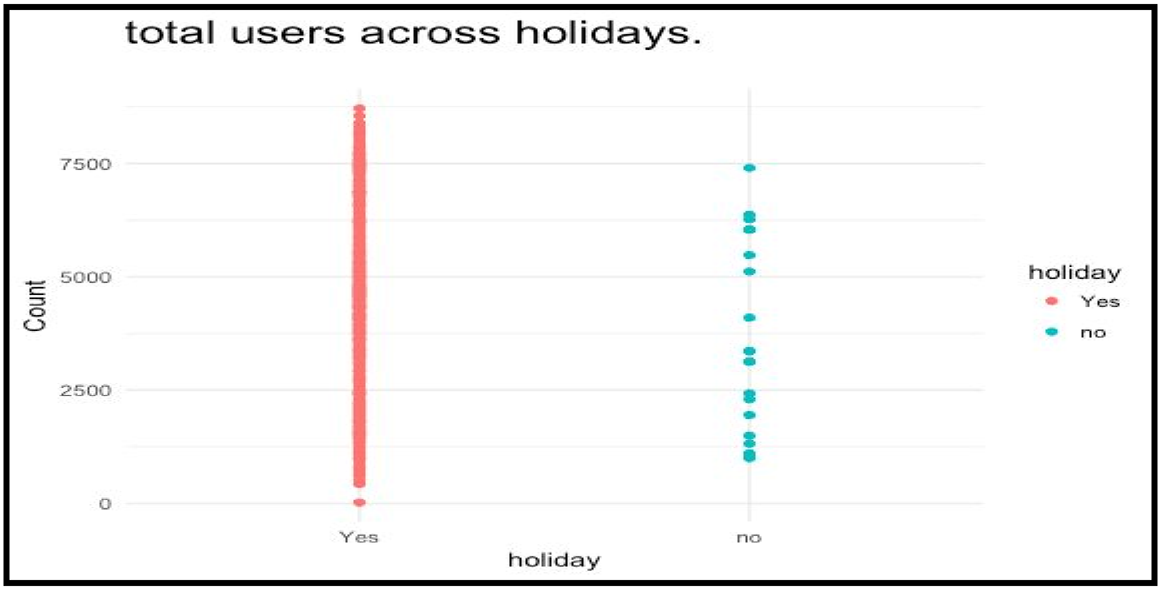

People rent bikes more in 2012, and much less i
7500
5000
2500
0
Count
2011
2012
year
yr
2011
2012


total users across different months.
Count
7500
5000
2500
0
jan
feb
march
apr
may
jun
jul
aug
sep
oct
nov
dec
month
mnth
jan
feb
march
apr
may
jun
jul
aug
sep
oct
nov
dec

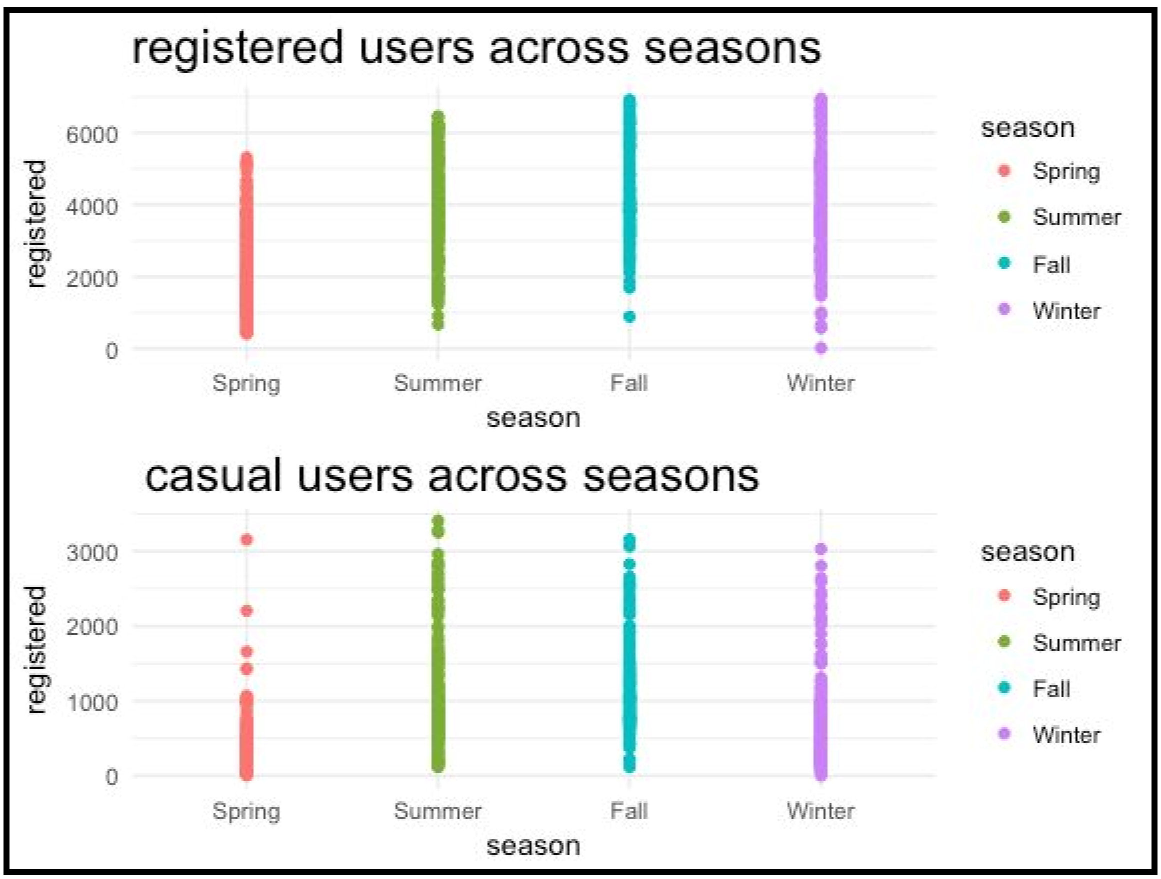

total users across different weathers.
Count
7500
5000
2500
0
Good
Normal
Bad
weathers
weathersit
Good
Normal
Bad

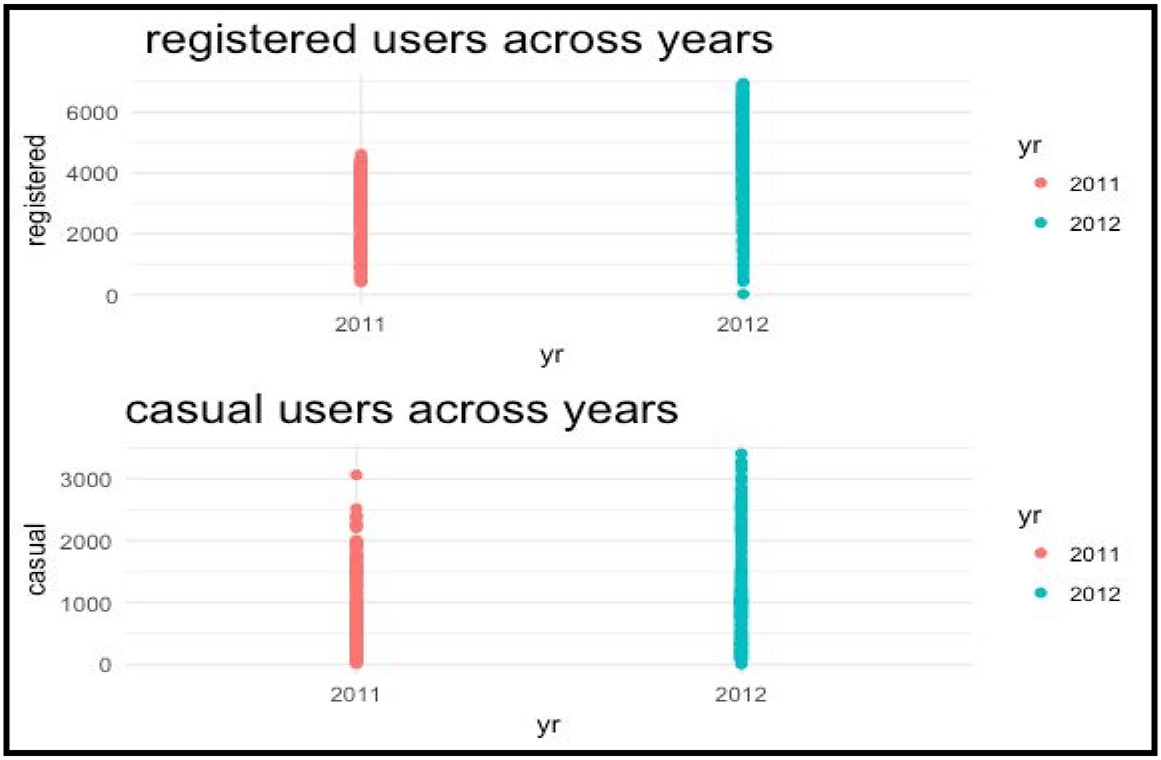

total users across different weekdays.
Count
7500
5000
2500
0
sun
mom
tue
wed
thu
fri
sat
days
weekday
sun
mom
tue
wed
thu
fri
sat

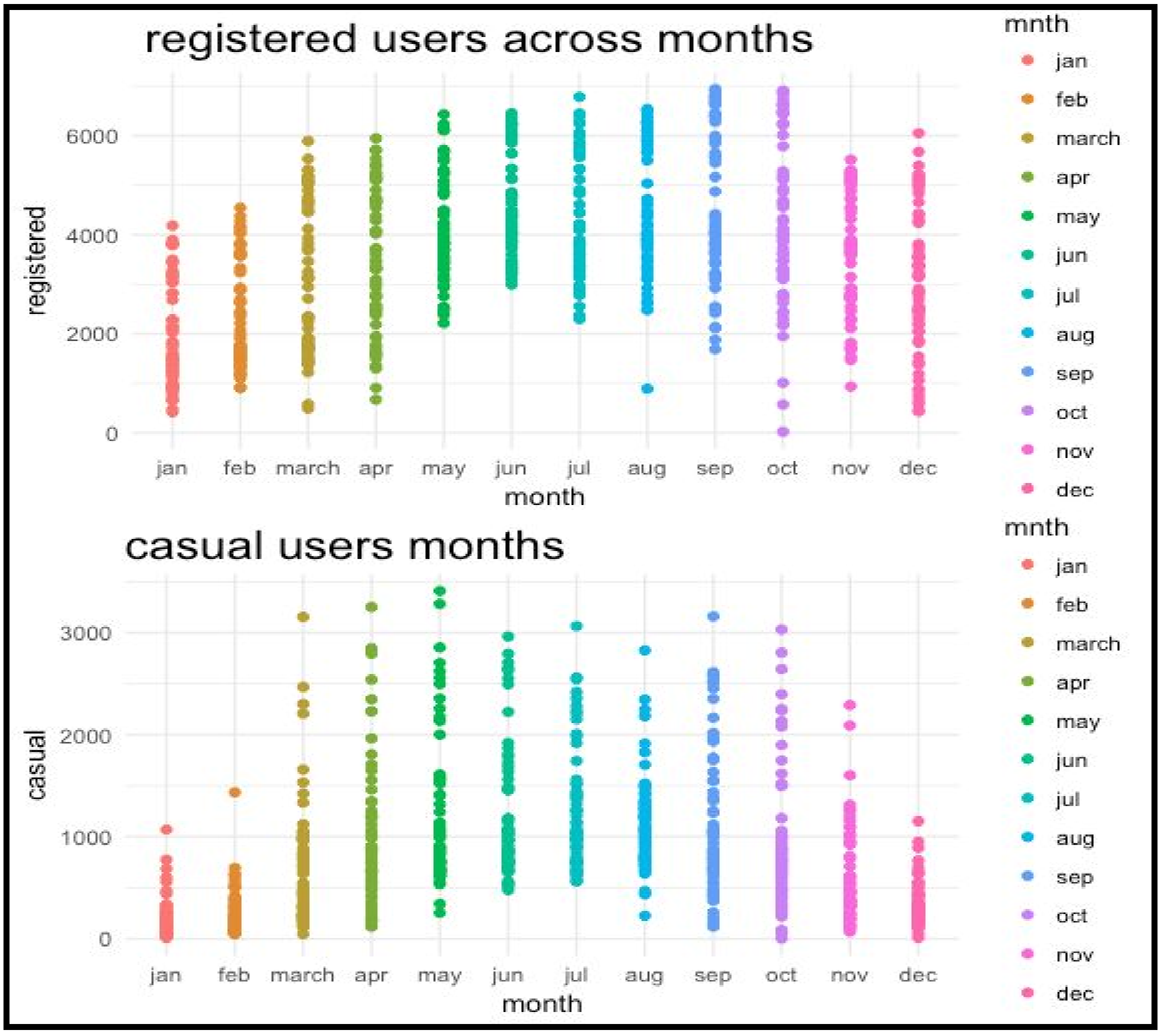

total users across workingdays.
Count
7500
5000
2500
0
No
Yes
working day
workingday
No
Yes



