**CASE STUDY REPORT**

**IE 7275: Data Mining in Engineering**
**Spring 2019**

# <u>Estimation of Bike Sharing Demand</u>

**Sec 04, Group 22**

Submitted to
Prof Xuemin Jin

Submitted by
Anushka Tak and Chandan Manjunath

## I. Background and Introduction

Bike Sharing Systems is one of the most popular commercial businesses, to rent bicycles where the process of obtaining membership, rental, and bike return is automated via a network of kiosk locations throughout a city. The absolute utility of this commercial business needs methods to anticipate the demand in advance and be able to provide service to customer to maximize commercial profits in the market.

We aim to use data mining techniques to predict the total count of bike rentals on an hourly basis, learning from the market behavior.

### Data Collection

We made an effort to reach out to Boston Bixi to collect Bike rental information of recent years in Boston itself, but couldn't manage to get it. Hence, we decided to go forward with publicly available data from a Washington D.C. based company 'Capital Bike Share' with data spanning two years from 2011.

Our response variable is a sum of bikes predicted to be rented in a particular hour. This anticipation will help the market profit of the commercial business, so that the supply always meets the demand. The problem we aim to address is to optimize the business approach of bike rentals.

### Possible Solution

We intend to use various prediction algorithms to predict a count for a particular hour using our developed model. We are going to implement following algorithms in order to achieve our aim and will evaluate performance using RMSE criterion.

a) Linear Regression
b) Random Forest
c) Extreme Gradient Boosting Regression

Our dataset includes following variables

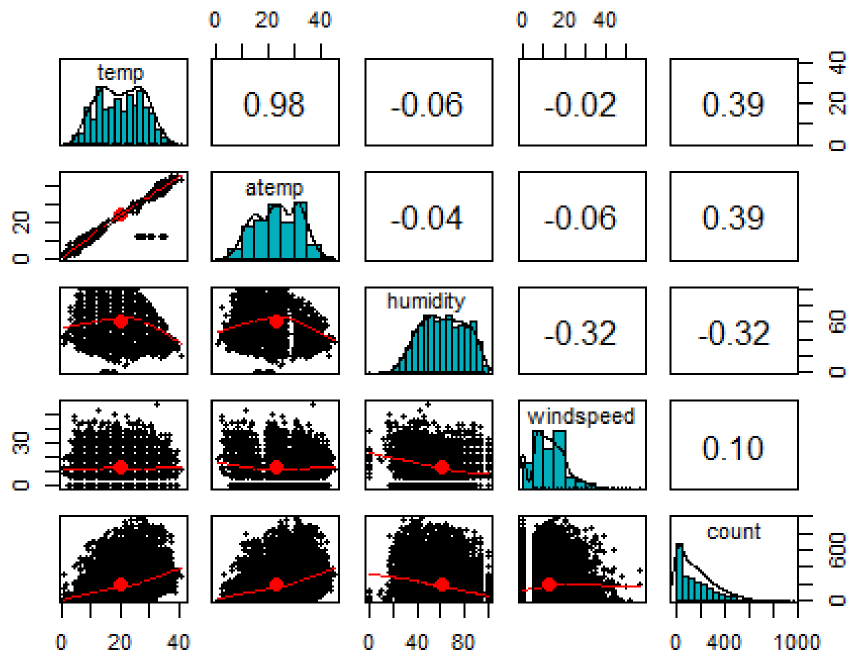
| Predictor | Predictor Description |
|---|---|
| datetime | hourly date + timestamp |
| season | 1 = spring, 2 = summer, 3 = fall, 4 = winter |
| holiday | whether the day is considered a holiday |
| workingday | whether the day is neither a weekend nor holiday |

| | |
|---|---|
| weather | 1: Clear, Few clouds, Partly cloudy, Partly cloudy<br>2: Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist<br>3: Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds<br>4: Heavy Rain + Ice Pallets + Thunderstorm + Mist, Snow + Fog |
| temp | temperature in Celsius |
| atemp | "feels like" temperature in Celsius |
| humidity | relative humidity |
| windspeed | wind speed |
| casual | number of non-registered user rentals initiated |
| registered | number of registered user rentals initiated |
| count | number of total rentals |

## II. Data Exploration and Visualization

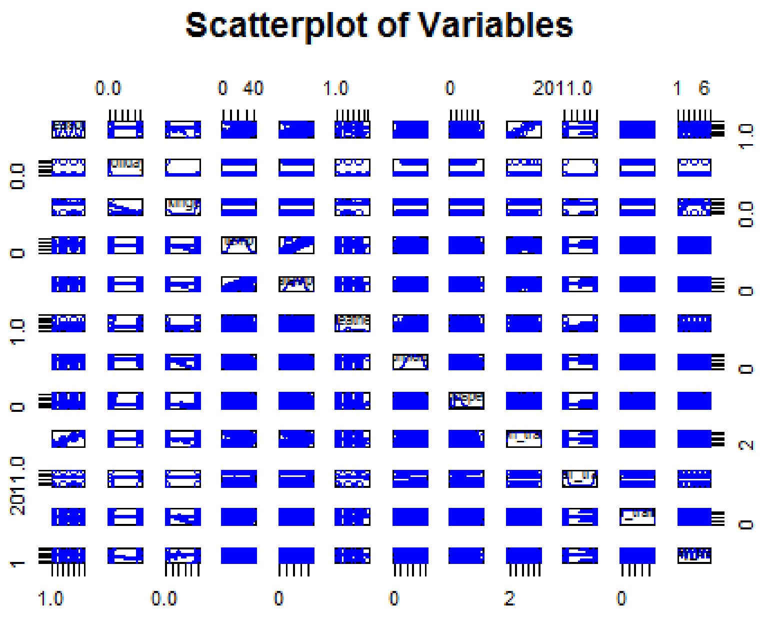
For Data Exploration and Visualization, we have incorporated Domain Knowledge, plotted several graphs, used One Variable and Two Variable Analysis to fetch trends, understand data distribution and Correlations.
There are a few variables which are highly correlated, example, temperature and “feels like” temperature(atemp); and temperature and humidity. This can induce errors in our model in the future(Multicollinearity problems). Therefore, we decide to drop atemp variable.

**Data Distribution**



We observe Normal Distribution for modt of the predictors. For the ones that are skewed, we use Log Transformations(Windspeed, count).

