HW4: Nonlinear and Ensemble Modeling (due 12/12 Monday 23:30)

Problem 1: Bike Sharing Dataset

This data set collects the bike sharing counts aggregated on daily basis (2011-2012) together with the calender information (year/season/weekday/holiday) and daily weather conditions. There are 4 weather realted variables:

- weathersit: (GOOD, MISTY, RAIN/SNOW/STORM)
- temperature (in Celsius)
- humidity (relative to 100%)
- windspeed

The objective is to predict the bike sharing counts given a date, and figure out the impacts on the counts due to weather conditions.

More data descriptions can be found at http://archive.ics.uci.edu/ml/datasets/Bike+Sharing+Dataset

Your analysis should include the following:

- EDA and preliminary findings.
- Fit tree-based models and summarize their prediction performance. The tuning parameters for tree-based models should be determined via cross validation.
- Fit a linear regression and a nonlinear regression model for the target variable **count**. Again, your models should go through a proper model selection procedure. Summarize the selected models with their prediction performance.
- Identify the important input variables for predicting bike sharing counts. In particular, characterize the effects of weather conditions on the bike counts **quantitatively**.
- Give a brief analysis summary.

```r
bike <- read.csv(file="bike.csv") #read data
head(bike)
```

```
##   days_since_2011 season year month    holiday weekday     workingday weather
## 1               1 WINTER 2011   JAN NO HOLIDAY     SAT NO WORKING DAY   MISTY
## 2               2 WINTER 2011   JAN NO HOLIDAY     SUN NO WORKING DAY   MISTY
## 3               3 WINTER 2011   JAN NO HOLIDAY     MON    WORKING DAY    GOOD
## 4               4 WINTER 2011   JAN NO HOLIDAY     TUE    WORKING DAY    GOOD
## 5               5 WINTER 2011   JAN NO HOLIDAY     WED    WORKING DAY    GOOD
## 6               6 WINTER 2011   JAN NO HOLIDAY     THU    WORKING DAY    GOOD
```

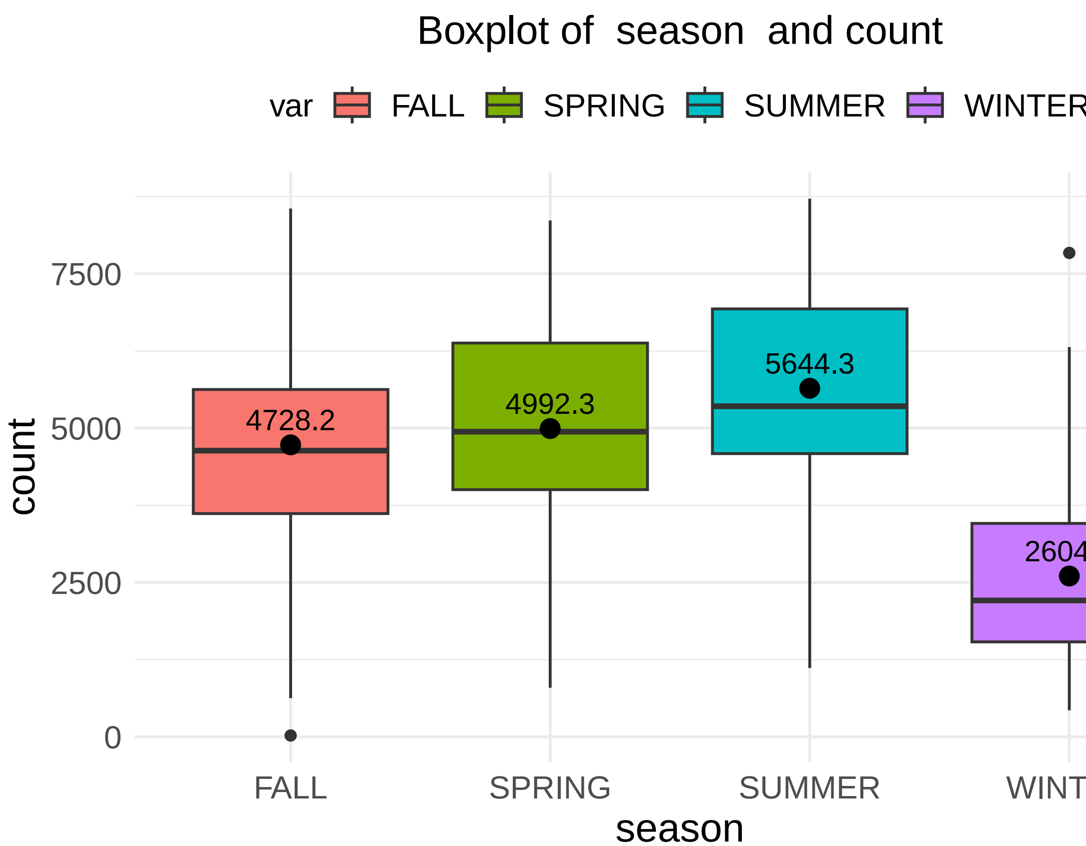
```
##       temp     hum windspeed count
## 1 8.175849 80.5833 10.749882   985
## 2 9.083466 69.6087 16.652113   801
## 3 1.229108 43.7273 16.636703  1349
## 4 1.400000 59.0435 10.739832  1562
## 5 2.666979 43.6957 12.522300  1600
## 6 1.604356 51.8261  6.000868  1606
```