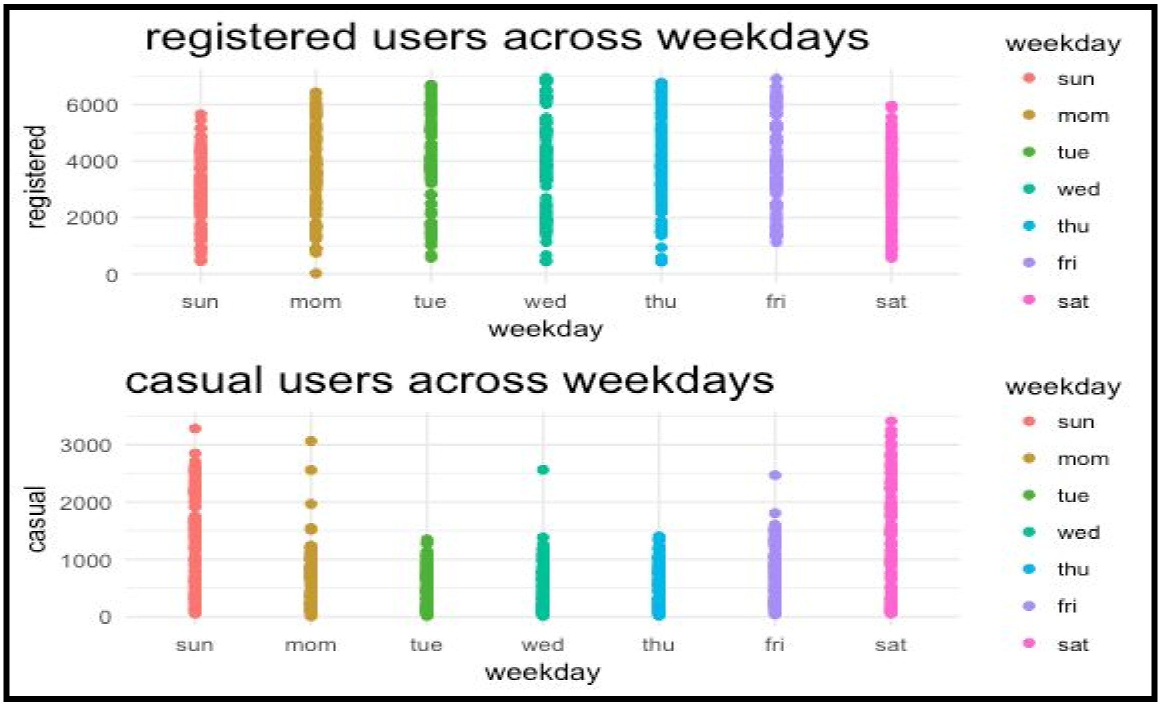

**Distribution of factor variables across casual and registered user count:**

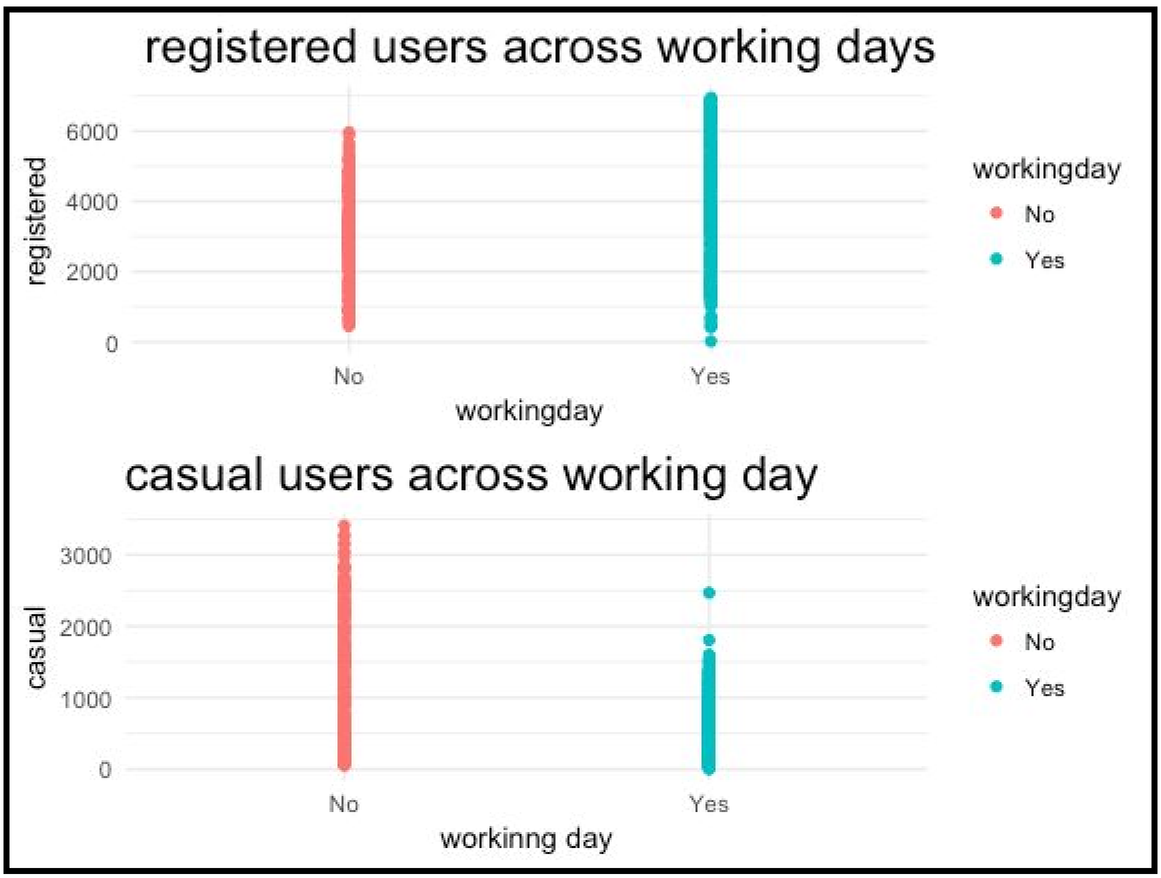


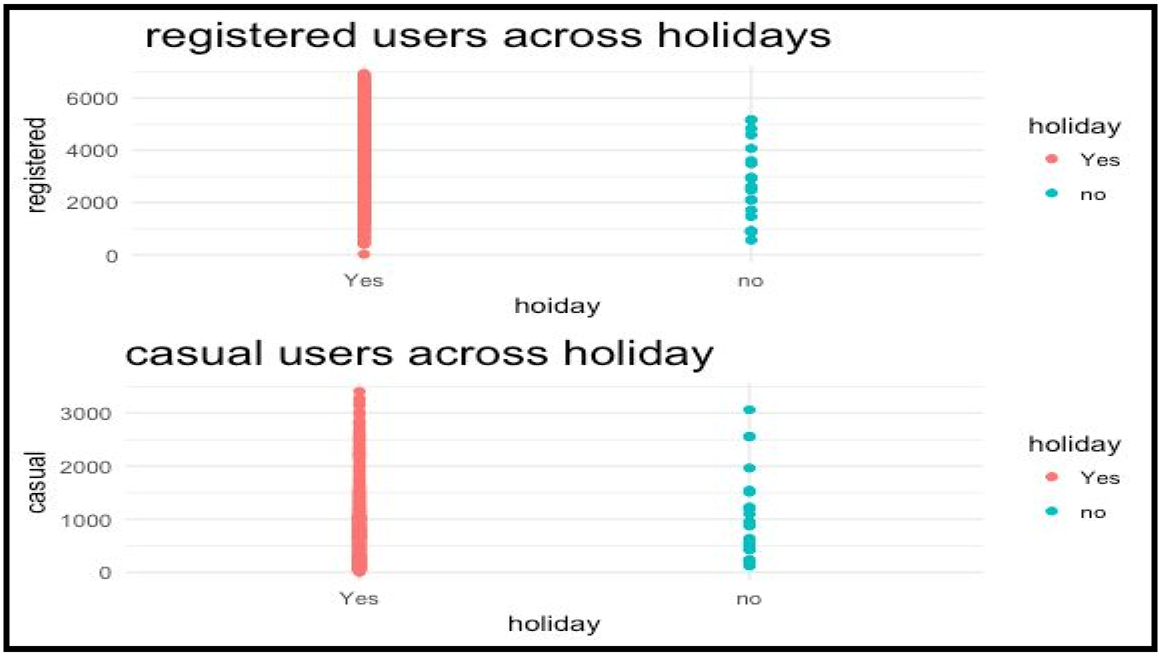

registered users across years
6000
4000
2000
0
registered
2011
2012
yr
yr
2011
2012
casual users across years
3000
2000
1000
0
casual
2011
2012
yr
yr
2011
2012

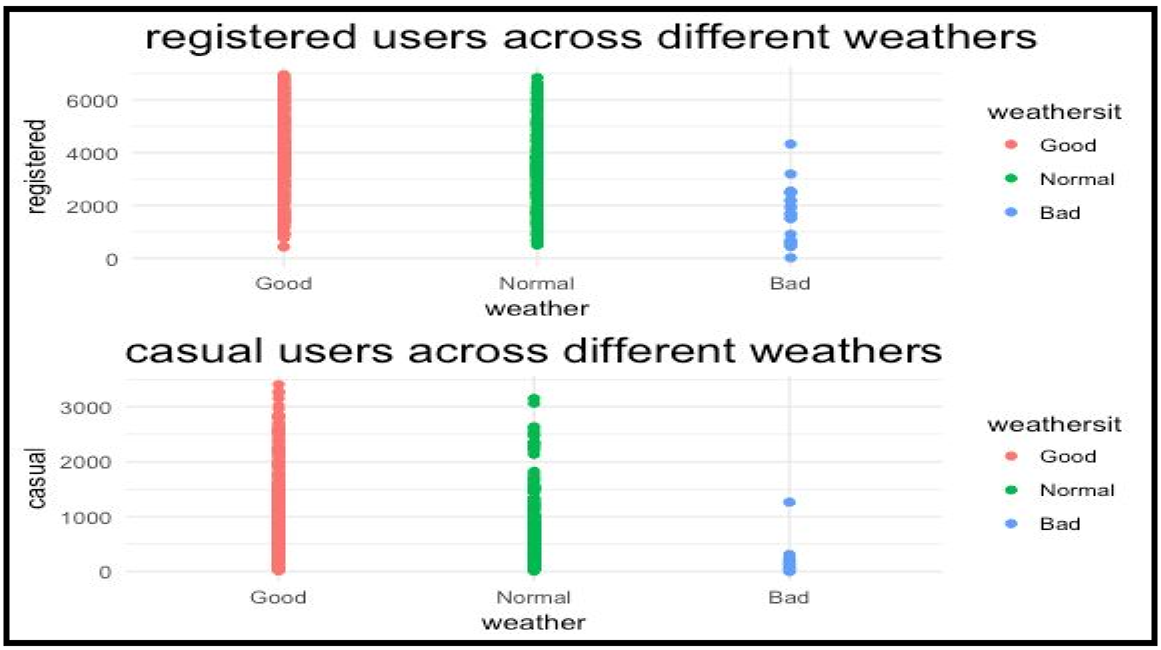


registered users across months
registered
6000
4000
2000
0
jan
feb
march
apr
may
jun
jul
aug
sep
oct
nov
dec
month
mnth
jan
feb
march
apr
may
jun
jul
aug
sep
oct
nov
dec
casual users months
casual
3000
2000
1000
0
jan
feb
march
apr
may
jun
jul
aug
sep
oct
nov
dec
month
mnth
jan
feb
march
apr
may
jun
jul
aug
sep
oct
nov
dec


registered users across weekdays
registered
6000
4000
2000
0
sun
mom
tue
wed
thu
fri
sat
weekday
weekday
sun
mom
tue
wed
thu
fri
sat
casual users across weekdays
casual
3000
2000
1000
0
sun
mom
tue
wed
thu
fri
sat
weekday
weekday
sun
mom
tue
wed
thu
fri
sat

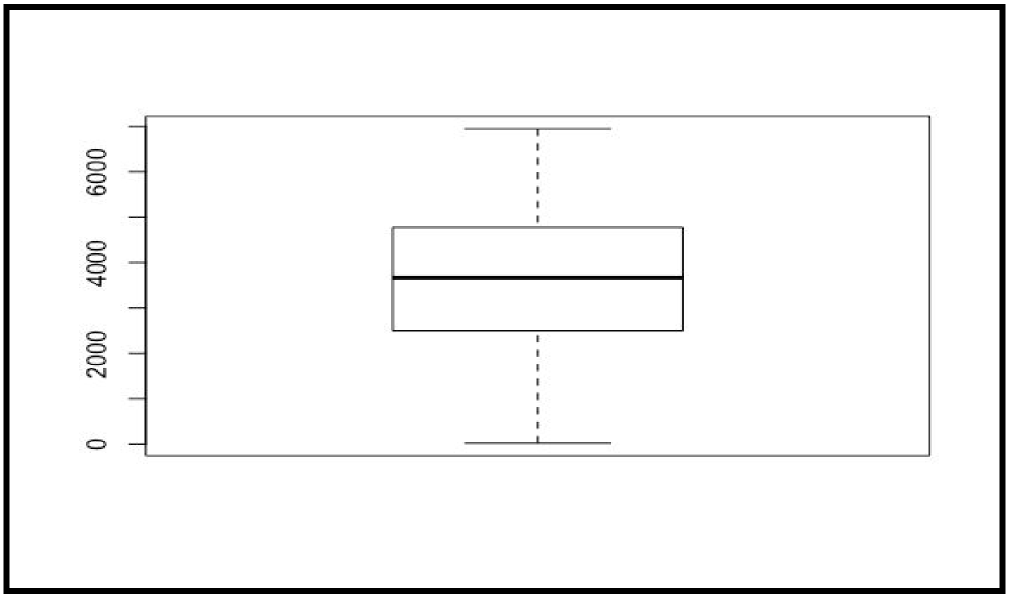

registered users across working days
registered
6000
4000
2000
0
No
Yes
workingday
workingday
No
Yes
casual users across working day
casual
3000
2000
1000
0
No
Yes
workinng day
workingday
No
Yes


registered users across holidays
6000
4000
2000
0
registered
Yes
no
hoiday
holiday
Yes
no
casual users across holiday
3000
2000
1000
0
casual
Yes
no
holiday
holiday
Yes
no

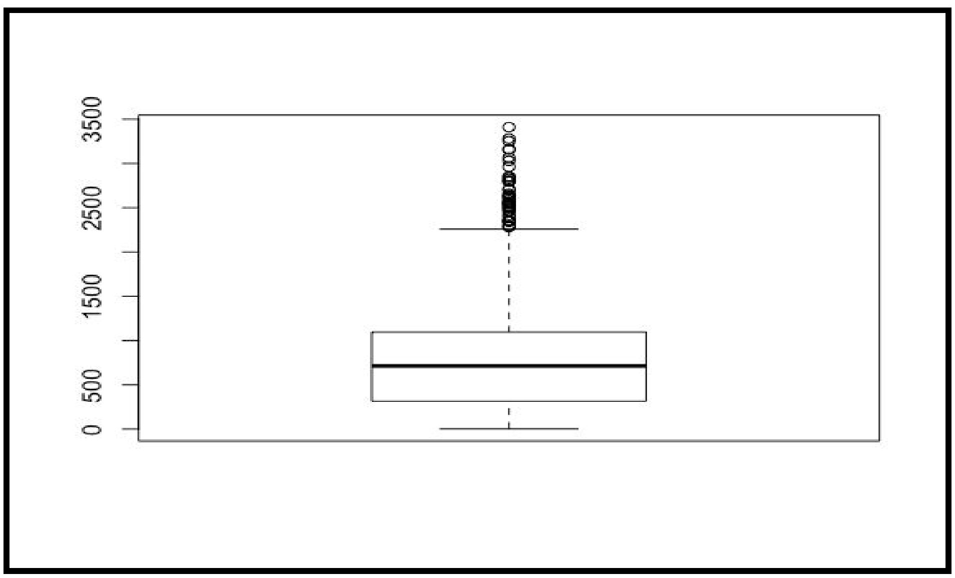

registered users across different weathers
6000
4000
2000
0
registered
Good
Normal
Bad
weather
weathersit
Good
Normal
Bad
casual users across different weathers
3000
2000
1000
0
casual
Good
Normal
Bad
weather
weathersit
Good
Normal
Bad