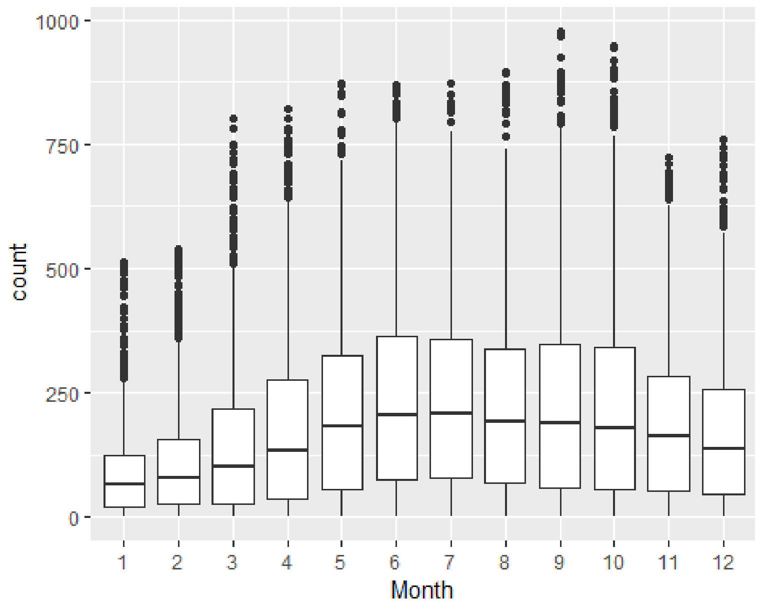
**Correlations**

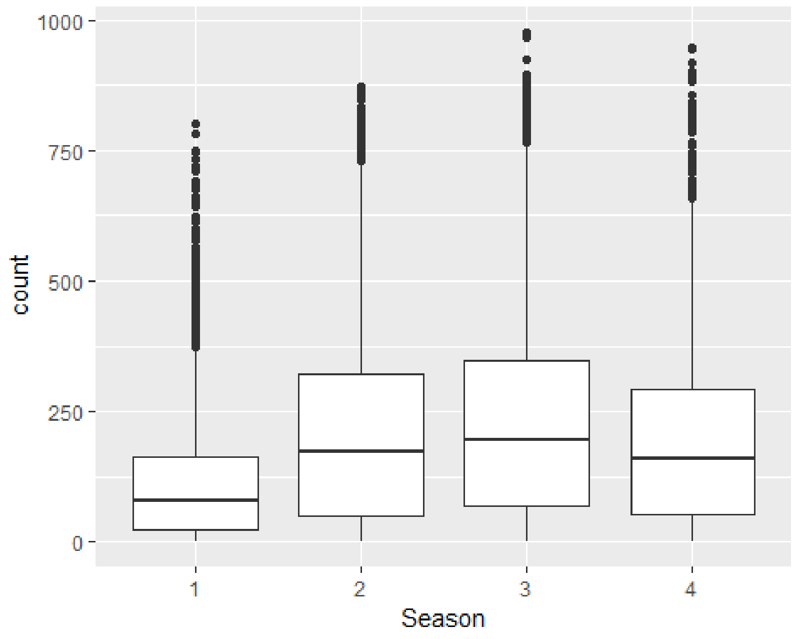


The correlations in the plot are not easily understood by the plot. Therefore, we proceed to One Variable and Two Variable Analysis.

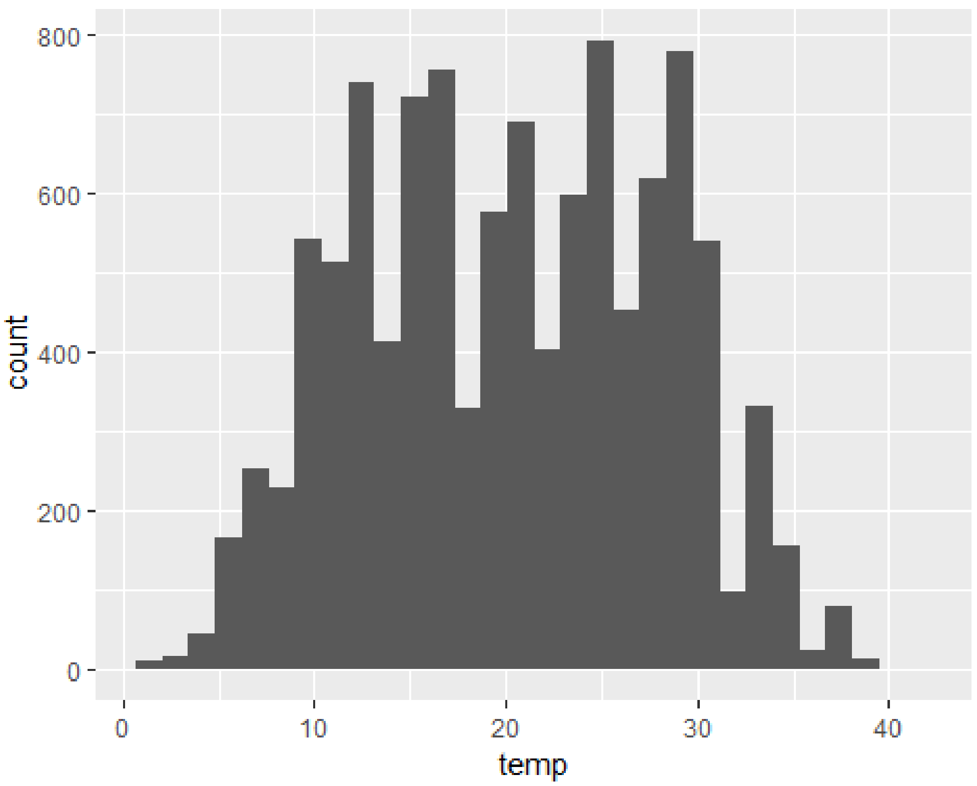
## Trends and Outlier Analysis

From the plot below, we observe the number of bike rented is higher in the months of May to October. The entire distribution, though normal faces a shift upwards. The same trend is observable in season too(higher in Summer and Fall than Winter and Spring) .This is pretty evident from the domain knowledge itself too.