1. EDA

Guess: The count may be related to the season, weekday, and the weather condition. For example: In winter, people may not like to ride a bile. Besides, On weekdays, people need to take more rides to go to work. Also, when the weather is bad, it is unlikely people would like to ride a bike.

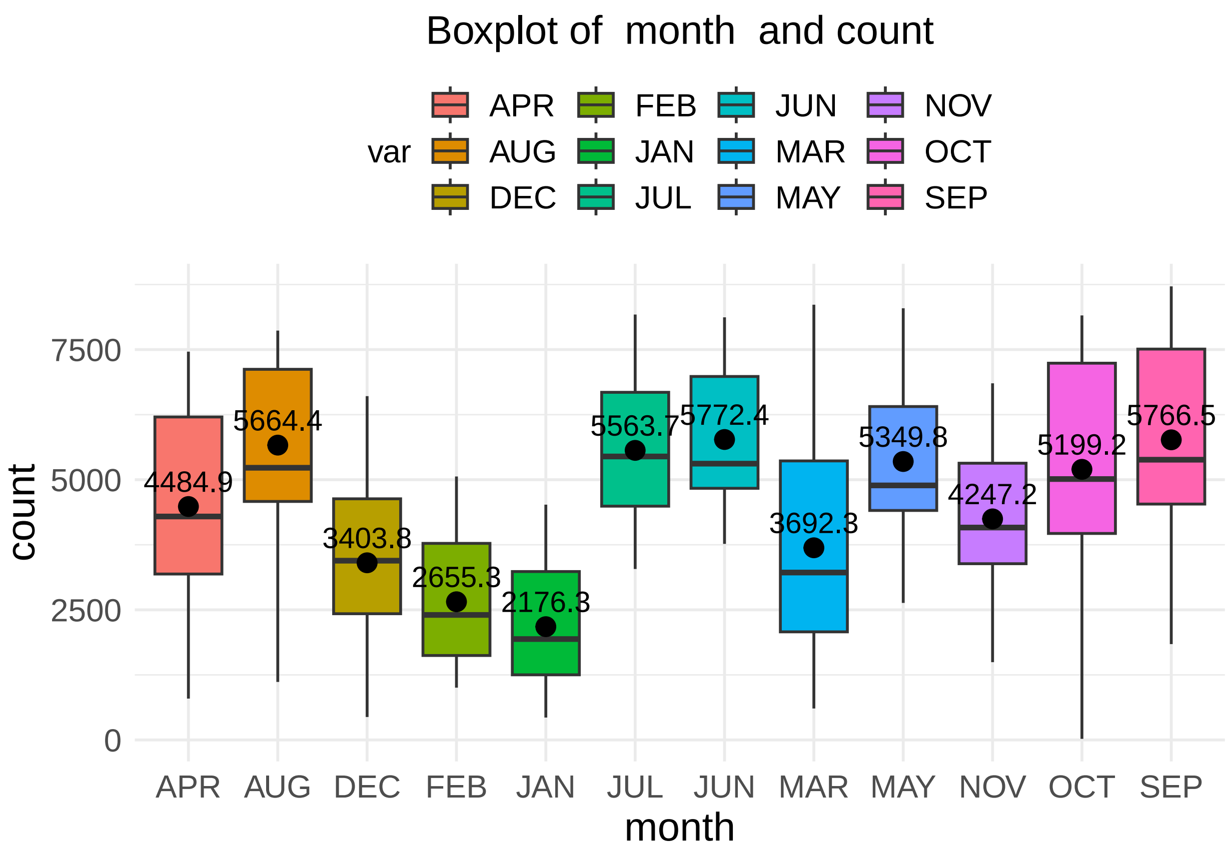
```
fun_mean <- function(x){
  return(data.frame(y=mean(x), label= round(mean(x,na.rm=T), 1)))}

plot_box <- function(var, str){
  ggplot(bike,aes(x=var, y=count,fill=var))+
    geom_boxplot()+ # box plot
    stat_summary(fun.y=mean, geom="point", size=3,show.legend=F)+   # mean
    stat_summary(fun.data =fun_mean, geom="text", vjust=-0.7,show.legend=F)+  # mean
    xlab(str)+
    ylab('count')+
    ggtitle(paste("Boxplot of ", str, " and count"))+
    theme_minimal() +
    theme(plot.title=element_text(hjust=0.5,size=15),
          axis.title=element_text(size=15),
          axis.text = element_text(size=12),
          legend.text=element_text(size=12),
          legend.title=element_text(size=12),
          legend.position = 'top')}
```

```
plot_box(bike$season, "season")
```