## Boxplot of continuous variables

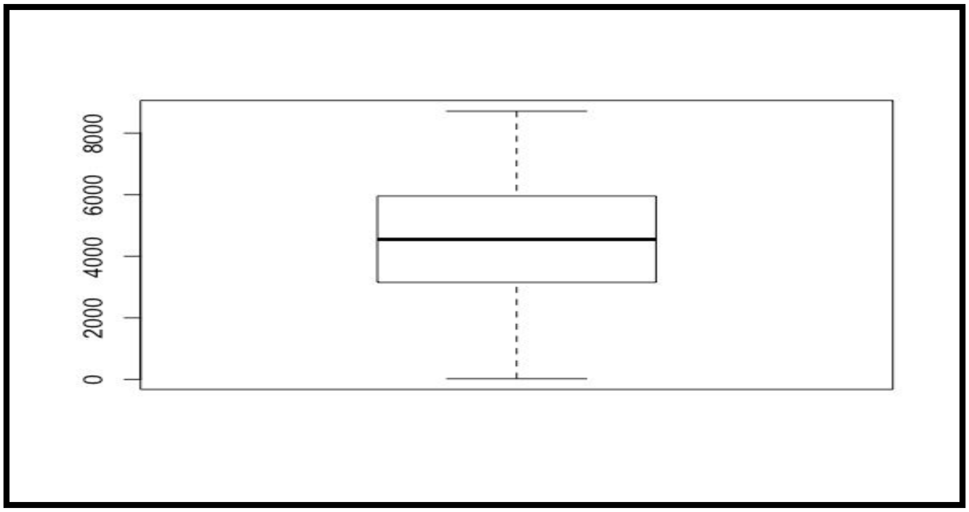
**Registered**



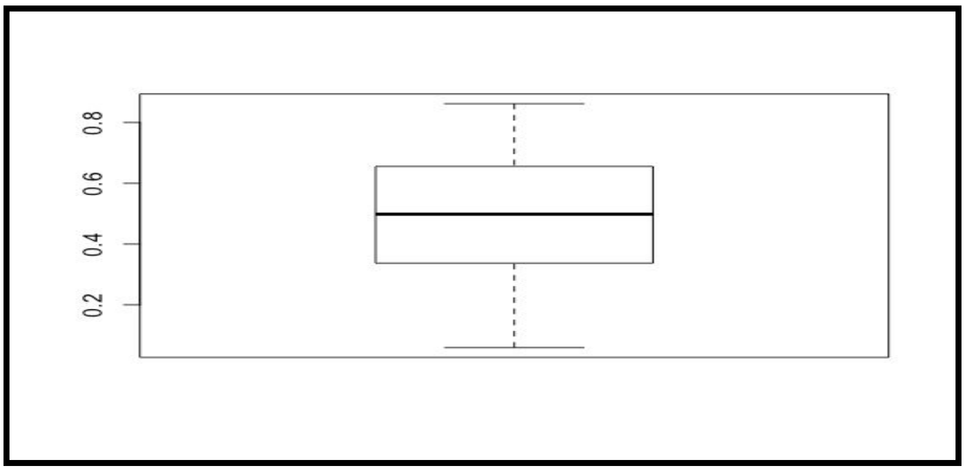

**Casual(lot of outliers detected)**



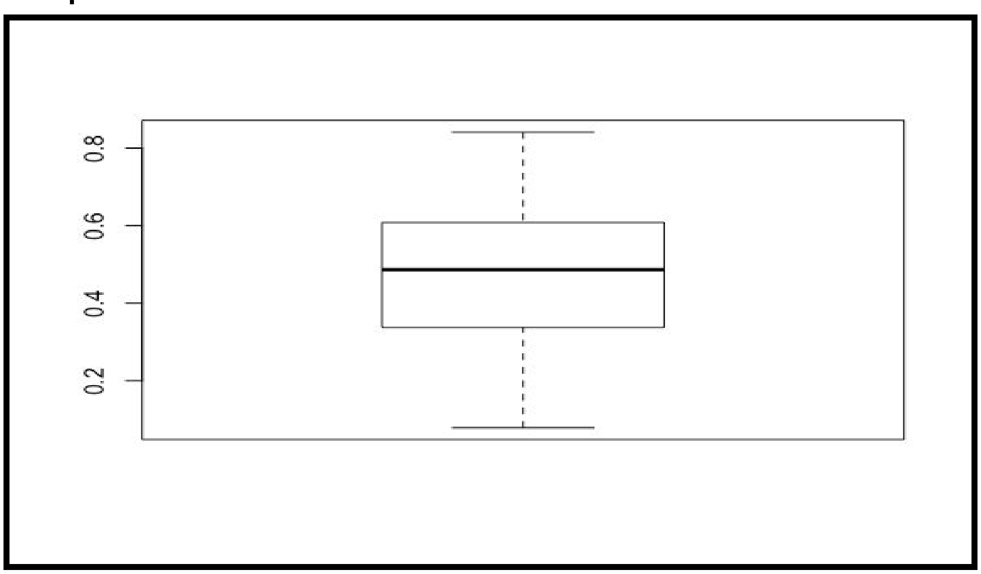
**Total count(casual+registered)**



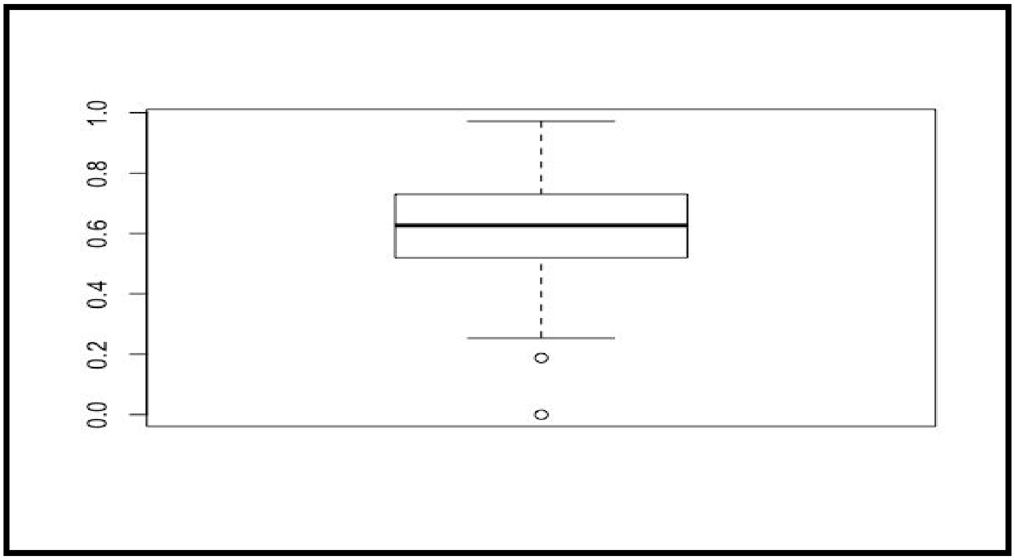

**temp**



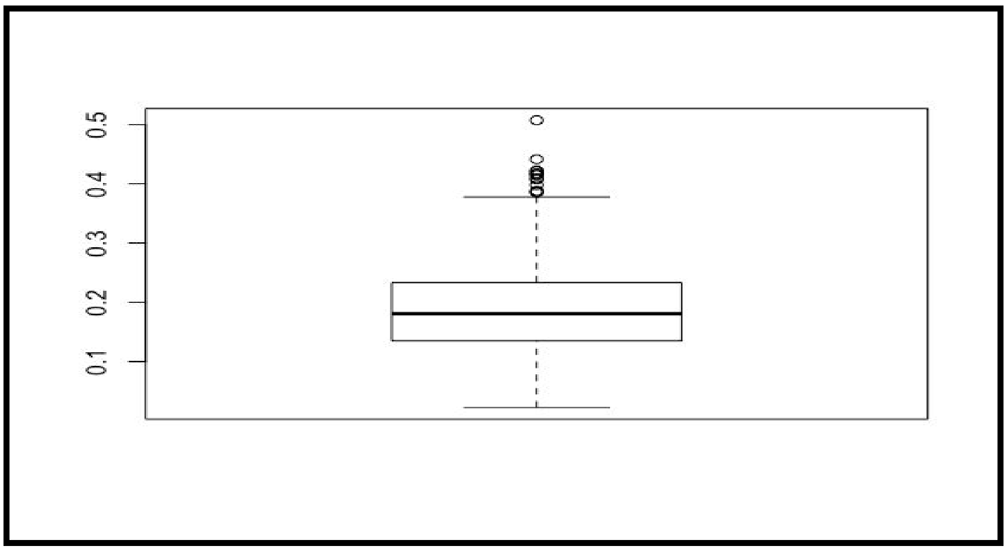

**atemp**



**Humidity**




**Windspeed(outliers detected)**



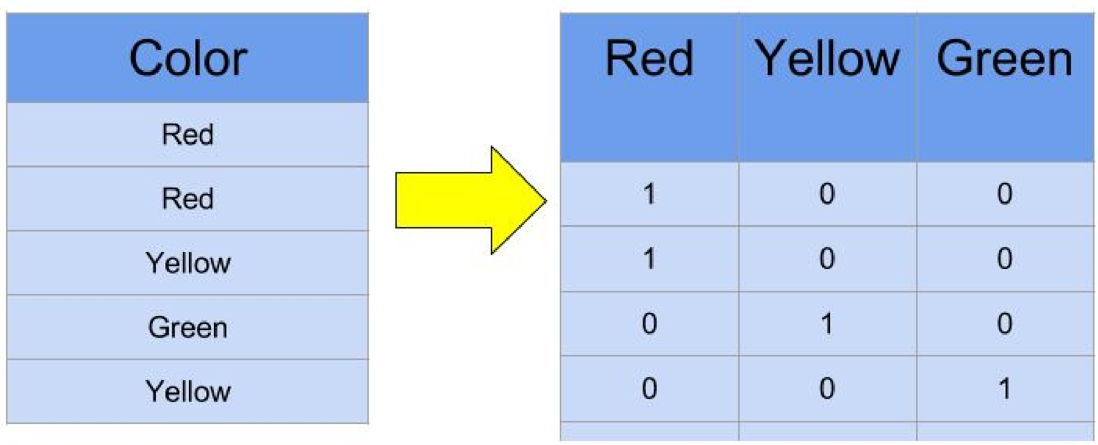
## Data preprocessing:

### One hot encoding of factor variables

**One hot encoding** is a representation of categorical variables as binary vectors. This first requires that the categorical values be mapped to integer values.

Then, each integer value is represented as a binary vector that is all zero values except the index of the integer, which is marked with a 1
I have done one hot encoding of categorical data

So the number of binary columns depend on number of factor levels in the raw column
Now all the data is converted to numeric features
By one hot encoding we will convert the factor to binary columns which is numeric in nature
For example




### Removing correlated features from the data:

Correlation is a statistical technique that can show whether and how strongly pairs of variables are related.

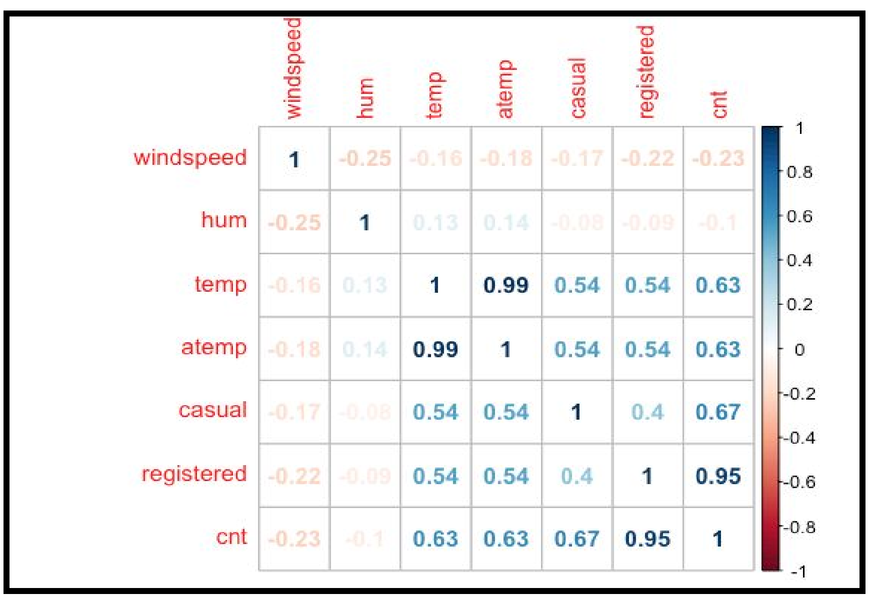
The main result of a correlation is called the **correlation coefficient** . It ranges from -1.0 to +1.0. The closer r is to +1 or -1, the more closely the two variables are related.

If r is close to 0, it means there is no relationship between the variables. If r is positive, it means that as one variable gets larger the other gets larger. If r is negative it means that as one gets larger, the other gets smaller (usally called an inverse correlation).

While correlation coefficients are normally reported as r = (a value between -1 and +1), squaring them makes then easier to understand. The square of the coefficient (or r square) is equal to the percent of the variation in one variable that is related to the variation in the other. After squaring r, ignore the decimal point. An r of .5 means 25% of the variation is related (.5 squared =.25). An r value of .7 means 49% of the variance is related (.7 squared = .49).

**I have removed highly correlated features (having high correlation coefficient [>0.95] )**

"atemp" is variable is not taken into since "atemp" and "temp" has
got strong correlation with each other.
During model building any one of the variable has to be dropped since
they will exhibit multicollinearity in the data.



## Chapter 4:

### Feature Engineering:

Feature engineering is the process of using **domain knowledge** of the data to create features that make machine learning algorithms work. If feature engineering is done correctly, it increases the predictive power of machine learning algorithms by creating features from raw data that help facilitate the machine learning process. Feature Engineering is an art.

**Steps which are involved while solving any problem in machine learning are as follows:**

- Gathering data.
- Cleaning data.
- **Feature engineering**.
- Defining model.
- Training, testing model and predicting the output.

I have used Boruta package in r
And selectKbest in python

**Boruta**

Boruta is a feature selection algorithm. Precisely, it works as a wrapper algorithm around Random Forest. This package derive its name from a demon in Slavic mythology who dwelled in pine forests.

We know that feature selection is a crucial step in predictive modeling. This technique achieves supreme importance when a data set comprised of several variables is given for model building.