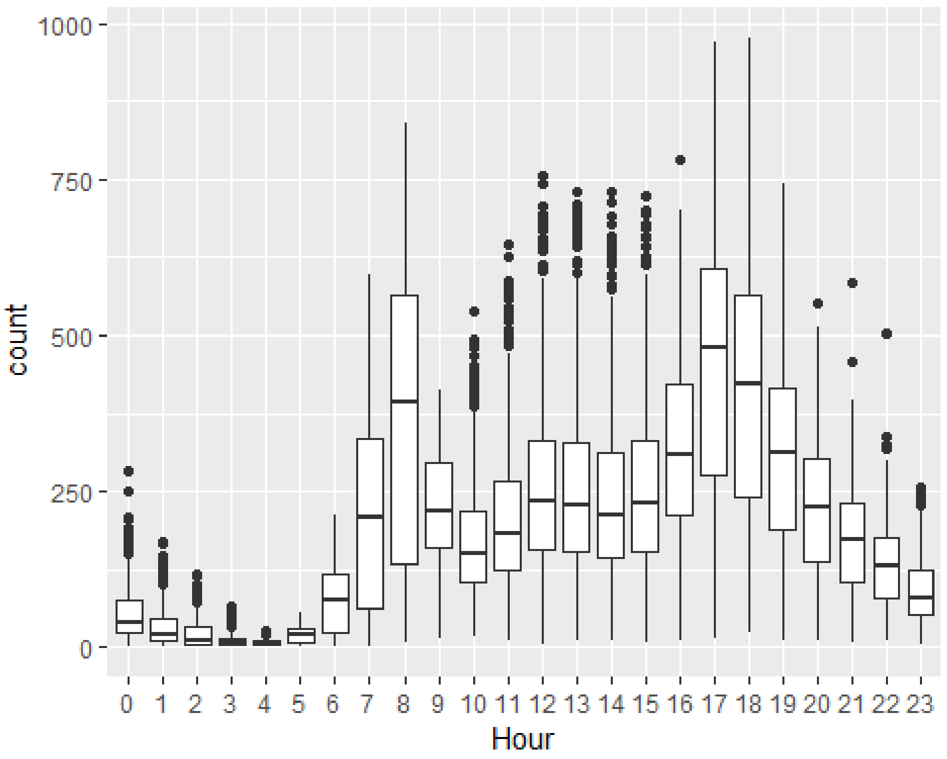




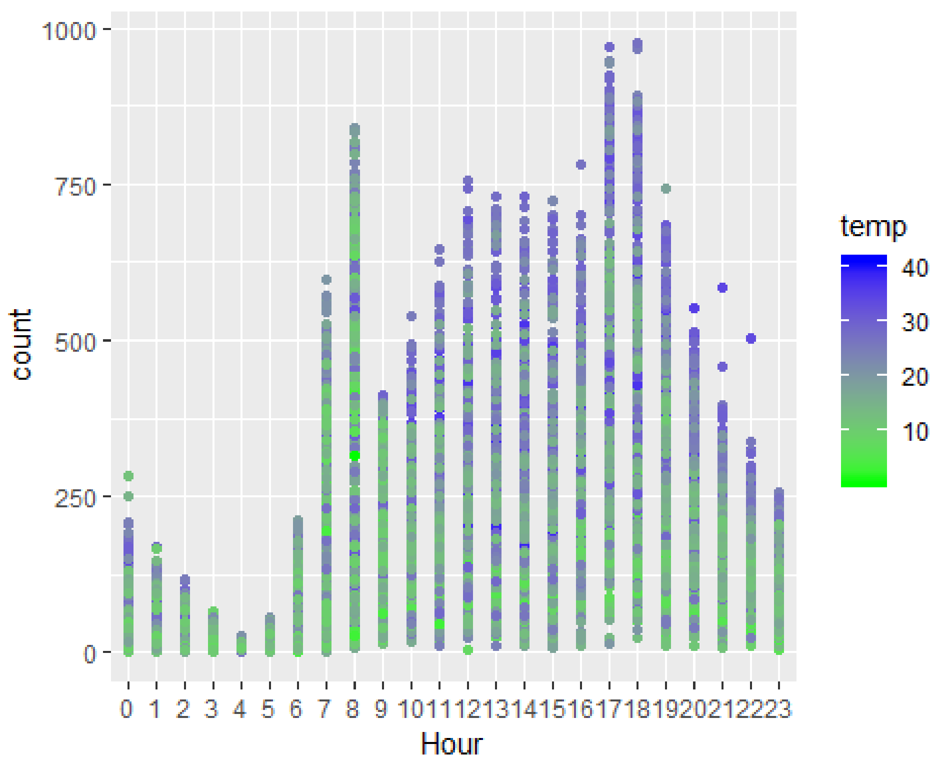
For temperatures, we expect higher bike rentals in pleasant weather and higher temperature readings. This is supported by the plot below.



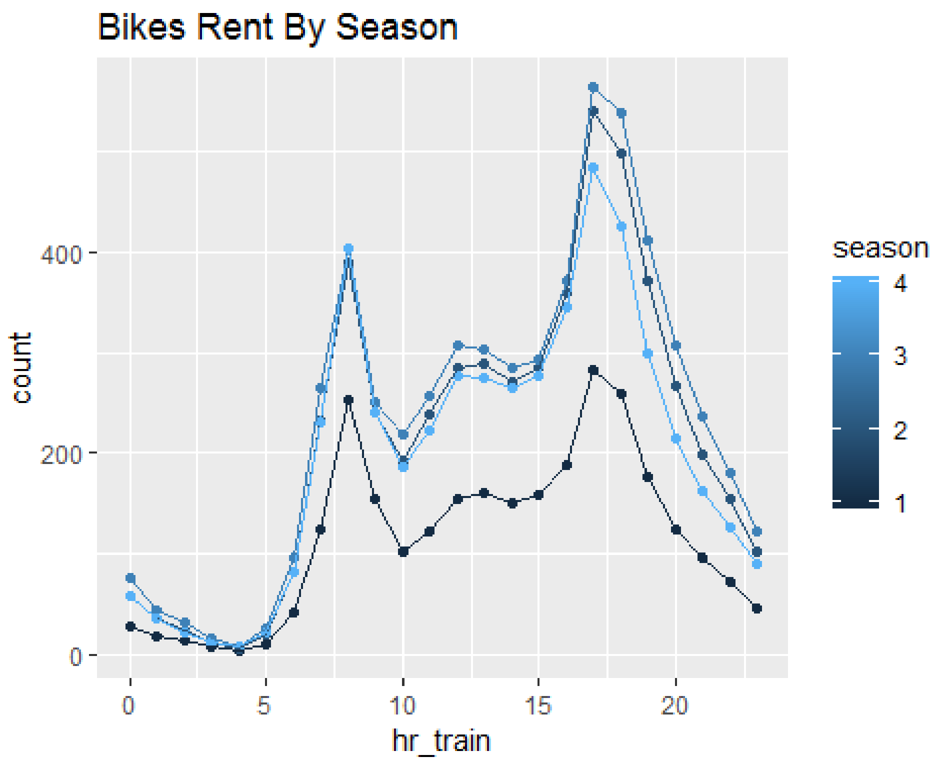
Higher bike rentals are observed in peak hours from 8am-8pm.