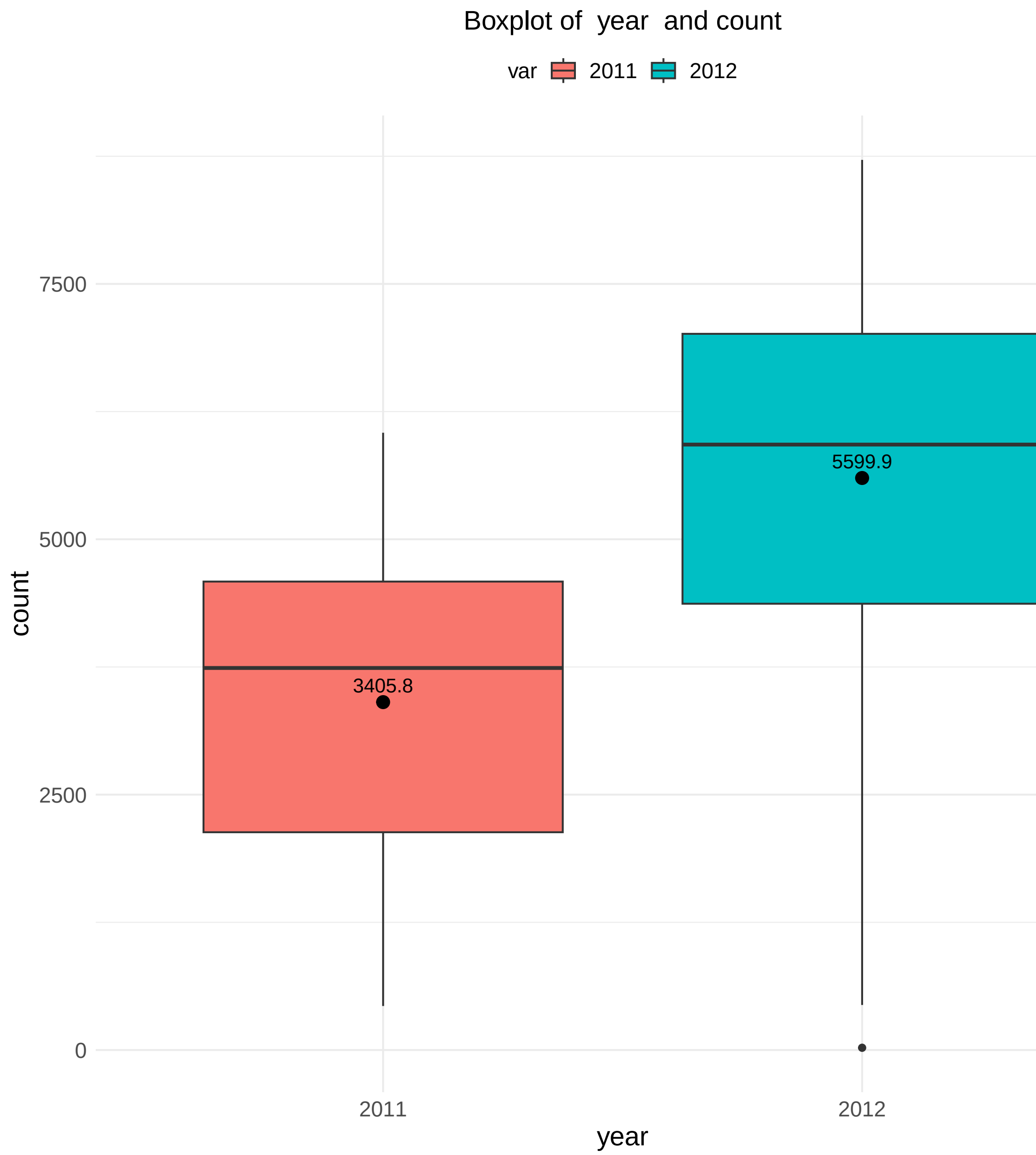
(1) The Inpact of Season

```
plot_box(bike$month, "month")
```