Boruta can be your algorithm of choice to deal with such data sets. Particularly when one is interested in understanding the mechanisms related to the variable of interest, rather than just building a black box predictive model with good prediction accuracy

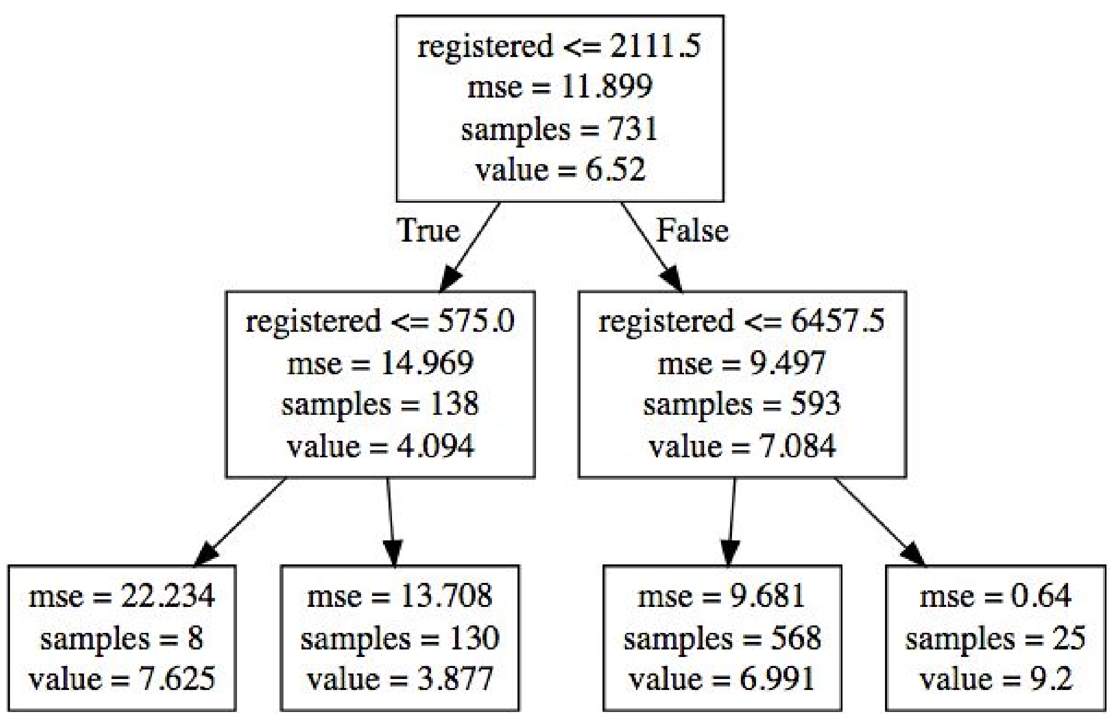
**seleckKBest:**

SelectKBest selects the top k features that have maximum relevance with the target variable. It takes two parameters as input arguments, "k" (obviously) and the score function to rate the relevance of every feature with the target variable. For example, for a regression problem, you can supply

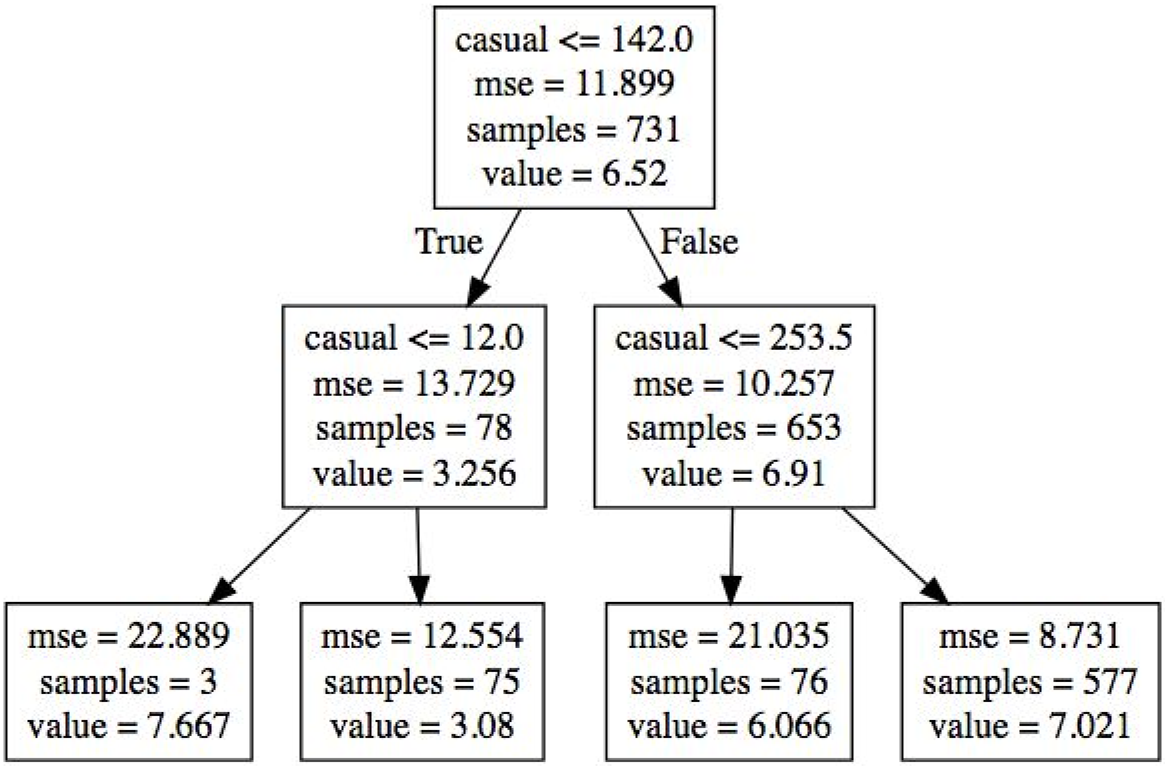
Select features according to the k highest scores. The score function must return an array of scores, one for each feature (additionally, it can also return p-values, but these are neither needed nor required). SelectKBest then simply retains the first k features with the highest scores.

**Feature selection:**

**New features are created according to the graph(registered and month)**



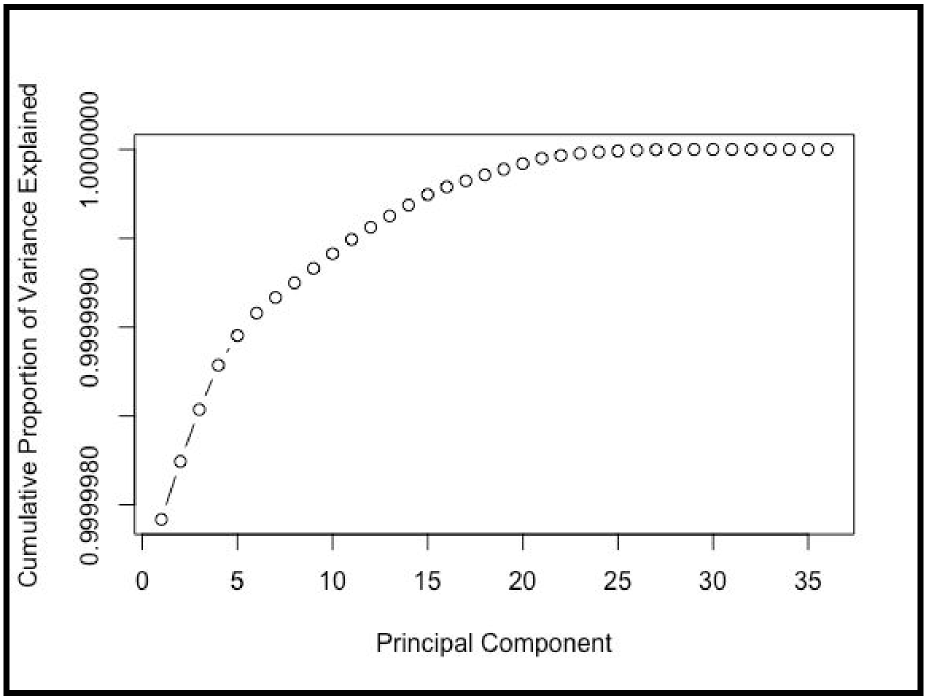

**New features are created according to the graph(casual and month)**




**Principal component analysis:**

As we have more features in the data(new columns obtained from one hot encoding) the dimension will be large.
So i have used Principal component analysis to reduce the dimension of the data keeping the variance unchanged
Principal component analysis (PCA) is a technique used to emphasize variation and bring out strong patterns in a dataset. It's often used to make data easy to explore and visualize.

Principal component analysis is a technique for feature extraction — so it combines our input variables in a specific way, then we can drop the least important variables while still retaining the most valuable parts of all of the variables! As an added benefit, each of the new variables after PCA are all independent of one another. This is a benefit because the assumptions of a linear model require our independent variables to be independent of one another. If we decide to fit a linear regression model with these new variables (see principal component regression below), this assumption will necessarily be satisfied.




From the above plot ~25 variables clearly explains the almost 100% of variance in the data

**So PCA has reduced 36 variables to 25 without compromising the variance in the data**

**Sampling methods**
K-fold repeated CV

- k-fold cross-validation randomly divides the data into k blocks of roughly equal size. Each of the blocks is left out in turn and the other k-1 blocks are used to train the model. The held out block is predicted and these predictions are summarized into some type of performance measure (e.g. accuracy, root mean squared error (RMSE), etc.). The k estimates of performance are averaged to get the overall resampled estimate. k is 10 or sometimes 5.
- Repeated k-fold CV does the same as above but more than once. For example, five repeats of 10-fold CV would give 50 total resamples that are averaged. Note this is not the same as 50-fold CV.

## Chapter 5:

**Multiple Linear Regression:**
Multiple linear regression (MLR) is a statistical technique that uses several explanatory variables to predict the outcome of a response variable.
The goal of multiple linear regression (MLR) is to model the relationship between the explanatory and response variables.
Multiple linear regression (MLR) is used to determine a mathematical relationship among a number of random variables.
In other terms, MLR examines how multiple independent variables are related to one dependent variable.
Once each of the independent factors have been determined to predict the dependent variable, the information on the multiple variables can be used to cr
eate an accurate prediction on the level of effect they have on the outcome variable. The model creates a relationship in the form of a straight line (linear) that best approximates all the individual data points.

The model for MLR, given n observations, is:
$y_i = B_o + B_1x_{i1} + B_2x_{i2} + ... + B_px_{ip} + E$ where i = 1,2, ..., n

**Decision tree**

Decision Tree algorithm belongs to the family of supervised learning algorithms. Unlike other supervised learning algorithms, decision tree algorithm can be used for solving regression and classification problems too.
The general motive of using Decision Tree is to create a training model which can use to predict class or value of target variables by learning decision rules inferred from prior data(training data).

The understanding level of Decision Trees algorithm is so easy compared with other classification algorithms. The decision tree algorithm tries to solve the problem, by using tree representation. Each internal node of the tree corresponds to an attribute, and each leaf node corresponds to a class label.

Decision Tree Algorithm Pseudocode

1. Place the best attribute of the dataset at the root of the tree.
2. Split the training set into subsets. Subsets should be made in such a way that each subset contains data with the same value for an attribute.

3. Repeat step 1 and step 2 on each subset until you find leaf nodes in all the branches of the tree.

**Random forest**

A group of decision tree is nothing but a random forest

The Random Forest is one of the most effective machine learning models for predictive analytics, making it an industrial tool for machine learning.

Background process

The random forest model is a type of additive model that makes predictions by combining decisions from a sequence of base models. More formally we can write this class of models as:

$$g(x)=f0(x)+f1(x)+f2(x)+...$$

where the final model $g$ is the sum of simple base models $f_i$. Here, each base classifier is a simple decision tree. This broad technique of using multiple models to obtain better predictive performance is called model ensembling. In random forests, all the base models are constructed independently using a different subsample of the data.

**Lasso regression:**

Lasso is another extension built on regularized linear regression, but with a small twist

The only difference from Ridge regression is that the regularization term is in absolute value. But this difference has a huge impact.Lasso method overcomes the disadvantage of Ridge regression by not only punishing high values of the coefficients β but actually setting them to zero if they are not relevant. Therefore, we might end up with fewer features included in the model than you started with, which is a huge advantage.

**Ridge regression:**

Ridge regression is an extension for linear regression. It's basically a regularized linear regression model. The λ parameter is a scalar that should be learned as well, using a method called cross validation that will be discussed in another post.

A super important fact we need to notice about ridge regression is that it enforces the β coefficients to be lower, but it does not enforce them to be zero. That is, it will not get rid of irrelevant features but rather minimize their impact on the trained model.

# Chapter 6:
## Model results

**[1] "ridge"**
[1] " model on casual count"
Ridge Regression

500 samples
 33 predictor

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 6 times)
Summary of sample sizes: 449, 449, 451, 449, 451, 450, ...
Resampling results across tuning parameters:

| lambda | RMSE | Rsquared | MAE |
|---|---|---|---|
| 0e+00 | 335.0056 | 0.7110740 | 243.1953 |
| 1e-04 | 335.0042 | 0.7110768 | 243.1931 |
| 1e-01 | 339.0056 | 0.7074126 | 247.1517 |

RMSE was used to select the optimal model using the smallest value.
The final value used for the model was lambda = 1e-04.
**[1] "test RMSE of casual count prediction"**
**[1] 426.2306**
[1] "model on registered model"
[1] "registered count"
Ridge Regression

500 samples
 29 predictor

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 6 times)
Summary of sample sizes: 450, 451, 449, 450, 451, 450, ...
Resampling results across tuning parameters:

| lambda | RMSE | Rsquared | MAE |
|---|---|---|---|
| 0e+00 | 480.7583 | 0.8382233 | 361.2641 |
| 1e-04 | 480.7457 | 0.8382312 | 361.2606 |
| 1e-01 | 488.1834 | 0.8345103 | 370.3040 |

RMSE was used to select the optimal model using the smallest value.
The final value used for the model was lambda = 1e-04.
**[1] "test RMSE of registered prediction"**
**[1] 896.3894**

**[1] "Test RMSE on Total count(casual +registered)"**
**[1] 1873.119**

[1] "lm"
[1] " model on casual count"
Linear Regression

500 samples
30 predictor

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 6 times)
Summary of sample sizes: 450, 448, 450, 451, 451, 449, ...
Resampling results:

| RMSE | Rsquared | MAE |
|---|---|---|
| 310.6568 | 0.719439 | 231.7858 |

Tuning parameter 'intercept' was held constant at a value of TRUE
[1] "test RMSE of casual count prediction"
[1] 345.5146
[1] "model on registered model"
[1] "registered count"
Linear Regression

500 samples
29 predictor

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 6 times)
Summary of sample sizes: 450, 451, 450, 449, 450, 451, ...
Resampling results:

| RMSE | Rsquared | MAE |
|---|---|---|
| 481.7545 | 0.8402131 | 362.0418 |

Tuning parameter 'intercept' was held constant at a value of TRUE
[1] "test RMSE of registered prediction"
[1] 896.4213
[1] "Test RMSE on Total count(casual +registered)"
[1] 1842.415

[1] "lasso"
[1] " model on casual count"
The lasso

500 samples
30 predictor

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 6 times)
Summary of sample sizes: 449, 451, 449, 451, 449, 449, ...
Resampling results across tuning parameters:

| fraction | RMSE | Rsquared | MAE |
|---|---|---|---|
| 0.1 | 425.0802 | 0.6214633 | 318.4660 |
| 0.5 | 319.3486 | 0.7073917 | 235.5467 |
| 0.9 | 308.3910 | 0.7224082 | 230.7805 |

RMSE was used to select the optimal model using the smallest value.
The final value used for the model was fraction = 0.9.
[1] "test RMSE of casual count prediction"
[1] 345.5146
[1] "model on registered model"
[1] "registered count"
The lasso