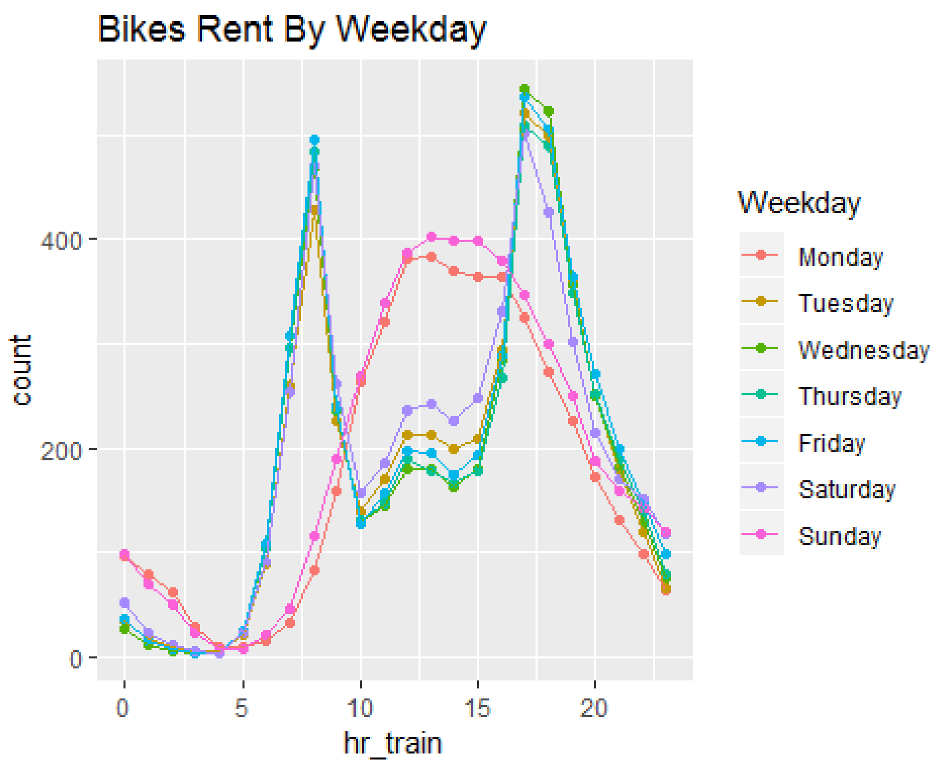
There are higher number of bike rentals at higher temperature during evening hours than in the noon.(Two Variable Analysis)



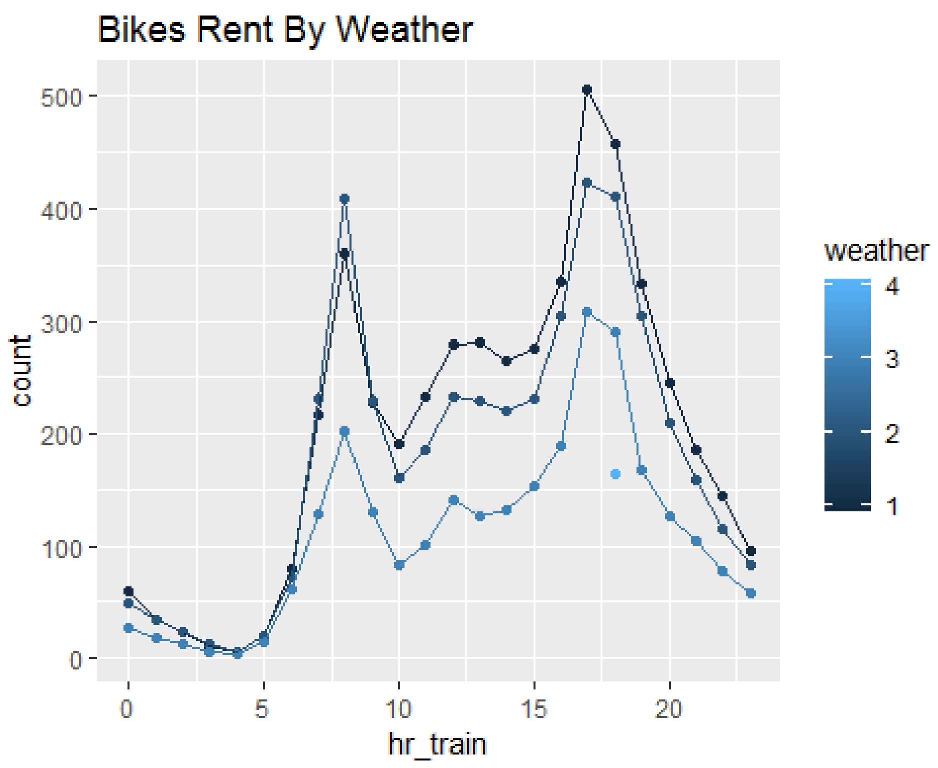
We observe higher bike rentals in morning (from 7-9th hour) and evening(16-19th hour) and People rent bikes more in Fall, and much less in Spring.



We observe higher bike rentals on weekdays during morning and evening; and higher rentals on weekends during daytime.



We observe higher bike rentals in pleasant weather. However, only at 6 pm we observe significant bike rentals even in bad weather.



We have included three predictor graphs to see an influence of each on the other. We observe as temperature drops, bike usage drops with hours the same from the above graph.

## III. Data Preparation and Preprocessing

### Data Summary

The data is fairly clean, doesn't contain any missing values. However, many windspeed values are 0, which can make the model biased. Therefore, we replace zeroes in the windspeed with the mean value. The dataset included datetime stamp which we made cleaner for understanding purposes by splitting into date, hour, minute, second.