So the guess is corrct – It seems like the colder it is, the more unlikely people tend to ride a bike.

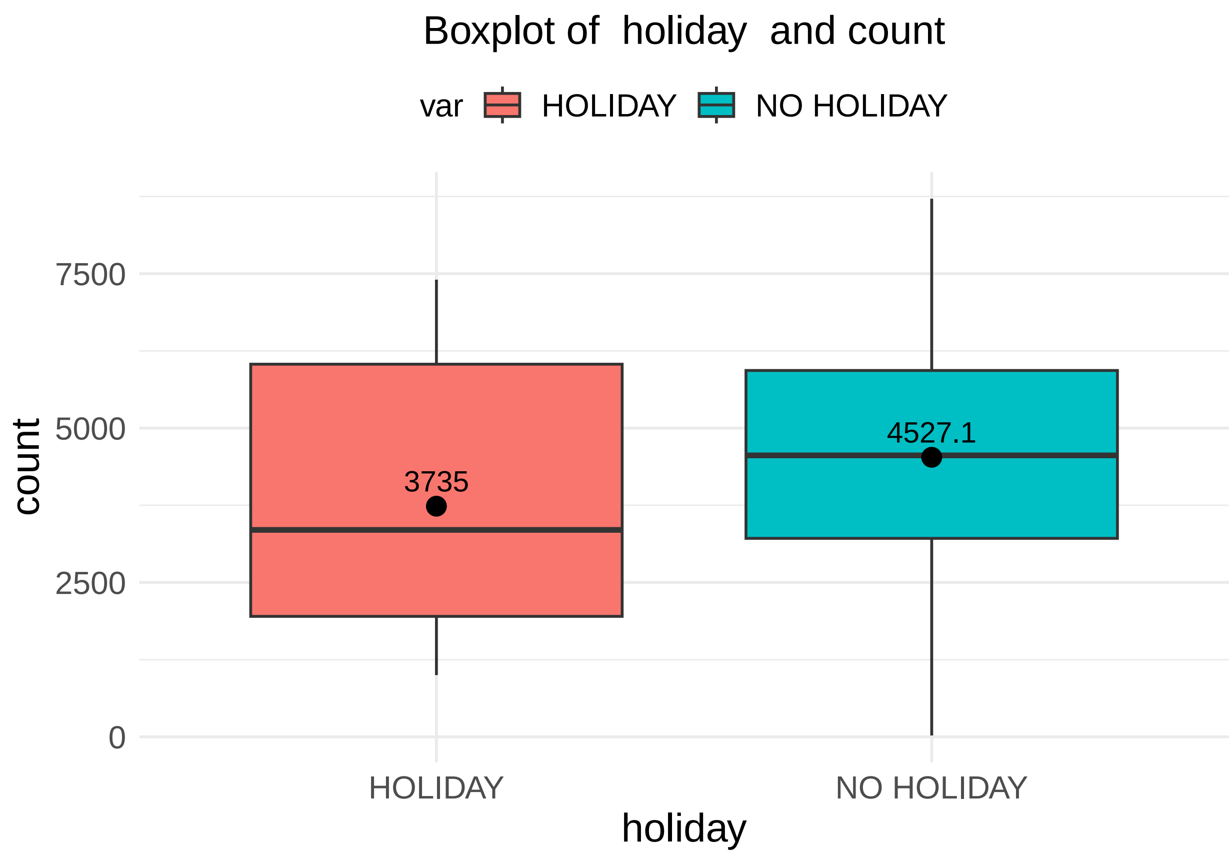
```
bike$year <- as.character(bike$year)
plot_box(bike$year, "year")
```