500 samples
29 predictor

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 6 times)
Summary of sample sizes: 449, 450, 450, 451, 448, 450, ...
Resampling results across tuning parameters:

| fraction | RMSE | Rsquared | MAE |
|---|---|---|---|
| 0.1 | 762.3609 | 0.6945451 | 618.3600 |
| 0.5 | 504.8551 | 0.8290416 | 387.7460 |
| 0.9 | 480.1704 | 0.8397033 | 361.3526 |

RMSE was used to select the optimal model using the smallest value.
The final value used for the model was fraction = 0.9.
[1] "test RMSE of registered prediction"
[1] 896.4213
[1] "Test RMSE on Total count(casual +registered)"
[1] 1842.415

[1] "glm"
[1] " model on casual count"
Generalized Linear Model

500 samples
 30 predictor

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 6 times)
Summary of sample sizes: 451, 449, 450, 449, 450, 450, ...
Resampling results:

| RMSE | Rsquared | MAE |
|---|---|---|
| 310.1827 | 0.7214924 | 232.12 |

[1] "test RMSE of casual count prediction"
[1] 345.5146
[1] "model on registered model"
[1] "registered count"
Generalized Linear Model

500 samples
 29 predictor

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 6 times)
Summary of sample sizes: 451, 450, 450, 450, 451, 451, ...
Resampling results:

| RMSE | Rsquared | MAE |
|---|---|---|
| 481.5329 | 0.8393977 | 362.6352 |

[1] "test RMSE of registered prediction"
[1] 896.4213
**[1] "Test RMSE on Total count(casual +registered)"**

**[1] 1842.415**

[1] "enet"
[1] " model on casual count"
Elasticnet

500 samples
30 predictor

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 6 times)
Summary of sample sizes: 450, 451, 450, 452, 449, 449, ...
Resampling results across tuning parameters:

| lambda | fraction | RMSE | Rsquared | MAE |
|---|---|---|---|---|
| 0e+00 | 0.050 | 469.2667 | 0.5926228 | 357.1032 |
| 0e+00 | 0.525 | 320.8617 | 0.7112272 | 234.7054 |
| 0e+00 | 1.000 | 310.0510 | 0.7228402 | 232.2465 |
| 1e-04 | 0.050 | 538.4552 | 0.5456557 | 410.5538 |
| 1e-04 | 0.525 | 336.9844 | 0.6840356 | 242.2628 |
| 1e-04 | 1.000 | 310.0489 | 0.7228449 | 232.2454 |
| 1e-01 | 0.050 | 537.9420 | 0.5491185 | 410.2240 |
| 1e-01 | 0.525 | 336.2158 | 0.6824469 | 242.6476 |
| 1e-01 | 1.000 | 312.5700 | 0.7219067 | 236.3410 |

RMSE was used to select the optimal model using the smallest value.
The final values used for the model were fraction = 1 and lambda = 1e-04.
[1] "test RMSE of casual count prediction"
[1] 345.4918
[1] "model on registered model"
[1] "registered count"
Elasticnet

500 samples
29 predictor

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 6 times)
Summary of sample sizes: 451, 450, 449, 450, 449, 450, ...
Resampling results across tuning parameters:

| lambda | fraction | RMSE | Rsquared | MAE |
|---|---|---|---|---|
| 0e+00 | 0.050 | 869.4870 | 0.6092200 | 714.5591 |

```
 0e+00   0.525      494.4995  0.8324754  379.7170
 0e+00   1.000      481.6494  0.8375684  362.1400
 1e-04   0.050     1089.8830  0.4574527  901.2973
 1e-04   0.525      533.4634  0.8151820  424.0403
 1e-04   1.000      481.6357  0.8375767  362.1313
 1e-01   0.050     1106.2667  0.4154972  913.8777
 1e-01   0.525      590.2589  0.7912315  483.8361
 1e-01   1.000      488.5703  0.8338632  370.7218

RMSE was used to select the optimal model using the smallest value.
The final values used for the model were fraction = 1 and lambda = 1e-04.
[1] "test RMSE of registered prediction"
[1] 896.3894
```

**[1] "Test RMSE on Total count(casual +registered)"**
**[1] 1842.288**

```
[1] "rpart"
[1] " model on casual count"
CART

500 samples
 30 predictor

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 6 times)
Summary of sample sizes: 451, 450, 448, 451, 450, 451, ...
Resampling results across tuning parameters:

 cp          RMSE      Rsquared   MAE
 0.02492299  341.4464  0.6594160  236.1094
 0.02887968  347.6229  0.6441928  244.4587
 0.32414292  479.7115  0.5732009  353.8031

RMSE was used to select the optimal model using the smallest value.
The final value used for the model was cp = 0.02492299.
[1] "test RMSE of casual count prediction"
[1] 449.8056
[1] "model on registered model"
[1] "registered count"
CART

500 samples
 29 predictor
```

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 6 times)
Summary of sample sizes: 451, 450, 450, 450, 449, 448, ...
Resampling results across tuning parameters:

| cp | RMSE | Rsquared | MAE |
|---|---|---|---|
| 0.1169177 | 934.7520 | 0.3953798 | 761.8839 |
| 0.1243528 | 974.0045 | 0.3453282 | 795.0006 |
| 0.3125103 | 1118.7767 | 0.2190088 | 922.2833 |

RMSE was used to select the optimal model using the smallest value.
The final value used for the model was cp = 0.1169177.
[1] "test RMSE of registered prediction"
[1] 1529.423
[1] "Test RMSE on Total count(casual +registered)"
[1] 2313.237

**Python results:**
**1.model_results(Ridge_model)**
test-RMSE of casual user count
305.360522612
coefficient of determination R^2 of the prediction
0.768492435934
test-RMSE of registered user count
539.04674324
coefficient of determination R^2 of the prediction
0.875461206306
**RMSE of total count(registered+casual)**
**699.856823931**
**2.model_results(lasso_model)**
test-RMSE of casual user count
303.540496172
coefficient of determination R^2 of the prediction
0.771243899729
test-RMSE of registered user count
539.262937572
coefficient of determination R^2 of the prediction
0.875361289254
**RMSE of total count(registered+casual)**

**697.458102753**

**3.model_results(lin_reg_model)**
test-RMSE of casual user count
306.027355775
coefficient of determination R^2 of the prediction
0.767480219418
test-RMSE of registered user count
540.929219525
coefficient of determination R^2 of the prediction
0.87458985075
**RMSE of total count(registered+casual)**
**703.99924603**

**4. model_results(rf_model)**
test-RMSE of casual user count
240.452837371
coefficient of determination R^2 of the prediction
0.856451315399
test-RMSE of registered user count
476.859579145
coefficient of determination R^2 of the prediction
0.902538555995
**RMSE of total count(registered+casual)**
**572.848588392**

**5.model_results(DT_model)**
test-RMSE of casual user count
361.070453267
coefficient of determination R^2 of the prediction
0.676314453591
test-RMSE of registered user count
576.441593842
coefficient of determination R^2 of the prediction
0.857582806781
**RMSE of total count(registered+casual)**

**687.942621154**

## PCA results

**R**

**[1] "Princiapl component analysis"**

**[1] "ridge"**

**[1] " model on casual count"**

**Ridge Regression**

550 samples
 25 predictor

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 6 times)
Summary of sample sizes: 495, 495, 495, 496, 496, 495, ...
Resampling results across tuning parameters:

```
  lambda  RMSE          Rsquared   MAE
  0e+00   3.932391e-06  1.0000000  3.126141e-06
  1e-04   5.457959e-03  1.0000000  4.261238e-03
  1e-01   4.903872e+00  0.9999542  3.831106e+00
```

RMSE was used to select the optimal model using the smallest value.
The final value used for the model was lambda = 0.
[1] "test RMSE of casual count prediction"
[1] 4.393815e-06
[1] "model on registered model"
[1] "registered count"
Ridge Regression

550 samples
 25 predictor

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 6 times)
Summary of sample sizes: 495, 494, 494, 496, 496, 495, ...
Resampling results across tuning parameters:

| lambda | RMSE | Rsquared | MAE |
|---|---|---|---|
| 0e+00 | 2.732269e-06 | 1.0000000 | 2.126584e-06 |
| 1e-04 | 1.194283e-02 | 1.0000000 | 9.163671e-03 |
| 1e-01 | 1.072713e+01 | 0.9999523 | 8.238587e+00 |

RMSE was used to select the optimal model using the smallest value.
The final value used for the model was lambda = 0.
[1] "test RMSE of registered prediction"
[1] 3.5235e-06
**[1] "Test RMSE on Total count(casual +registered)"**
**[1] 7.455615e-06**

[1] "Princiapl component analysis"
[1] "lm"
[1] " model on casual count"
**Linear Regression**

550 samples
25 predictor

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 6 times)
Summary of sample sizes: 495, 495, 494, 495, 495, 495, ...
Resampling results:

| RMSE | Rsquared | MAE |
|---|---|---|
| 3.917672e-06 | 1 | 3.119728e-06 |

Tuning parameter 'intercept' was held constant at a value of TRUE
[1] "test RMSE of casual count prediction"
[1] 4.393815e-06
[1] "model on registered model"
[1] "registered count"
Linear Regression

550 samples
25 predictor

No pre-processing

Resampling: Cross-Validated (10 fold, repeated 6 times)
Summary of sample sizes: 494, 494, 496, 495, 495, 494, ...
Resampling results:

| RMSE | Rsquared | MAE |
|---|---|---|
| 2.740865e-06 | 1 | 2.137358e-06 |

Tuning parameter 'intercept' was held constant at a value of TRUE
[1] "test RMSE of registered prediction"
[1] 3.523501e-06
**[1] "Test RMSE on Total count(casual +registered)"**
**[1] 7.455616e-06**

[1] "Princiapl component analysis"
[1] "lasso"
[1] " model on casual count"
**The lasso**

550 samples
25 predictor

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 6 times)
Summary of sample sizes: 496, 494, 495, 495, 495, 494, ...
Resampling results across tuning parameters:

| fraction | RMSE | Rsquared | MAE |
|---|---|---|---|
| 0.1 | 554.35104 | 1 | 428.45249 |
| 0.5 | 307.97231 | 1 | 238.02878 |
| 0.9 | 61.59357 | 1 | 47.60506 |

RMSE was used to select the optimal model using the smallest value.
The final value used for the model was fraction = 0.9.
[1] "test RMSE of casual count prediction"
[1] 70.02412
[1] "model on registered model"
[1] "registered count"
The lasso

550 samples
 25 predictor

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 6 times)
Summary of sample sizes: 495, 494, 494, 494, 496, 495, ...
Resampling results across tuning parameters:

| fraction | RMSE | Rsquared | MAE |
|---|---|---|---|
| 0.1 | 1188.9124 | 1 | 960.9984 |
| 0.5 | 660.5067 | 1 | 533.8878 |
| 0.9 | 132.1010 | 1 | 106.7773 |

RMSE was used to select the optimal model using the smallest value.
The final value used for the model was fraction = 0.9.
[1] "test RMSE of registered prediction"
[1] 229.2209
**[1] "Test RMSE on Total count(casual +registered)"**
**[1] 267.4817**

[1] "Princiapl component analysis"
[1] "glm"
[1] " model on casual count"
**Generalized Linear Model**

550 samples
 25 predictor

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 6 times)
Summary of sample sizes: 497, 494, 494, 495, 495, 495, ...
Resampling results:

| RMSE | Rsquared | MAE |
|---|---|---|
| 3.922186e-06 | 1 | 3.127059e-06 |

[1] "test RMSE of casual count prediction"
[1] 4.393815e-06

[1] "model on registered model"
[1] "registered count"
Generalized Linear Model

550 samples
25 predictor

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 6 times)
Summary of sample sizes: 494, 494, 494, 496, 495, 494, ...
Resampling results:

| RMSE | Rsquared | MAE |
|---|---|---|
| 2.728588e-06 | 1 | 2.127038e-06 |

[1] "test RMSE of registered prediction"
[1] 3.523501e-06
[1] "Test RMSE on Total count(casual +registered)"
[1] 7.455616e-06

[1] "Principl component analysis"
[1] "enet"
[1] " model on casual count"
**Elasticnet**

550 samples
25 predictor

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 6 times)

```
Summary of sample sizes: 495, 494, 495, 495, 494, 496, ...
Resampling results across tuning parameters:

  lambda  fraction  RMSE          Rsquared   MAE         
  0e+00   0.050     5.854962e+02  1.0000000  4.522145e+02
  0e+00   0.525     2.927476e+02  1.0000000  2.261068e+02
  0e+00   1.000     3.935200e-06  1.0000000  3.128242e-06
  1e-04   0.050     5.854954e+02  1.0000000  4.522139e+02
  1e-04   0.525     2.927393e+02  1.0000000  2.261004e+02
  1e-04   1.000     5.459708e-03  1.0000000  4.218047e-03
  1e-01   0.050     5.847381e+02  1.0000000  4.516291e+02
  1e-01   0.525     2.847877e+02  1.0000000  2.199604e+02
  1e-01   1.000     4.905737e+00  0.9999548  3.794031e+00

RMSE was used to select the optimal model using the smallest value.
The final values used for the model were fraction = 1 and lambda = 0.
[1] "test RMSE of casual count prediction"
[1] 4.393815e-06
[1] "model on registered model"
[1] "registered count"
Elasticnet 

550 samples
 25 predictor

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 6 times) 
Summary of sample sizes: 495, 495, 494, 495, 495, 495, ... 
Resampling results across tuning parameters:

  lambda  fraction  RMSE          Rsquared   MAE         
  0e+00   0.050     1.255152e+03  1.0000000  1.014410e+03
  0e+00   0.525     6.275757e+02  1.0000000  5.072049e+02
  0e+00   1.000     2.727754e-06  1.0000000  2.122407e-06
  1e-04   0.050     1.255150e+03  1.0000000  1.014409e+03
  1e-04   0.525     6.275572e+02  1.0000000  5.071900e+02
  1e-04   1.000     1.178293e-02  1.0000000  8.941161e-03
  1e-01   0.050     1.253543e+03  1.0000000  1.013111e+03
  1e-01   0.525     6.106795e+02  1.0000000  4.935670e+02
```

1e-01   1.000     1.058525e+01  0.9999543  8.040639e+00

RMSE was used to select the optimal model using the smallest value.
The final values used for the model were fraction = 1 and lambda = 0.
[1] "test RMSE of registered prediction"
[1] 3.5235e-06
**[1] "Test RMSE on Total count(casual +registered)"**
**[1] 7.455615e-06**

## PCA results
## Python

pca_model_results(Ridge_model)
test-RMSE PCA model
6.87384334901e-06
coefficient of determination R^2 of the prediction
1.0
test-RMSE PCA model
4.4653597263e-06
coefficient of determination R^2 of the prediction
1.0
**RMSE of total count(registered+casual)**
**9.22966398454e-06**