### Correlation Analysis

We observe temp and atemp are correlated to each other. Hence, we decide to drop one of these variables, in order to avoid multicollinearity. Moreover, neither gives additional information, hence it is safe to drop one of these variables. Remaining Analysis is included in the aforementioned section.

**Data Transformation**

We needed to transform count variable to avoid the bias led by its skewed distribution. We used log transformation.

**Filtering**

Weekday provides no relevant information for count(from the given boxplot and Stepwise Regression results). We find casual and registered predictor do not provide us with any relevant information. Hence, we drop them.

**For variable selection**, we've used Stepwise Regression and chose AIC and ANOVA results to find significant variables. Results are shown in the document. But most important variables observed were Temperature, Humidity, Hour, Month, Season, and Workingday. All these variables seem intuitive from the domain knowledge too. Additionally, Anova and Scatterplot Matrices tempt us to include windspeed, workingday to our model.

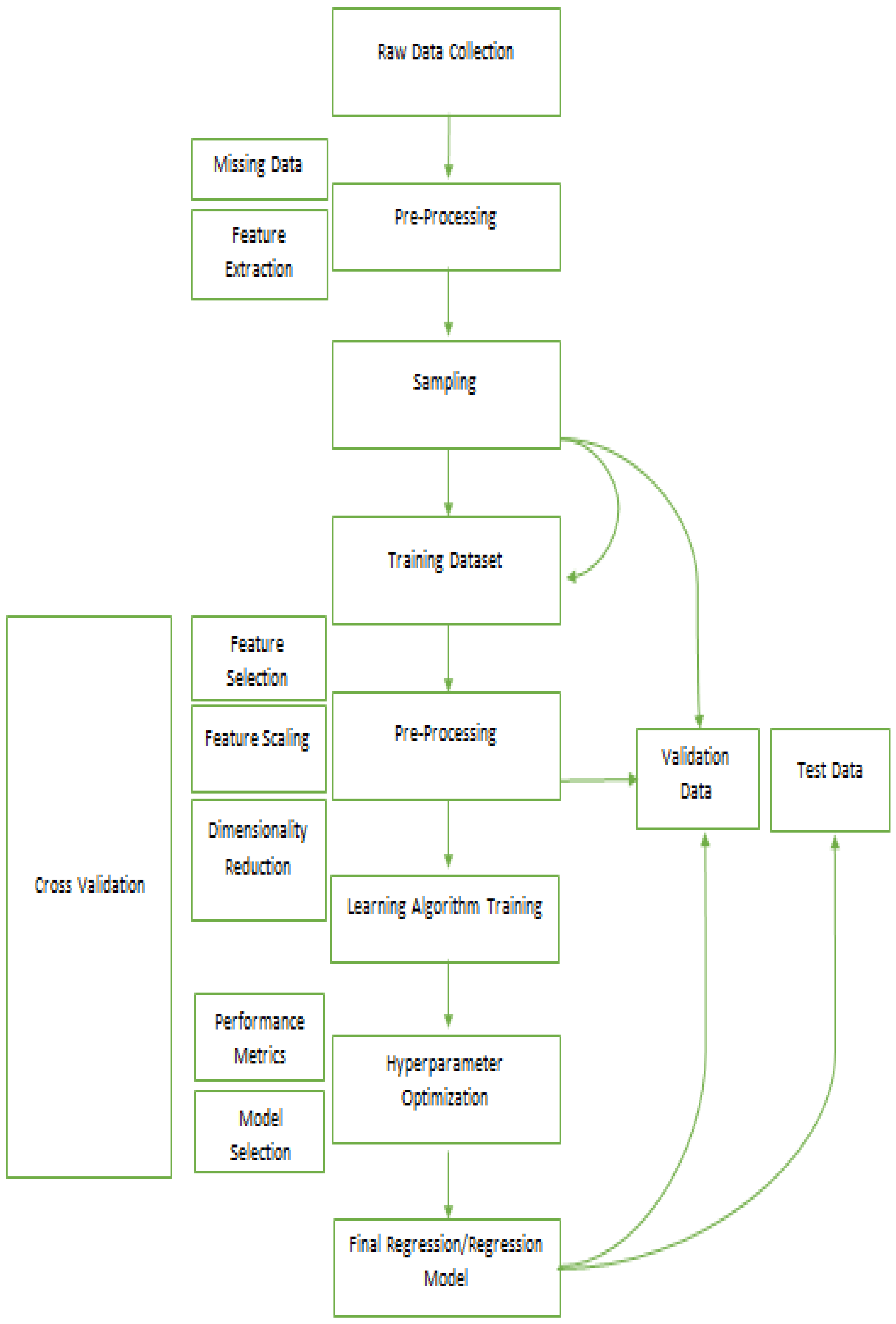
Since Lasso has the ability to select predictors, we seek to use it for the same. From performing the Lasso Regression, we found hour, temp, humidity, and season and to be the most influential predictors in our model.

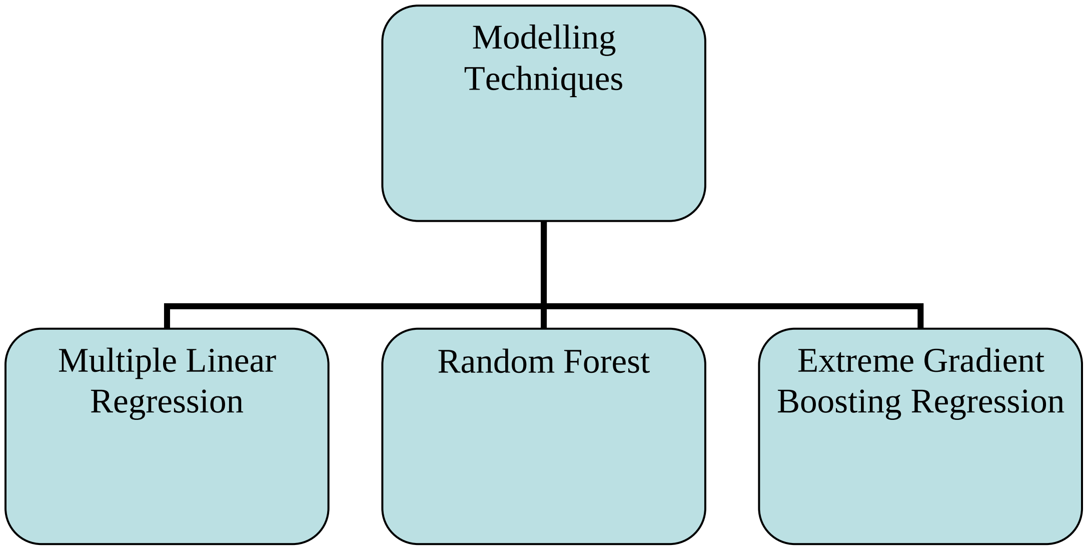
**PCA Analysis**

Most of the predictors in our dataset are categorical, therefore it is not a good idea to use PCA Analysis.