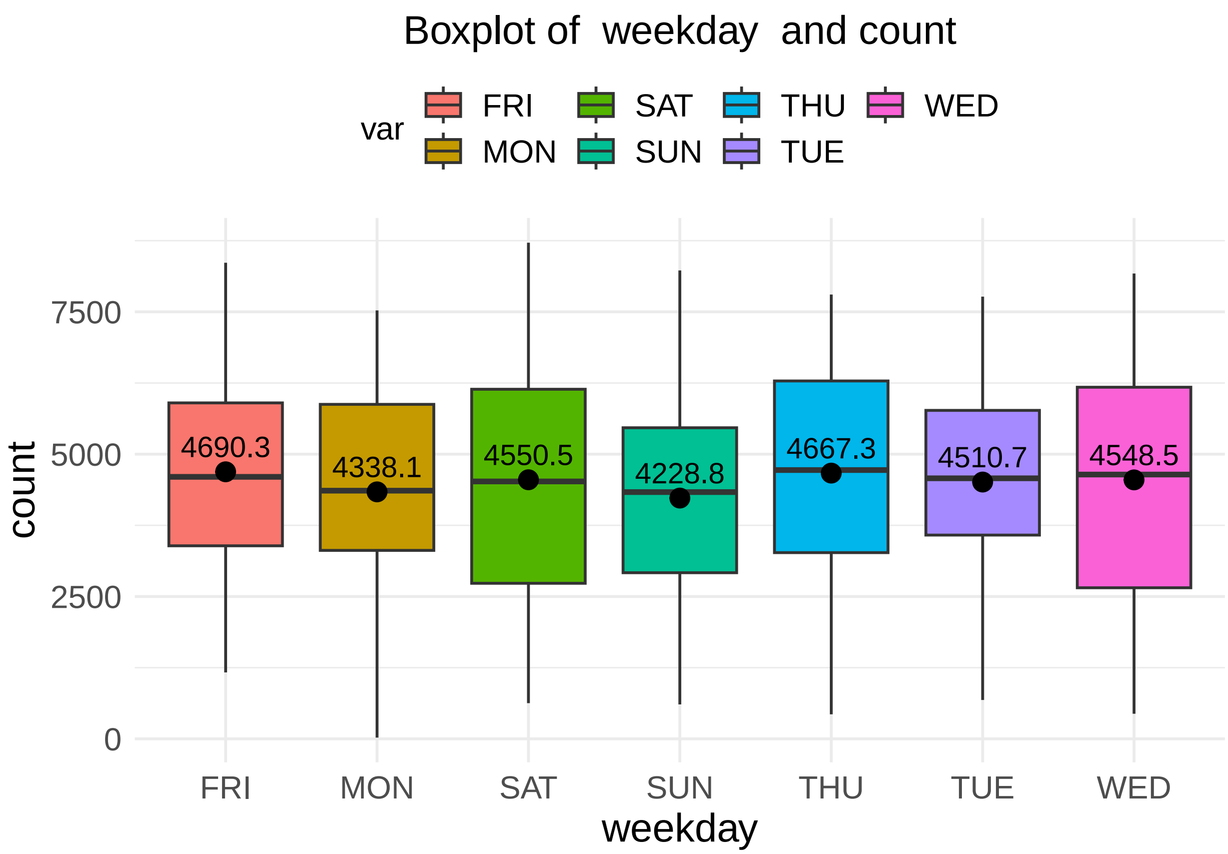
(2) The Inpact of Time

In year 2012, people are more likely to ride a bike than 2011.