**pca_model_results(lin_reg_model)**
test-RMSE PCA model
4.2846475917e-06
coefficient of determination R^2 of the prediction
1.0
test-RMSE PCA model
3.17754368194e-06
coefficient of determination R^2 of the prediction
1.0
**RMSE of total count(registered+casual)**
**6.11408842905e-06**

**pca_model_results(lasso_model)**
test-RMSE PCA model
0.00159008940721
coefficient of determination R^2 of the prediction
0.999999999994
test-RMSE PCA model
0.000652521213658
coefficient of determination R^2 of the prediction
1.0
RMSE of total count(registered+casual)
0.00198170567784

**pca_model_results(ela_net)**
test-RMSE PCA model
0.00159008721435
coefficient of determination R^2 of the prediction
0.999999999994
test-RMSE PCA model
0.000652521066928
coefficient of determination R^2 of the prediction
1.0
RMSE of total count(registered+casual)
0.00198170349383

pca_model_results(rf_model)
test-RMSE PCA model
11.5971069189
coefficient of determination R^2 of the prediction
0.999666082848
test-RMSE PCA model
43.9180056722
coefficient of determination R^2 of the prediction
0.999173320777
RMSE of total count(registered+casual)
48.4319017912

**Comparing results of the models**
**Results in r(feature selection using ‘Boruta’)**

| **Model** | **R-square of casual count** | **R-square of registered count** | **Test RMSE of total predictions (casual + registered)** |
|---|---|---|---|
| Linear regression | **0.7276002** | **0.8383362** | **1855.027** |
| Ridge regression | **0.7310268** | **0.838133** | **1854.948** |
| Lasso regression | **0.7311360** | **0.8382036** | **1855.027** |
| Elastic net regression | **0.7314965** | **0.8400089** | **1854.972** |
| GLM | **0.7315754** | **0.8380084** | **1855.027** |

**PCA results**

| **Model** | **R-square of casual count** | **R-square of registered count** | **Test RMSE of total predictions (casual + registered)= count** |
|---|---|---|---|
| Linear regression | **1** | **1** | **7.455616e-06** |
| Ridge regression | **1** | **1** | **7.455615e-06** |
| Lasso regression | **1** | **1** | **267.4817** |
| Elastic net regression | **1** | **1** | **7.455615e-06** |
| GLM | **1** | **1** | **7.455616e-06** |

**model results in python**

| **Model** | **R-square of casual count** | **R-square of registered count** | **Test RMSE of total predictions (casual + registered)= count** |
|---|---|---|---|
| Linear regression | **0.767480219418** | **0.87458985075** | **703.99924603** |
| Ridge regression | **0.768492435934** | **0.875461206306** | **699.856823931** |
| Lasso regression | **0.771243899729** | **0.875361289254** | **697.458102753** |
| Elastic net regression | **0.58634102819** | **0.718459994794** | **1002.89775988** |
| Random forest | **0.827661295807** | **0.893328818247** | **591.324006456** |
| | | | |

**PCA results in python**

| **Model** | **R-square of casual count** | **R-square of registered count** | **Test RMSE of total predictions (casual + registered)= count** |
|---|---|---|---|
| Linear regression | **1** | **1** | **6.1145e-06 ~ 0** |
| Ridge regression | **1** | **1** | **9.2299e-06 ~ 0** |
| Lasso regression | **1** | **1** | **0.0019856 ~ 0** |
| Elastic net regression | **1** | **1** | **0.00198170~ 0** |
| Random forest | **0.99961748340** | **0.999315745056** | **44.0029658091** |
| | | | |

**So over all PCA has produced the best results for the above problem**

NOTE:RMSE that I have provided in the above tables is of *** test data***.

RMSE of test data is less than the training data ( the PCA models has achieved good results and it has not overfitted ) with r_squared value of 100%.