## IV. Data Mining Techniques and Implementation

The case study we've chosen is a Regression Problem, for which Linear Regression is the natural first step. Recognizing multiple non-linear relationships present in our model, we selected Decision trees for Regression, then Random Forest as an ensemble step further ahead of regression Trees. This was followed by a boosting technique in Random Forests, called Gradient Boosting Regression. To supplement our learning, we explored and implemented an advanced technique of Gradient Boosting Regression, which is called Extreme Gradient Regression. It is further explained in the following sections.


Raw Data Collection
Missing Data
Feature Extraction
Pre-Processing
Sampling
Training Dataset
Cross Validation
Feature Selection
Feature Scaling
Dimensionality Reduction
Pre-Processing
Validation Data
Test Data
Learning Algorithm Training
Performance Metrics
Model Selection
Hyperparameter Optimization
Final Regression/Regression Model


Modelling Techniques
Multiple Linear Regression
Random Forest
Extreme Gradient Boosting Regression

## V. Performance Evaluation

For the modeling phase, we split the dataset further into validation and test in the ratio Training(50), Validation(20) and Test(30). We have used attributes in the final model and used several evaluation measures such as RMSE and Adjusted R-sqaure to get an estimate of model's perfomance. As the problem is Regression in nature, we chose RMSE values to evaluate the performance of different models on the same scale. Validation dataset was used to fine tune some parameters in the modeling phase, and for evaluation of the model's performance test data was used. Both Validation and test dataset were unseen during Training phase.

| Algorithm | RMSE |
|---|---|
| Linear Regression | 1.231438 |
| Random Forest | 0.031084 |
| Extreme Gradient Boosting Regression | 0.007105 |

In conclusion, we think Extreme Gradient Boosting Regression is the best choice for our Case study. This is possibly because XGBR being an advanced data science technique gives the best performance. Gradient boosting is an approach where new models are created that predict the residuals or errors of prior models and then added together to make the final prediction. It is called gradient boosting because it uses a gradient descent algorithm to minimize the loss when adding new models.

## VI. Discussion and Recommendation

We started with Linear Regression model since it is the most intuitive to implement and understand. Even if the fit wasn't ideal, we could get an approximate nature of the relationship among two variables. However, we observed low accuracy and accounted non-linear relationship of some predictors with the outcome variable for the same. We realized we should use Cross Validation and other techniques to further refine our modeling. Linear Regression, being sensitive to anomalies could have affected the model's performance.

Therefore, we proceeded to run Random Forest on our model. This was because Decision Trees are more interpretable and flexible. To overcome Overfitting and have less variance, we implemented Random Forest, which is an advanced modification for Decision Trees. As can be seen from the table, the error dropped from 1.23 to 0.03. Since our dataset is only ~12k records, we didn't face computational problems (like delay or extensive use of resources).

Furthermore, we thought of applying advanced techniques and improve Prediction accuracy of Random Forests. Gradient Boosting Regression is a machine learning technique that creates a strong learner from an ensemble of weak learners. This is a Boosting technique over Random Forests. In order to increase the performance, the technique itself optimizes the loss function. We have included the iterative errors on the model. Extreme Gradient Boosting Regression technique is one step ahead of Gradient Boosting Regression and even stronger. In XGBoost the trees can have a varying number of terminal nodes and left weights of the trees that are calculated with less evidence is shrunk more heavily. Newton Boosting uses Newton-Raphson method of approximations which provides a direct route to the minima than gradient descent. The extra randomisation parameter can be used to reduce the correlation between the trees to make a better ensemble. In addition to this, XGBoost is faster than gradient boosting. This is a relatively new machine learning technique that we thought of trying to implement and obtained a smaller error of 0.007105.

***Challenges***

One of the challenges of the project was too many Categorical data. All our initial models struggled to capture relationships correctly. This is why, our Case study required extensive Data Preprocessing.

We also observed many predictors having nonlinear relationships with the outcome variable. This was handled by using transformations and complex classification models.

***Recommendations***

During the course of our Project, we learnt ensemble techniques and Cross Validation techniques greatly increase the model's performance. Tweaking a few parameters in each model can vary its performance. Therefore, in the future, we would like to try more complex models such as neural networks to provide better fits for our data. In addition to this, we would recommend using the auto encoding of H2O library on top of our XGB model. This would even further boost its performance because encoding in H2O library handles Categorical features. Since our dataset is largely categorical based, it may be a useful technique.

## VII. Summary

Throughout our case study, we observed how important is Data Preprocessing, the variables need to be transformed appropriately to be made use of in the later stages. Through exploratory analysis on the data about bike sharing rental counts, we discovered that hour of the day and temperature are the two most important factors that drives the demand of bike sharing rental. For the same, we used multiple techniques such as Stepwise Regression, statistical methods like analyzing ANOVA, Data Visualizationa and Lasso Regression.

For our Modelling phase, we performed comparative analysis of several Data Mining Algorithms, analysed their results, observed their shortcmoings and advantages over others, and reasoned them with our Case Study Problem Statement. Using Linear Regression, Random Forest and Extreme Gradient Boosting Regression, we successfully predited the count of bikes rented on an hourly basis given the weather conditions, time and several other factors. We believe the final model correctly identifies the relationships among the predictors and anticipates the demand of bikes to a fair degree.