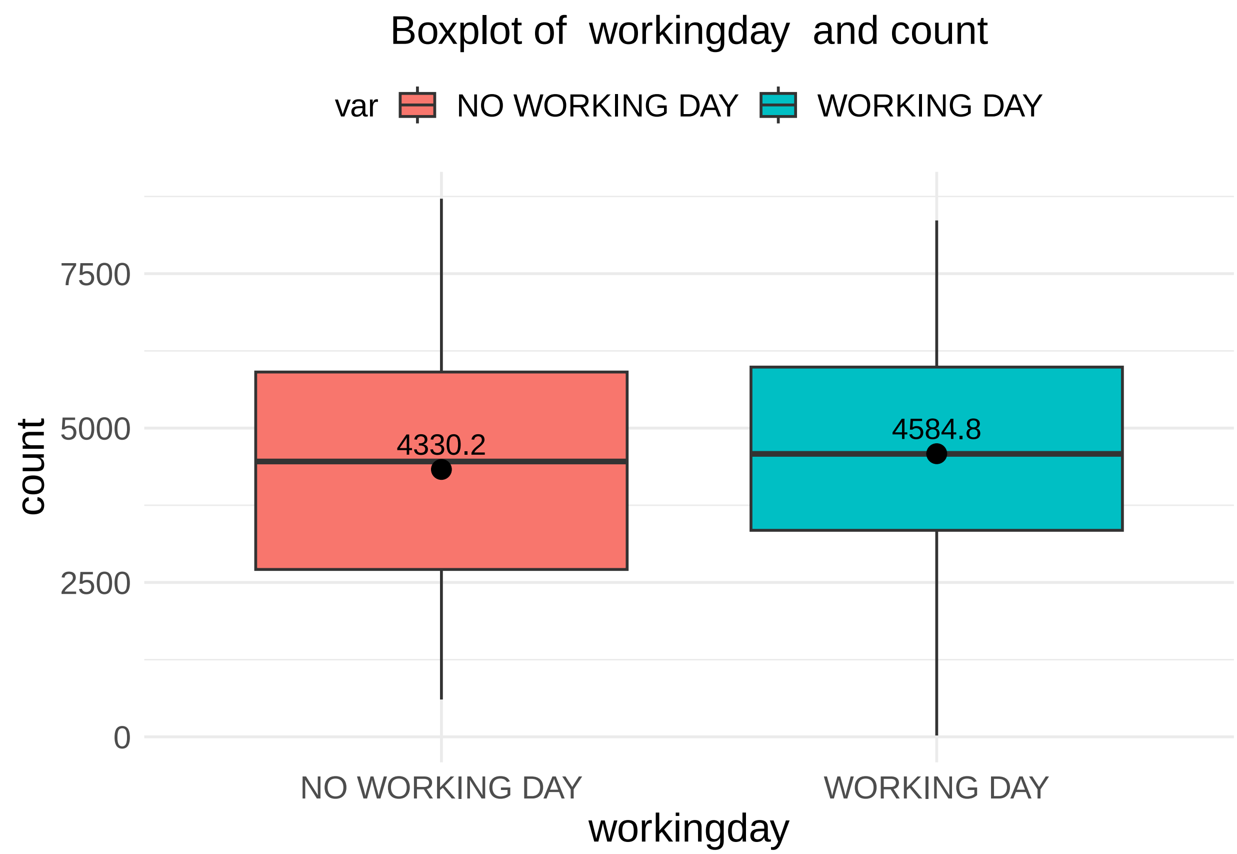
```
plot_box(bike$holiday, "holiday")
```




```
plot_box(bike$weekday, "weekday")
```

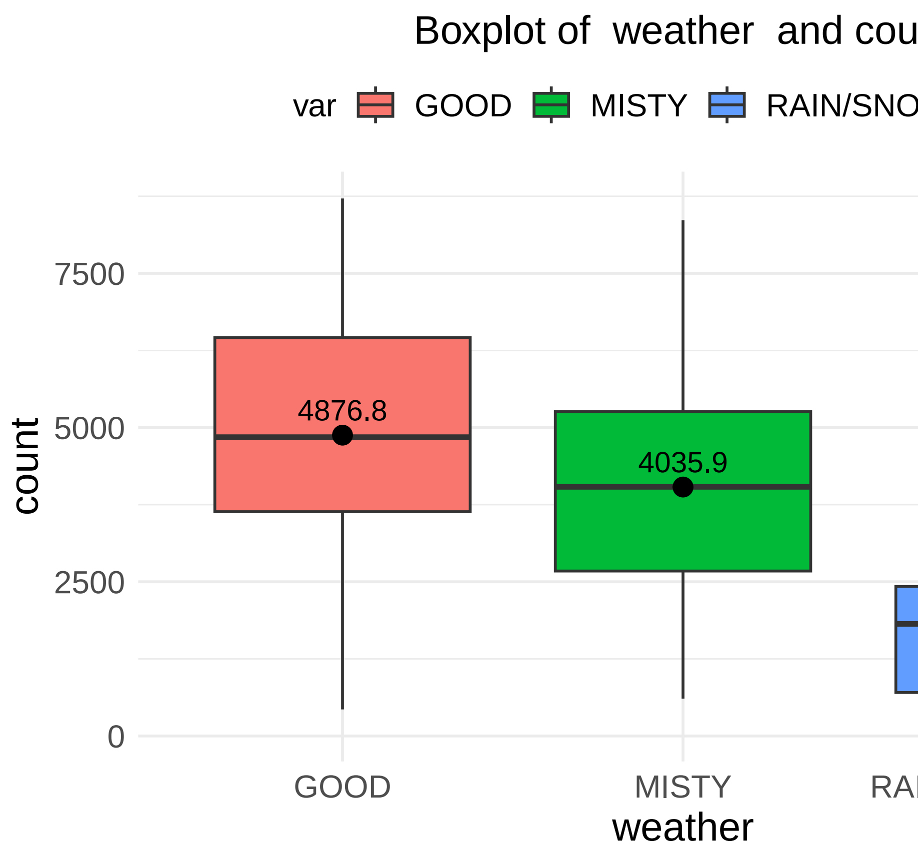



```r
plot_box(bike$workingday, "workingday")
```