## Chapter 7:

### R code

```
#here total count is the sum of registered and casual users
# we have to predict casual and registered users and sum them and
#compare to count(find RMSE on total count)




#loading data

train=read.csv("day.csv")
summary(train)

#encoding factor varibles
train$season  <- factor(train$season, labels = c("Spring", "Summer", "Fall", "Winter"))
train$weathersit <- factor(train$weathersit, labels = c("Good", "Normal", "Bad"))
train$yr  <- factor(train$yr, labels = c(2011,2012))
train$mnth=factor(train$mnth,labels=c('jan','feb','march',
                          'apr','may','jun','jul','aug','sep',
                          'oct','nov','dec'))
train$holiday = factor(train$holiday, labels = c('Yes','no'))
train$weekday=factor(train$weekday,labels=c('sun','mom','tue','wed',
                              'thu','fri','sat'))
train$workingday=factor(train$workingday,labels = c('No','Yes'))
```

```
table(is.na(train)) # There is no missing data in the data set
dat=train

########################
#
#Exploratory data analysis
#
########################

## Understanding the distribution of numerical variables and generating a frequency
table for numeric variables
q<- par(mfrow = c(4,2))
p <- par(mar = rep(2,4))
hist(train$registered)
hist(train$casual)
hist(train$cnt)
#As it is visible from the below figures that "count"
#variable is skewed towards right.
#It is desirable to have Normal distribution as most
#of the machine learning techniques require dependent variable to be Normal.
#One possible solution is to take log transformation on "count" variable after
#removing outlier data points. After the transformation the data looks lot better
#but still not ideally following normal distribution.


hist(train$temp)
hist(train$atemp)
hist(train$hum)
hist(train$windspeed)
#plot a histogram for each numerical variables and analyze the distribution.


library(ggplot2)
ggplot(train, aes(x = season, fill = season)) + geom_bar()
ggplot(train, aes(x = weathersit, fill = weathersit)) + geom_bar()
ggplot(train, aes(x = mnth, fill = mnth)) + geom_bar()
ggplot(train, aes(x = yr, fill = yr)) + geom_bar()
ggplot(train, aes(x = holiday, fill = holiday)) + geom_bar()
ggplot(train, aes(x = workingday, fill = workingday)) + geom_bar()
```

```
ggplot(train, aes(x = weekday, fill = weekday)) + geom_bar()

#continuous varibles

ggplot(data=dat,aes(dat$hum))+geom_histogram(aes(fill=..count..),bins=70)+scale_fill_gradient("Count", low="Steelblue",high = "Red")+labs(title="Histogram for humidity") +labs(x="humidity", y="Count")

ggplot(data=dat,aes(dat$temp))+geom_histogram(aes(fill=..count..),bins=10)+scale_fill_gradient("Count", low="Steelblue",high = "Red")+labs(title="Histogram for temp") +labs(x="temp", y="Count")

ggplot(data=dat,aes(dat$atemp))+geom_histogram(aes(fill=..count..),bins=10)+scale_fill_gradient("Count", low="Steelblue",high = "Red")+labs(title="Histogram for atemp") +labs(x="atemp", y="Count")

ggplot(data=dat,aes(dat$registered))+geom_histogram(aes(fill=..count..),bins = 70)+scale_fill_gradient("Count", low="Steelblue",high = "Red")+labs(title="Histogram for registered users") +labs(x="registered users", y="Count")

ggplot(data=dat,aes(dat$casual))+geom_histogram(aes(fill=..count..),binwidth = 100)+scale_fill_gradient("Count", low="Steelblue",high = "Red")+labs(title="Histogram for casual users") +labs(x="casual users", y="Count")

ggplot(data=dat,aes(dat$windspeed))+geom_histogram(aes(fill=..count..),bins = 60)+scale_fill_gradient("Count", low="Steelblue",high = "Red")+labs(title="Histogram of windspeed") +labs(x="windspeed", y="Count")
ggplot(data=dat,aes(dat$cnt))+geom_histogram(aes(fill=..count..),bins = 60)+scale_fill_gradient("Count", low="Steelblue",high = "Red")+labs(title="Histogram of total count") +labs(x="total count(cas+reg)", y="Count")

#checking for outliers in the data
boxplot(train$registered)
boxplot(train$casual)#outliers detected
boxplot(train$cnt)
```

```
boxplot(train$temp)
boxplot(train$atemp)
boxplot(train$hum)
boxplot(train$windspeed)#outliers detected
library(ggplot2)

#season vs count

s<- par(mfrow = c(2,1))
h <- par(mar = rep(2,1))
par(s)

ggplot(dat, aes(x = season, y = cnt, colour = season)) +
  geom_point( aes(group = season)) +
  #geom_line( aes(group = season)) +
  scale_x_discrete("season") +
  scale_y_continuous("Count") +
  theme_minimal() +
  ggtitle("count vs season") +
  theme(plot.title=element_text(size=18))



#year vs count
ggplot(train, aes(x = yr, y = cnt, colour = yr)) +
  geom_point( aes(group = yr)) +
  #geom_line( aes(group = season)) +
  scale_x_discrete("year") +
  scale_y_continuous("Count") +
  theme_minimal() +
  ggtitle("People rent bikes more in 2012, and much less in 2011.\n") +
  theme(plot.title=element_text(size=18))

#month vs count
ggplot(train, aes(x = mnth, y = cnt, colour = mnth)) +
  geom_point( aes(group = mnth)) +
  #geom_line( aes(group = season)) +
  scale_x_discrete("month") +
  scale_y_continuous("Count") +
```

```
  theme_minimal() +
  ggtitle("total users across different months.\n") +
  theme(plot.title=element_text(size=18))

#weather vs count
ggplot(train, aes(x = weathersit, y = cnt, colour = weathersit)) +
  geom_point( aes(group = weathersit)) +
  #geom_line( aes(group = season)) +
  scale_x_discrete("weathers") +
  scale_y_continuous("Count") +
  theme_minimal() +
  ggtitle("total users across different weathers.\n") +
  theme(plot.title=element_text(size=18))

#weekday vs count
ggplot(train, aes(x = weekday, y = cnt, colour =weekday)) +
  geom_point( aes(group = weekday)) +
  #geom_line( aes(group = season)) +
  scale_x_discrete("days") +
  scale_y_continuous("Count") +
  theme_minimal() +
  ggtitle("total users across different weekdays.\n") +
  theme(plot.title=element_text(size=18))

#working day vs count

ggplot(train, aes(x = workingday, y = cnt, colour = workingday)) +
  geom_point( aes(group = workingday)) +
  #geom_line( aes(group = season)) +
  scale_x_discrete("working day") +
  scale_y_continuous("Count") +
  theme_minimal() +
  ggtitle("total users across workingdays.\n") +
  theme(plot.title=element_text(size=18))

#holiday vs count
ggplot(train, aes(x = holiday, y = cnt,colour=holiday)) +
  geom_point( aes(group = holiday)) +
  #geom_line( aes(group = season)) +
```

```
  scale_x_discrete("holiday") +
  scale_y_continuous("Count") +
  theme_minimal() +
  ggtitle("total users across holidays.\n") +
  theme(plot.title=element_text(size=18))


#boxplot of data
boxplot(dat)
################################
#
#
#
# plotting registered users and casual users across
#different feature
#
#
#
###############################

#season vs registered
sea_reg=ggplot(train, aes(x = season, y = registered,colour=season)) +
  geom_point( aes(group = season)) +
  #geom_line( aes(group = season)) +
  scale_x_discrete("season") +
  scale_y_continuous("registered") +
  theme_minimal() +
  ggtitle("registered users across seasons") +
  theme(plot.title=element_text(size=18))
#season vs casual users
sea_cas=ggplot(train, aes(x = season, y = casual,colour=season)) +
  geom_point( aes(group = season)) +
  #geom_line( aes(group = season)) +
  scale_x_discrete("season") +
  scale_y_continuous("registered") +
  theme_minimal() +
  ggtitle(" casual users across seasons") +
  theme(plot.title=element_text(size=18))
grid.arrange(sea_reg,sea_cas)
```

```
#yr vs registered users
year_reg=ggplot(train, aes(x = yr, y = registered,colour=yr)) +
  geom_point( aes(group = yr)) +
  #geom_line( aes(group = season)) +
  scale_x_discrete("yr") +
  scale_y_continuous("registered") +
  theme_minimal() +
  ggtitle(" registered users across years") +
  theme(plot.title=element_text(size=18))
#yr vs casual users
year_cas=ggplot(train, aes(x = yr, y = casual,colour=yr)) +
  geom_point( aes(group = yr)) +
  #geom_line( aes(group = season)) +
  scale_x_discrete("yr") +
  scale_y_continuous("casual") +
  theme_minimal() +
  ggtitle("casual users across years") +
  theme(plot.title=element_text(size=18))
grid.arrange(year_reg,year_cas)

#month vs registered users
month_reg=ggplot(train, aes(x = mnth, y = registered,colour=mnth)) +
  geom_point( aes(group = mnth)) +
  #geom_line( aes(group = season)) +
  scale_x_discrete("month") +
  scale_y_continuous("registered") +
  theme_minimal() +
  ggtitle(" registered users across months") +
  theme(plot.title=element_text(size=18))
#month vs casual users
month_cas=ggplot(train, aes(x = mnth, y = casual,colour=mnth)) +
  geom_point( aes(group = mnth)) +
  #geom_line( aes(group = season)) +
  scale_x_discrete("month") +
  scale_y_continuous("casual") +
```

```
  theme_minimal() +
  ggtitle("casual users months") +
  theme(plot.title=element_text(size=18))
grid.arrange(month_reg,month_cas)

#weekday vs registered users
weekday_reg=ggplot(train, aes(x = weekday, y = registered,colour=weekday)) +
  geom_point( aes(group = weekday)) +
  #geom_line( aes(group = season)) +
  scale_x_discrete("weekday") +
  scale_y_continuous("registered") +
  theme_minimal() +
  ggtitle(" registered users across weekdays") +
  theme(plot.title=element_text(size=18))
#weekday vs casual users
weekday_cas=ggplot(train, aes(x = weekday, y = casual,colour=weekday)) +
  geom_point( aes(group = weekday)) +
  #geom_line( aes(group = season)) +
  scale_x_discrete("weekday") +
  scale_y_continuous("casual") +
  theme_minimal() +
  ggtitle("casual users across weekdays") +
  theme(plot.title=element_text(size=18))

grid.arrange(weekday_reg,weekday_cas)


#working day vs registered users
workday_reg=ggplot(train, aes(x = workingday, y = registered,colour=workingday)) +
  geom_point( aes(group = workingday)) +
  #geom_line( aes(group = season)) +
  scale_x_discrete("workingday") +
  scale_y_continuous("registered") +
  theme_minimal() +
  ggtitle(" registered users across working days") +
  theme(plot.title=element_text(size=18))
#workingday vs casual users
workday_cas=ggplot(train, aes(x = workingday, y = casual,colour=workingday)) +
  geom_point( aes(group = workingday)) +
```

```
  #geom_line( aes(group = season)) +
  scale_x_discrete("workinng day") +
  scale_y_continuous("casual") +
  theme_minimal() +
  ggtitle("casual users across working day") +
  theme(plot.title=element_text(size=18))

grid.arrange(workday_reg,workday_cas)

#holidayday vs registered users
holiday_reg=ggplot(train, aes(x = holiday, y = registered,colour=holiday)) +
  geom_point( aes(group = holiday)) +
  #geom_line( aes(group = season)) +
  scale_x_discrete("hoiday") +
  scale_y_continuous("registered") +
  theme_minimal() +
  ggtitle(" registered users across holidays") +
  theme(plot.title=element_text(size=18))
#holiday vs casual users
holiday_cas=ggplot(train, aes(x = holiday, y = casual,colour=holiday)) +
  geom_point( aes(group = holiday)) +
  #geom_line( aes(group = season)) +
  scale_x_discrete("holiday") +
  scale_y_continuous("casual") +
  theme_minimal() +
  ggtitle("casual users across holiday") +
  theme(plot.title=element_text(size=18))

grid.arrange(holiday_reg,holiday_cas)

#weather  vs registered users
weather_reg=ggplot(train, aes(x = weathersit, y = registered,colour=weathersit)) +
  geom_point( aes(group = weathersit)) +
  #geom_line( aes(group = season)) +
  scale_x_discrete("weather") +
  scale_y_continuous("registered") +
  theme_minimal() +
  ggtitle(" registered users across different weathers") +
  theme(plot.title=element_text(size=18))
```

```
#weather vs casual users
weather_cas=ggplot(train, aes(x = weathersit, y = casual,colour=weathersit)) +
  geom_point( aes(group = weathersit)) +
  #geom_line( aes(group = season)) +
  scale_x_discrete("weather") +
  scale_y_continuous("casual") +
  theme_minimal() +
  ggtitle("casual users across different weathers") +
  theme(plot.title=element_text(size=18))

grid.arrange(weather_reg,weather_cas)


##########################
#
#
#Feature engineering
#
#
#########################

#correlation plot
library(corrplot)
num=c('windspeed','hum','temp','atemp','casual','registered','cnt')
corr=cor(dat[num])
corrplot(corr,method="number")


#"atemp" is variable is not taken into since "atemp" and "temp" has
#got strong correlation with each other.
#During model building any one of the variable has to be dropped since
#they will exhibit multicollinearity in the data.
cols=c("atemp")
train[cols]=NULL
#as it has many outliers i am replacing values that lie
qn = quantile(train$casual, c( 0.95), na.rm = TRUE)
print(qn)
train$casual[train$casual>qn[1]]=qn[1]
# as casual variable is updated we should update total count varible
```

```
# is sum of casual and registered varible
train$cnt=train$casual+train$registered
```

```
################
#
# creating new feature
#Creating bins for casual count variable based on its relation with
# month column
#
#
###############

library(rpart)
install.packages("rpart.plot")
library(rattle)
library(rpart.plot)
d <- rpart(casual ~ mnth, data = train)
#plotting tree
rpart.plot(d)
#creating new variable according to the graph
train$newcas=0
a=train$mnth=='jan' | train$mnth=='feb' |
  train$mnth=='march' |
  train$mnth=='nov' | train$mnth =='dec'
for (i in(1:731)){

if(a[i]==TRUE){
  train$newcas[train$mnth=='jan' | train$mnth=='feb' |
                 train$mnth=='march' |
                 train$mnth=='nov' | train$mnth =='dec']=1
}else{
  train$newcas=2
}
}
```

```
b=train$mnth=='jan' | train$mnth=='feb' |
  train$mnth=='dec'
for (i in(1:731)){
  if(a[i]==TRUE ){

  if( b[i]==TRUE){

    train$newcas[i]=3
  }else{
    train$newcas[i]=4
  }
  }
  else{}
}

c=train$mnth=='apr' | train$mnth=='oct'
for (i in(1:731)){
  if(a[i]==FALSE ){

    if( c[i]==TRUE){

      train$newcas[i]=5
    }else{
      train$newcas[i]=6
    }
  }
  else{}
}

d1= rpart(registered ~ mnth, data = train)
#plotting the tree
rpart.plot(d1)
#creating new varible according to decision tree
train$reg_mnth=0
d=train$mnth=='jan' | train$mnth=='feb' |
  train$mnth=='march'| train$mnth =='dec'
for (i in(1:731)){

  if(d[i]==TRUE){
```

```
    train$reg_mnth[train$mnth=='jan' | train$mnth=='feb' |
               train$mnth=='march' |
               train$mnth=='nov' | train$mnth =='dec']=1
  }else{
    train$reg_mnth=2
  }
}

e=train$mnth=='jan' | train$mnth=='feb'

for (i in(1:731)){
  if(d[i]==TRUE ){

    if( e[i]==TRUE){

      train$reg_mnth[i]=3
    }else{
      train$reg_mnth[i]=4
    }
  }
  else{}
}

f=train$mnth=='apr' | train$mnth=='nov'
for (i in(1:731)){
  if(d[i]==FALSE ){

    if( f[i]==TRUE){

      train$reg_mnth[i]=5
    }else{
      train$reg_mnth[i]=6
    }
  }
  else{}
}

library(dummies)
dummy_weather=data.frame(dummy(train$weathersit))
```

```
dummy_season=data.frame(dummy(train$season))
dummy_weekday=data.frame(dummy(train$weekday))
dummy_holiday=data.frame(dummy(train$holiday))
dummy_month=data.frame(dummy(train$mnth))
dummy_yr=data.frame(dummy(train$yr))
dummy_workingday=data.frame(dummy(train$workingday))
#removing the factor column
train = subset(train, select = -c(weathersit,season,
                            weekday,holiday,mnth,yr,
                            workingday))
#concatenation dummy variable

train=cbind(train,dummy_holiday,dummy_weather,dummy_month,
        dummy_season,dummy_weekday,dummy_yr,dummy_workingday)

#converting newly created varibles to category  type
train$newcas=as.factor(train$newcas)
train$reg_mnth=as.factor(train$reg_mnth)


#removing instant and dteday columns
train$instant=NULL
train$dteday=NULL

#############################
#
#
# Modelling
#
#
##############################

library(DAAG)
#feature selection using boruta package
library(Boruta)


set.seed(123)
train=train[sample(nrow(train)),]
```

```
library(caret)
#feature selection using boruta
finail.boruta_cas=Boruta(casual~., data = train[,c(1:4,7:40)], doTrace = 2)
selected_features_cas=getSelectedAttributes(finail.boruta_cas, withTentative = F)
formula_cas=as.formula(paste("casual~",paste(selected_features_cas,collapse = "+")))
#feature selection using boruta

finail.boruta_reg=Boruta(registered~., data = train[,c(1:3,5,7:40)], doTrace = 2)
selected_features_reg=getSelectedAttributes(finail.boruta_reg, withTentative = F)
formula_reg=as.formula(paste("registered~",paste(selected_features_reg,collapse =
"+")))

#creating model  using selected features
myfunction_model=function(model){
 print(model)
 #summary of the model


 train_control=trainControl(method = "repeatedcv",
                            number = 10,
                            repeats = 6)
 model_casual= train(casual~.,data=train[1:500,-(5:6)],
                    metric="RMSE", method=model,trControl=train_control)
 print(' model on casual count')
 print(model_casual)
 prediction_cas = predict(model_casual, train[501:731,])
 print('test RMSE of casual count prediction')

 print(RMSE(prediction_cas,train[501:731,4]))
 print('model on registered model')


 model_registered= train(registered~.,data=train[1:500,c(-4,-6)],
                         metric="RMSE", method=model,trControl=train_control)
 print('registered count')
 print(model_registered)
 prediction.registered = predict(model_registered, train[501:731,])
 print('test RMSE of registered prediction')
 print(RMSE(prediction.registered,train[501:731,5]))
```

```
  total_count=prediction_cas+prediction.registered
  print('Test RMSE on Total count(casual +registered)')
  print(RMSE(total_count,train[551:731,6]))
}

#ridge regression
myfunction_model('ridge')

#linear regression

myfunction_model('lm')

#elastic net regression
myfunction_model('lasso')

#generalized linear model
myfunction_model('glm')


#elastic regression

myfunction_model('enet')

#decision tree
myfunction_model('rpart')

#random forest
myfunction_model('rf')

############################
# from the above models even after removing statistically unsignificant variables
#the RMSE is high and R-squared value is very low
############################



####################
#
#
```

```
#Principal component analysis on  registered users
#
#
###################



#divide the new data



#removing casual,total count,new_cas,new_reg
pca.train = train[1:550,c(-4,-6,-7,-8)]
pca.test =train[551:731,c(-4,-6,-7,-8)]
#principal component analysis
prin_comp <- prcomp(pca.train)
#outputs the mean of variables
prin_comp$center

#outputs the standard deviation of variables
prin_comp$scale
dim(prin_comp$x)
biplot(prin_comp, scale = 0)

#compute standard deviation of each principal component
std_dev = prin_comp$sdev

#compute variance
pr_var = std_dev^2
#proportion of variance explained
prop_varex =pr_var/sum(pr_var)
#scree plot
plot(prop_varex, xlab = "Principal Component",
    ylab = "Proportion of Variance Explained",
    type = "b")

#cumulative scree plot
plot(cumsum(prop_varex), xlab = "Principal Component",
    ylab = "Cumulative Proportion of Variance Explained",
```

```
    type = "b")
#add a training set with principal components
train.data = data.frame(registered = pca.train$registered, prin_comp$x)

#we are interested in first 40 PCAs as we have seen from the graph
# and the target variable ,so in total 41(including target variable)
train.data =train.data[,1:26]

#transform test into PCA
test.data=predict(prin_comp, newdata = pca.test)
test.data= as.data.frame(test.data)

#select the first 40 components
test.data=test.data[,1:25]
#linear regression



###################
#
#
#Principal component analysis on casual users
#
#
###################


#removing registered,total count,new_cas,new_reg

pca.train.cas = train[1:550,c(-5,-6,-7,-8)]
pca.test.cas =train[551:731,c(-5,-6,-7,-8)]
#principal component analysis
prin_comp.cas <- prcomp(pca.train.cas)
#outputs the mean of variables
prin_comp.cas$center

#outputs the standard deviation of variables
prin_comp.cas$scale
dim(prin_comp.cas$x)
```

```
biplot(prin_comp, scale = 0)

#compute standard deviation of each principal component
std_dev.cas = prin_comp.cas$sdev

#compute variance
pr_var.cas = std_dev.cas^2
#proportion of variance explained
prop_varex.cas =pr_var.cas/sum(pr_var.cas)
#scree plot
plot(prop_varex.cas, xlab = "Principal Component",
    ylab = "Proportion of Variance Explained",
    type = "b")

#cumulative scree plot
plot(cumsum(prop_varex.cas), xlab = "Principal Component",
    ylab = "Cumulative Proportion of Variance Explained",
    type = "b")
#add a training set with principal components
train.data.cas = data.frame(casual = pca.train.cas$casual, prin_comp.cas$x)

#we are interested in first 40 PCAs as we have seen from the graph
# and the target variable ,so in total 41(including target variable)
train.data.cas =train.data.cas[,1:26]

#transform test into PCA
test.data.cas=predict(prin_comp.cas, newdata = pca.test.cas)
test.data.cas= as.data.frame(test.data.cas)

#select the first 40 components
test.data.cas=test.data.cas[,1:25]
#linear regression

###########################
#
#
# function that will predict casual and registered users and
#sum them and compare to total count(finding RMSE)
#
```

```
#
##############################

#creating model with PCA components

myfunction_pca=function(model){
  print('Princiapl component analysis')
  print(model)
  #summary of the model

  train_control=trainControl(method = "repeatedcv",
                         number = 10,
                         repeats = 6)
  pca_model_casual= train(casual ~.,data=train.data.cas,
               metric="RMSE", method=model,trControl=train_control)
  print(' model on casual count')
  print(pca_model_casual)
  pca.prediction_cas = predict(pca_model_casual, test.data.cas)
  print('test RMSE of casual count prediction')

  print(RMSE(pca.prediction_cas,train[551:731,4]))
  print('model on registered model')
  pca_model_registered= train(registered ~.,data=train.data,
               metric="RMSE", method=model,trControl=train_control)
  print('registered count')
  print(pca_model_registered)
  pca.prediction.registered = predict(pca_model_registered, test.data)
  print('test RMSE of registered prediction')
  print(RMSE(pca.prediction.registered,train[551:731,5]))
  total_count=pca.prediction_cas+pca.prediction.registered
  print('Test RMSE on Total count(casual +registered)')
  RMSE(total_count,train[551:731,6])
}



##############################
#
#
```

```
# Modelling
#
#
##############################

#ridge regression
myfunction_pca('ridge')

#linear regression

myfunction_pca('lm')

#elastic net regression
myfunction_pca('lasso')

#generalized linear model

myfunction_pca('glm')

#elastic regression
myfunction_pca('enet')


#decision tree
myfunction_pca('rpart')

#random forest
myfunction_pca('rf')
```

**Python code:**

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Thu Jul  5 09:30:58 2018

@author: vikramreddy
"""

#here total count is the sum of registered and casual users
# we have to predict casual and registered users and sum them and
#compare to count(find RMSE on total count)


import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
#loading data
train=pd.read_csv('day.csv')
#encoding of factor variables
rule0={1:"Spring", 2:"Summer",3: "Fall",4: "Winter"}
rule1={ 1:"clear",2:"mist",3:"light snow"}
rule2={0:2011,1:2012}
rule3={1:'jan',2:'feb',3:'march',4:'apr',5:'may',6:'jun',7:'jul',8:'aug',9:'sep',
```

```
                    10:'oct',11:'nov',12:'dec'}
rule4={0:'No',1:'Yes'}
rule5={0:'sun',1:'mom',2:'tue',3:'wed',4:'thu',5:'fri',6:'sat'}
train['season']=train['season'].replace(rule0)
train['weathersit']=train['weathersit'].replace(rule1)
train['yr']=train['yr'].replace(rule2)
train['mnth']=train['mnth'].replace(rule3)
train['holiday']=train['holiday'].replace(rule4)
train['workingday']=train['workingday'].replace(rule4)
train['weekday']=train['weekday'].replace(rule5)
dat=train.copy()

train.isnull().any().any()
# There is no missing data in the data set

########################
#
#Exploratory data analysis
#
########################
#histogram of categorical data

sns.countplot(dat['season'], color='orange')
sns.countplot(dat['weathersit'], color='red')
sns.countplot(dat['yr'], color='yellow')
sns.countplot(dat['mnth'], color='blue')
sns.countplot(dat['holiday'], color='pink')
sns.countplot(dat['workingday'], color='green')
sns.countplot(dat['weekday'], color='black')

#checking for outliers in the data(continuous variables)
sns.boxplot(dat['cnt'])
sns.boxplot(dat['temp'])
sns.boxplot(dat['registered'])
sns.boxplot(dat['casual'])
sns.boxplot(dat['atemp'])
sns.boxplot(dat['hum'])
sns.boxplot(dat['windspeed'])
```