## Appendix: R Code for use case study

## Exploratory Data Analysis

### *Data Distribution*

Libraries

```
library(MASS)
library(ggplot2)
library(car)
library(caret)
library(fastDummies)
library(psych)
library(dplyr)
library(tidyverse)
library(lubridate)
library(dlookr)
library(sqldf)
library(glmnet)
library(randomForest)
library(data.table)
library(mlr)
library(xgboost)
library(mvtnorm)
library(parallelMap)
library(ISLR)
```

Data Loading

```
train<-
read_csv("C:/Users/pc/Desktop/Spring2019/DM/dm_project/train.csv")
  test<-
read_csv("C:/Users/pc/Desktop/Spring2019/DM/dm_project/test.csv")
  head(train,n=5)

##         #         A         tibble:         5         x         12
##   datetime                  season holiday workingday weather  temp atemp
##   <dttm>                     <int>   <int>      <int>   <int> <dbl> <dbl>
## 1 2011-01-01 00:00:00          1       0          0       1  9.84  14.4
## 2 2011-01-01 01:00:00          1       0          0       1  9.02  13.6
## 3 2011-01-01 02:00:00          1       0          0       1  9.02  13.6
## 4 2011-01-01 03:00:00          1       0          0       1  9.84  14.4
## 5 2011-01-01 04:00:00          1       0          0       1  9.84  14.4
##  #  ...  with  5  more  variables:  humidity  <int>,  windspeed  <dbl>,
## #   casual <int>, registered <int>, count <int>

  time_train                 <-                  ymd_hms(train$datetime)
  year_train<-year(time_train)
  mon_train<-month(time_train)
  hr_train<-hour(time_train)
  minute_train<-minute(time_train)
  sec_train<-second(time_train)
  wday_train<-wday(time_train                                          )
  date_train<-date(time_train)

  time_test                  <-                    ymd_hms(test$datetime)
  year_test<-year(time_test)
  mon_test<-month(time_test)
  hr_test<-hour(time_test)
  minute_test<-minute(time_test)
  sec_test<-second(time_test)
  wday_test<-wday(time_test                                            )
 date_test<-date(time_test)

train_new<-
transmute(train,season,holiday,workingday,temp,atemp,weather,humidity,w
indspeed,date_train,year_train,mon_train,hr_train,minute_train,sec_trai
n,wday_train,count)
test_new<-
transmute(test,season,holiday,workingday,temp,atemp,weather,humidity,wi
ndspeed,date_test,year_test,mon_test,hr_test,minute_test,sec_test,wday_
test)

scatterplotMatrix(~season+holiday+workingday+temp+atemp+weather+humidit
y+windspeed+mon_train+year_train+hr_train+wday_train,    data=train_new,
   main="Scatterplot of Variables")
```

```
attach(train_new)
cor(train_new[c("temp","atemp","humidity","windspeed","count")])

##                  temp       atemp    humidity   windspeed      count
## temp       1.00000000  0.98494811 -0.06494877 -0.01785201  0.3944536
## atemp      0.98494811  1.00000000 -0.04353571 -0.05747300  0.3897844
## humidity  -0.06494877 -0.04353571  1.00000000 -0.31860699 -0.3173715
## windspeed -0.01785201 -0.05747300 -0.31860699  1.00000000  0.1013695
## count      0.39445364  0.38978444 -0.31737148  0.10136947  1.0000000

detach(train_new)
```

## Data Preprocessing

Substituting 0 with mean values (Windspeed)

```
train_new[!complete.cases(train_new),]

diagnose(train_new)

pairs.panels(train_new[,c(-1,-2,-3,-6,-9,-10,-11,-12,-13,-14,-
15)],method="pearson",hist.col = "#00AFBB",density=TRUE)
```

```
p<-mean(train_new$windspeed)
if (as.factor(train_new$windspeed)==0){train_new$windspeed<-p}
```

### *Data Analysis/ Trends/ Outliers*

## One and Two variable Analysis

```
ggplot(train_new)+geom_boxplot(aes(x=factor(mon_train),y=count))+labs(x
="Month")
```

```
ggplot(train_new)+geom_histogram(aes(x=temp))
```

```
ggplot(train_new)+geom_boxplot(aes(x=factor(season),y=count))+labs(x="S
eason")
```

```
ggplot(train_new)+geom_boxplot(aes(x=factor(hr_train),y=count))+labs(x=
"Hour")
```

```
ggplot(train_new)+geom_point(aes(x=factor(hr_train),y=count,color=temp)
)+labs(x="Hour")+scale_color_gradient(high="blue",low="green")
```

```
ggplot(train_new)+geom_bar(aes(x=mon_train,fill=as.factor(season)))+the
me_bw()+labs(x="months")
```

```
ggplot(train_new)+geom_point(aes(y=mon_train,x=as.factor(season)))+labs
(x="Season",y="Month")
```

```
ggplot(train_new)+geom_bar(aes(x=wday_train,fill=as.factor(workingday))
)+theme_bw()+labs(x="Days")
```

```
ggplot(train_new)+geom_point(aes(y=wday_train,x=as.factor(workingday)))
+labs(x="Season",y="Month")
```

```
season_summary_by_hour <- sqldf('select season, hr_train, avg(count) as
count     from     train_new     group     by     season,     hr_train')
ggplot(train_new,      aes(x=hr_train,      y=count,      color=season))+
geom_point(data   =   season_summary_by_hour,   aes(group   =   season))+
geom_line(data   =   season_summary_by_hour,   aes(group   =   season))+
ggtitle("Bikes Rent By Season")
```

```
a<-as.factor(wday_train)
train_new<-mutate(train_new,a)
day_summary_by_hour <- sqldf('select a, hr_train, avg(count) as count
from          train_new          group          by          a,hr_train')

ggplot(train_new,                              aes(x=hr_train,y=count,
color=a))+geom_point(data=day_summary_by_hour,       aes(group       =
a))+geom_line(data      =      day_summary_by_hour,      aes(group      =
a))+ggtitle("Bikes Rent By Weekday")+ scale_colour_hue('Weekday',breaks
=                                              levels(train_new$a),
labels=c('Monday','Tuesday','Wednesday','Thursday','Friday','Saturday',
'Sunday'))
```

```
weather_summary_by_hour <- sqldf('select weather, hr_train, avg(count)
as    count    from    train_new    group    by    weather,    hr_train')
ggplot(train_new,                aes(x=hr_train,                y=count,
color=weather))+geom_point(data = weather_summary_by_hour, aes(group =
weather))+geom_line(data   =   weather_summary_by_hour,   aes(group   =
weather))+ggtitle("Bikes Rent By Weather")
```

```
fit<-
lm(log(count)~season+holiday+workingday+temp+humidity+atemp+weather+hr_
train+windspeed+mon_train,data=train_new)
step<-stepAIC(fit,direction = "both")

step$anova
```

We observe temp and atemp are correlated to each other. Hence, we decide to drop one of these variables, in order to avoid multicollinearity. Moreover, neither gives additional information, hence it is safe to drop one of these variables.