From the box plots, no significance difference on counts are shown for working days or not. This is out of my initial guess.

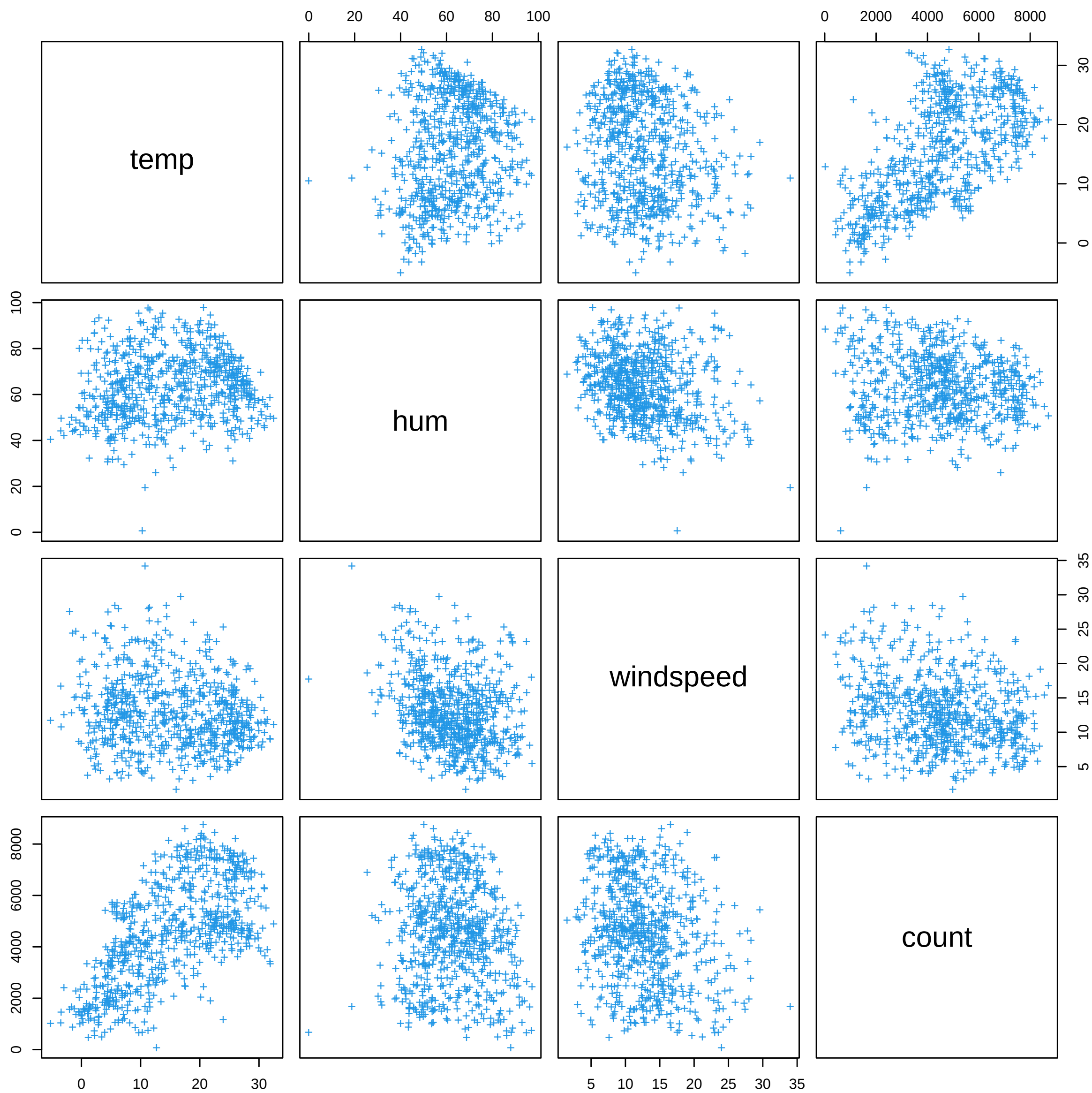
```
plot_box(bike$weather, "weather")
```



(3) The Inpact of Weather Condition.