```
#distribution of categorical data vs count
sns.boxplot(x='season', y='cnt', data=dat)
sns.boxplot(x='weathersit', y='cnt', data=dat)
sns.boxplot(x='yr', y='cnt', data=dat)
sns.boxplot(x='mnth', y='cnt', data=dat)
sns.boxplot(x='holiday', y='cnt', data=dat)
sns.boxplot(x='weekday', y='cnt', data=dat)
sns.boxplot(x='workingday', y='cnt', data=dat)
##################################
#
#
#
# plotting registered users and casual users across
#different feature
#
#
#
#################################

sns.boxplot(x='season', y='registered', data=dat)
sns.boxplot(x='weathersit', y='registered', data=dat)
sns.boxplot(x='yr', y='registered', data=dat)
sns.boxplot(x='mnth', y='registered', data=dat)
sns.boxplot(x='holiday', y='registered', data=dat)
sns.boxplot(x='weekday', y='registered', data=dat)
sns.boxplot(x='workingday', y='registered', data=dat)



sns.boxplot(x='season', y='casual', data=dat)
sns.boxplot(x='weathersit', y='casual', data=dat)
sns.boxplot(x='yr', y='casual', data=dat)
sns.boxplot(x='mnth', y='casual', data=dat)
sns.boxplot(x='holiday', y='casual', data=dat)
sns.boxplot(x='weekday', y='casual', data=dat)
sns.boxplot(x='workingday', y='casual', data=dat)

###########################
#
```

```
#
#Feature engineering
#
#
##########################

#correlation plot
colormap = plt.cm.RdBu
plt.figure(figsize=(15,15))
plt.title('Pearson Correlation of Features', y=1.0, size=10)
sns.heatmap(train[['cnt','temp',
 'atemp',
 'hum',
 'windspeed',
 'casual',
 'registered',]].corr(),linewidths=0.2,vmax=1.0,
            square=True, cmap=colormap, linecolor='white', annot=True)
#"atemp" is variable is not taken into since "atemp" and "temp" has
#got strong correlation with each other.
#During model building any one of the variable has to be dropped since
#they will exhibit multicollinearity in the data.

train=train.drop(['atemp'],axis=1)

################
#
# creating new feature
#Creating bins for casual casual variable based on its relation with
# month column
#
#
###############
from sklearn.tree import DecisionTreeRegressor, export_graphviz
from sklearn import tree
DT_cas_mnth=DecisionTreeRegressor(max_depth=2)
dat['mnth']=dat['mnth'].astype('category')
DT_cas_mnth.fit(dat['casual'].values.reshape(-1,1),dat['mnth'])
feat_cas=list(train.columns[13:14])
tar_mnth=list(train.columns[4:5])
```

```
#exporting the graph
############
#  stepd to veiw this tree
#1.open the .dot file in text editor
#2.copy all the code
#3.go to the link:http://webgraphviz.com/
#4.paste the code and run
#5.you will geta graph

tree.export_graphviz(DT_cas_mnth,out_file='tr.dot',feature_names =
feat_cas,class_names=tar_mnth)

#on the basis of avove graph i have create a new varible
cas_mnth=pd.Series([])

for i in range(731):
    if(train.iloc[i,13] <= 142):
        cas_mnth[i]=1
    else:
        cas_mnth[i]=2

for i in range(731):
    if(cas_mnth[i]==1):
        if(train.iloc[i,13] <= 12):
            cas_mnth[i]=3
        else:
            cas_mnth[i]=4
    else:
        ''



for i in range(731):
    if(cas_mnth[i]==2):
        if(train.iloc[i,13] <= 253.5):
            cas_mnth[i]=5
        else:
            cas_mnth[i]=6
    else:
        ''
```

```
################
#
# creating new feature
#Creating bins for registered variable based on its relation with
# month column
#
#
###############
DT_reg_mnth=DecisionTreeRegressor(max_depth=2)
DT_reg_mnth.fit(dat['registered'].values.reshape(-1,1),dat['mnth'])
target_mnth=list(dat.columns[4:5])
feat_reg=list(dat.columns[14:15])



tree.export_graphviz(DT_reg_mnth,out_file='tr1.dot',feature_names =
feat_reg,class_names=target_mnth)
############
#  stepd to veiw this tree
#1.open the .dot file in text editor
#2.copy all the code
#3.go to the link:http://webgraphviz.com/
#4.paste the code and run
#5.you will geta graph
###############
reg_mnth=pd.Series([])

for i in range(731):
   if(train.iloc[i,14] <= 2111.5):
      reg_mnth[i]=1
   else:
      reg_mnth[i]=2

for i in range(731):
   if(reg_mnth[i]==1):
      if(train.iloc[i,14] <= 575.0):
         reg_mnth[i]=3
```

```
        else:
            reg_mnth[i]=4
    else:
        ''



for i in range(731):
    if(reg_mnth[i]==2):
        if(train.iloc[i,14] <= 6457.5):
            reg_mnth[i]=5
        else:
            reg_mnth[i]=6
    else:
        ''
#merging newly created varibles to the main data

train=train.merge(cas_mnth.to_frame(), left_index=True, right_index=True)
train=train.merge(reg_mnth.to_frame(), left_index=True, right_index=True)

#replacing the valus that lie greather than 0.95 quantile with the value
#that lie in 0.95 quantile

train.casual.quantile([0.95]) #value is 2355
train.casual=train.casual.mask(train.casual >2355,2355)
#as the casual variable is changed
# so we have to update total count variables as it is sum of casual and registered users
train.cnt=train.casual+train.registered

###########################
#
#
#
# Feature engineering
#
#
###########################
#conveting this variables in to their respective type
train['mnth']=train['mnth'].astype('category')
train['season']=train['season'].astype('category')
```

```
train['weekday']=train['weekday'].astype('category')
train['weathersit']=train['weathersit'].astype('category')
train['workingday']=train['workingday'].astype('category')
train['workingday']=train['workingday'].astype('category')
train['holiday']=train['holiday'].astype('category')
#convertinf newly creaed variables to category type
train['0_y']=train['0_y'].astype('category')
train['0_x']=train['0_x'].astype('category')



################
#
#One hot encoding of factor varibles
#
#
###############

mnth_dummy=pd.get_dummies(train['mnth'])
season_dummy=pd.get_dummies(train['season'])
weekday_dummy=pd.get_dummies(train['weekday'])
weather_dummy=pd.get_dummies(train['weathersit'])
working_dummy=pd.get_dummies(train['workingday'])
holiday_dummy=pd.get_dummies(train['holiday'])
yr_dummy=pd.get_dummies(train['yr'])

holiday_dummy.columns=['no_hol','yes_hol']

#combining one hot encoded column to the main data
train=train.join([mnth_dummy,season_dummy,weekday_dummy,weather_dummy,wo
rking_dummy,holiday_dummy,yr_dummy])
#removing that factor columns for which we have done one hot encoding
train=train.drop(['mnth','season','weekday','weathersit','workingday','instant',
 'dteday','holiday','yr'],axis=1)

   #importing models
from sklearn.tree import DecisionTreeRegressor
from sklearn.linear_model import Ridge, Lasso, LinearRegression,ElasticNet
from sklearn.ensemble import  RandomForestRegressor
```

```
from sklearn.metrics import mean_squared_error

from sklearn.utils import shuffle
#shuffling data
train = shuffle(train)
###################
#
#Feature selection using select K best
#
#
###################
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import f_regression
from sklearn.preprocessing import scale
#data for predicting  casual count
train_cas=train.drop(['casual','registered','cnt'],axis=1)
#target (casual count)
test_cas=train['casual']
##data for predicting  registered count

train_reg=train.drop(['casual','registered','cnt'],axis=1)
##target  (registered count)

test_reg=train['registered']
#selecting top 30 features
train
X_new_cas = SelectKBest(f_regression, k=30).fit_transform(train_cas,test_cas)
X_new_reg = SelectKBest(f_regression, k=30).fit_transform(train_reg,test_reg)

##########################
#
# spliting manually because we are adding predictions(casual+registerd) and compare
to toal count which does not belong
#two components(random state will shuffle data)
###################

#################
#
# casual users
```

```
#
###################



x_train_cas=X_new_cas[:550:1]
x_test_cas=X_new_cas[551::1]
y_train_cas=test_cas.iloc[:550,]
y_test_cas=test_cas.iloc[551:,]

x_train_cas=train_cas.iloc[:550,:]
x_test_cas=train_cas.iloc[551:,:]
y_train_cas=test_cas.iloc[:550,]
y_test_cas=test_cas.iloc[551:,]

#################
#
#registerd users
#
###################

x_train_reg=X_new_reg[:550:1]
x_test_reg=X_new_reg[551::1]
#x_train_reg=train_reg.iloc[:550,:]
#x_test_reg=train_reg.iloc[551:,:]
y_train_reg=test_reg.iloc[:550,]
y_test_reg=test_reg.iloc[551:,]

test_count=train.iloc[551:,5]

#####################
#
#
#creating  function model for casual and registerd users and summing their predictions
#and comparing with total count of test data
#
#
#
#####################
```

```
def model_results(model):

    ###########casual users

   #fitting the data
    model.fit(x_train_cas,y_train_cas)

    #test predictions
    test_predictions_cas=model.predict(x_test_cas)
    #RMSE of test data

    RMSE_cas=np.sqrt(mean_squared_error(y_test_cas, test_predictions_cas))

    print("test-RMSE of casual user count ")
    print(RMSE_cas)
    print('coefficient of determination R^2 of the prediction')
    #model score on test data

    print(model.score(x_test_cas, y_test_cas))


    ######registered users

    #fitting the raw data(with outliers tothe model)
    model.fit(x_train_reg,y_train_reg)

    #test predictions
    test_predictions_reg=model.predict(x_test_reg)
    #RMSE on test data
    RMSE_reg=np.sqrt(mean_squared_error(y_test_reg, test_predictions_reg))
    print("test-RMSE of registered user count ")
    print(RMSE_reg)
    print('coefficient of determination R^2 of the prediction')
    #model score on test data
    print(model.score(x_test_reg, y_test_reg))
```

```
    #summing casual and registered predictions to get total count predictions

    count_predictions=test_predictions_reg+test_predictions_cas

    #finding RMSE on total count(by summing up predictions)
    RMSE_count=np.sqrt(mean_squared_error(count_predictions, test_count))
    print('RMSE of total count(registered+casual)')
    print(RMSE_count)


    return ""
```

```
################
#
#
# PCA on registered and casual
#
#
################
data_cas=train.drop(['registered','cnt','0_y','0_x'],axis=1)
data_reg=train.drop(['casual','cnt','0_y','0_x'],axis=1)
X=data_cas.values
#X=scale(X)
X1=data_reg.values
#X1=scale(X1)
#total count(casual +registered) of test data

test_count=train.iloc[551:,5]


#target variable
target_pca_cas=train['casual']
```

```
target_pca_reg=train['registered']


#passing the total number of components to the PCA
from sklearn.decomposition import PCA

pca_cas = PCA(n_components=36)
pca_reg=PCA(n_components=36)

#fitting the values to PCA
pca_cas.fit(scale(X))
pca_reg.fit(scale(X1))

#pca_digits=PCA(0.99)
#X1_cas = pca_digits.fit_transform(X)
#X1_reg=pca_digits.fit_transform(X1)

#The amount of variance that each PC explained
var_cas= pca_cas.explained_variance_ratio_
var_reg= pca_reg.explained_variance_ratio_


#Cumulative Variance
var1_cas=np.cumsum(np.round(pca_cas.explained_variance_ratio_,
decimals=4)*100)
var1_reg=np.cumsum(np.round(pca_reg.explained_variance_ratio_,
decimals=4)*100)


#graph of the variance

plt.plot(var1_cas)
plt.plot(var1_reg)



##############################
## from the above plot
```

```
#The plot above shows that all components explains around 99% variance in the data set.
#
#############################


#Looking at above plot I'm taking 40 variables
pca_cas = PCA(n_components=25)
pca_reg = PCA(n_components=25)


#now fitting the selected components to the data
pca_cas.fit(X)
pca_cas.fit(X1)


#PCA selected features
X1_cas=pca_cas.fit_transform(X)
X1_reg=pca_reg.fit_transform(X1)

#splitting train and test data

x_train_pca_cas=X1_cas[:550:1]
x_test_pca_cas=X1_cas[551::1]
y_train_pca_cas=target_pca_cas.iloc[:550,]
y_test_pca_cas=target_pca_cas.iloc[551:,]

x_train_pca_reg=X1_reg[:550:1]
x_test_pca_reg=X1_reg[551::1]
y_train_pca_reg=target_pca_reg.iloc[:550,]
y_test_pca_reg=target_pca_reg.iloc[551:,]




#################################
#
#
#creating a function that displays PCA results of their models
```

```
#
#
################################

def pca_model_results(model):



    #fitting training data to the model
    model.fit(x_train_pca_cas,y_train_pca_cas)





    #test predictions
    test_pred_pca_cas=model.predict(x_test_pca_cas)

    #RMSE of test predictions and test data
    RMSE=np.sqrt(mean_squared_error(y_test_pca_cas, test_pred_pca_cas))
    print("test-RMSE PCA model ")
    print(RMSE)


    # Returns the coefficient of determination R^2 of the prediction.
    print('coefficient of determination R^2 of the prediction')
    print(model.score(x_test_pca_cas, y_test_pca_cas))




    #fitting training data to the model
    model.fit(x_train_pca_reg,y_train_pca_reg)





    #test predictions
```

```
    test_pred_pca_reg=model.predict(x_test_pca_reg)

    #RMSE of test predictions and test data
    RMSE=np.sqrt(mean_squared_error(y_test_pca_reg, test_pred_pca_reg))
    print("test-RMSE PCA model ")
    print(RMSE)


    # Returns the coefficient of determination R^2 of the prediction.
    print('coefficient of determination R^2 of the prediction')
    print(model.score(x_test_pca_reg, y_test_pca_reg))


    #summing casual and registered predictions to get total count predictions

    count_predictions=test_pred_pca_reg+test_pred_pca_cas
    RMSE_count=np.sqrt(mean_squared_error(count_predictions, test_count))
    print('RMSE of total count(registered+casual)')
    print(RMSE_count)
    return ""



####################
#
# Regularisation methods
#
####################


#ridge regression
Ridge_model=Ridge()#***
model_results(Ridge_model)
pca_model_results(Ridge_model)


#Linear regression

lin_reg_model=LinearRegression()#****
```

```
model_results(lin_reg_model)
pca_model_results(lin_reg_model)


#Lasso
lasso_model=Lasso()
model_results(lasso_model)
pca_model_results(lasso_model)

#elatic net
ela_net=ElasticNet()
model_results(ela_net)
pca_model_results(ela_net)

####################
#
#Regression trees
#
#
##################

#random forest
rf_model=RandomForestRegressor()
model_results(rf_model)
pca_model_results(rf_model)

#Decision tree regressor
DT_model=DecisionTreeRegressor()
model_results(DT_model)
pca_model_results(DT_model)
```

**References**

1.Towards data science
2.Machine learning by bretty lantz
3.Augmented startup videos
4.Analytics vidhya