Dropping useless variables that provide no information

```
#date_time,                                          minute_train,sec_train

train_new<-train_new[,-c(9,13,14)]
test_new<-test_new[,-c(9,13,14)]
```

Weekday provides no relevant information for count(from the given boxplot and Stepwise Regression results). Hence,we drop it.

```
ggplot(data=train_new)+geom_bar(aes(x=wday_train))
```

PCA

Most of the predictors in our dataset are categorical, therefore it is not a good idea to use PCA Analysis.

## Lasso

Since Lasso has the ability to select predictors, we seek to use it for the same.

```
test_ypred = lass.predict(test_x)

(np.sum((test_ypred-test_y)**2)/len(test_y))**0.5

regr = RandomForestRegressor(max_depth=2, random_state=0,n_estimators=100)

regr = regr.fit(train_x, train_y)

r = regr.predict(test_x)

(np.sum((r-test_y)**2)/len(test_y))**0.5
```

### *Modelling*

Method 1

## Linear Regression

```
model1<-
lm(count~season+workingday+humidity+windspeed+temp+hr_train+mon_train,d
ata=train_new)
summary(model1)

##  Residual  standard  error:  147.6  on  10879  degrees  of  freedom
##  Multiple  R-squared:    0.3363,  Adjusted  R-squared:    0.3359
## F-statistic: 918.8 on 6 and 10879 DF,  p-value: < 2.2e-16
```

Creating Dummy variable for season, weather, month, hour

```
train_new1<-
dummy_cols(train_new,select_columns=c("season","weather","mon_train","h
r_train"),remove_first_dummy=TRUE)
#View(train_new1)
train_new1<-train_new1[-c(1,2,5,6,9,10,11,12)]

smp_size<-floor(0.6*nrow(train_new1))
set.seed(123)
train_ind<-sample(seq_len(nrow(train_new1)),size=smp_size)
train_new1<-train_new1[train_ind,]
valid<-train_new1[-train_ind,]

model1_1<-lm(count~.,data=train_new1)



##  Residual  standard  error:  109.5  on  6484  degrees  of  freedom
##  Multiple  R-squared:    0.6367,  Adjusted  R-squared:    0.6342
## F-statistic: 247.1 on 46 and 6484 DF,  p-value: < 2.2e-16

model1_1<-lm(log(count+1)~.,data=train_new1)
summary(model1_1)

##  Residual  standard  error:  0.624  on  6484  degrees  of  freedom
##  Multiple  R-squared:    0.8065,  Adjusted  R-squared:    0.8051
## F-statistic: 587.6 on 46 and 6484 DF,  p-value: < 2.2e-16

fit_LR<-predict(model1_1,valid,se.fit=TRUE)
```

For training dataset, After creating dummy variables, we got an improved R-square of 63% from 45%. However, even this is an unacceptable model accuracy. This is possibly because not all predictors have a linear relationship with the outcome variable ‘count’. For instance, humidity has an inverse relationship; the more humid the less bike rentals. Therefore, we switch to other method ‘Random Forest Regression’

However, by subjecting the response variable to log transformation, we observe much better R-square value of 99.85%.

## Random Forest

```
train_new1$log_count<-log(train_new1$count)
set.seed(415)
fit34         <-         randomForest(log(count+1)         ~         .,
data=train_new1,importance=TRUE,eval_metric="rmse",    ntree=250,mtry=7)
fit34

##
##                                                                 Call:
##   randomForest(formula = log(count + 1) ~ ., data = train_new1,
importance = TRUE, eval_metric = "rmse", ntree = 250, mtry = 7)
##                                Type of random forest: regression
##                                             Number of trees: 250
##     No.     of     variables     tried     at     each     split:     7
##
##                          Mean of squared residuals: 0.02638778
##                         % Var explained: 98.68

attach(valid)
count<-(count+1)
log_count<-log(count)
fit_RF<-predict(fit34,valid,se.fit=TRUE)
rmse<-sqrt(mean(fit_RF-log_count)^2)
  rmse

## [1] 0.01613513

detach(valid)
```

Gradient Boosting Regression

```
cv.ctrl    <-    trainControl(method    =    "repeatedcv",number    =    5,
                        verboseIter=T,
                        classProbs=                                    F)

xgb.grid         <-         expand.grid(nrounds         =         666,
                        max_depth                 =                 c(5),
                        eta                 =                       c(0.1),
                        gamma               =                       c(0.01),
                        colsample_bytree          =                 c(0.75),
                        min_child_weight           =                 c(0),
                        subsample                 =                  c(0.9))

train_new12<-train_new1[,-5]


p<-as.matrix(train_new1)



 q<-as.matrix(valid)
 dtrain <- xgb.DMatrix(data = p,label = train_new1$count)

 dtest <- xgb.DMatrix(data = q,label=count)



  params  <-  list(booster  =  "gbtree",  eta=0.3,  gamma=0,  max_depth=6,
min_child_weight=1, subsample=1, colsample_bytree=1)



  xgbcv <- xgb.cv( params = params, data = dtrain,nrounds = 100, nfold
= 5, showsd = T, stratified = T, print.every.n = 10, early.stop.round =
20, maximize = F)
```

Auto encoding(H2O)

```
#           train_new1<-train_new1[complete.cases(train_new),          ]
#    LR    <-    train(log(count+1)    ~    .,    data    =    train_new1,
#                                                   method="glmnet_h2o",
#                                                      stratified=TRUE,
#                                                      verbose    =   1,
#                                                   eval_metric="rmse",
#                                                trControl   =   cv.ctrl)
#
#    LR3    <-    train(log(count+1)    ~    .,    data    =    train_new1,
#                                                         method="pcr",
#                                                      stratified=TRUE,
#                                                      verbose    =   1,
#                                                   eval_metric="rmse",
#                                                trControl   =   cv.ctrl)
#
#                                                                   LR3

#    RF    <-    train(log(count+1)    ~    .,    data    =    train_new1,
#                                                          method="rf",
#                                                      stratified=TRUE,
#                                                      verbose    =   1,
#                                                   eval_metric="rmse",
#                                                trControl   =   cv.ctrl)
# RF
```