My guess is correct again. When the weather is bad, people tend not to ride a bike.

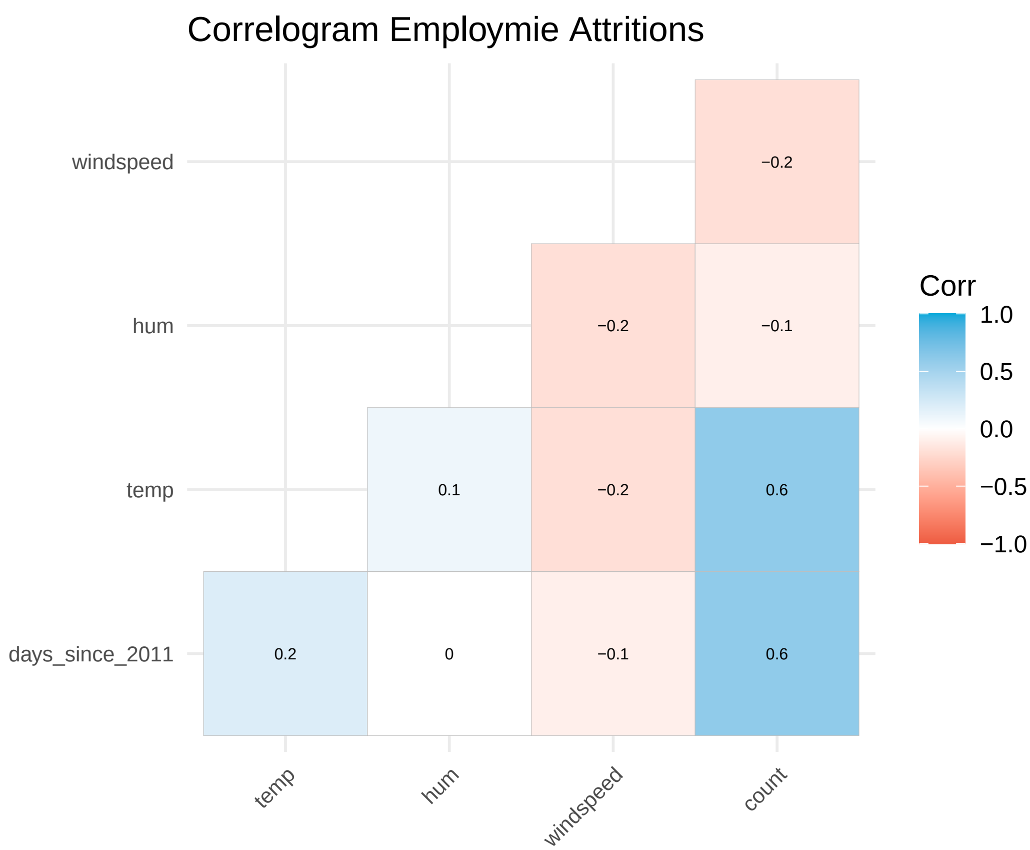
```
pairs(bike[,9:12], col=12, pch="+")
```



However, given the weather data such as `temp`, `hum` and `windspeed`, only `temp` shows a strong pattern. This is correspond with my initial inference that the colder it is, the fewer people ride bike.

```
select_if(bike, is.numeric) %>%
  cor %>%
  round(.,1) %>%
  ggcorrplot(type='lower',
             lab = TRUE,
             lab_size = 2,
             tl.cex = 8,
             method="square",
```

```
             colors = c("tomato2", "white", "#01A9DB"),
             title="Correlogram Employmie Attritions",
             ggtheme=theme_minimal())
```



(4) Correlation plot

Notice that no numeric features have a strong correlation, which is a good news.

```
# train-test split (70%-30%)
set.seed(1)
train_index <- sample(1:nrow(bike),(0.7*nrow(bike)))
train <- bike[train_index,2:12]
test <- bike[-train_index,2:12]
x_train <- train[,1:10]
y_train <- train[,11]
x_test <- test[,1:10]
y_test <- test[,11]
```

(5) Data preprocessing

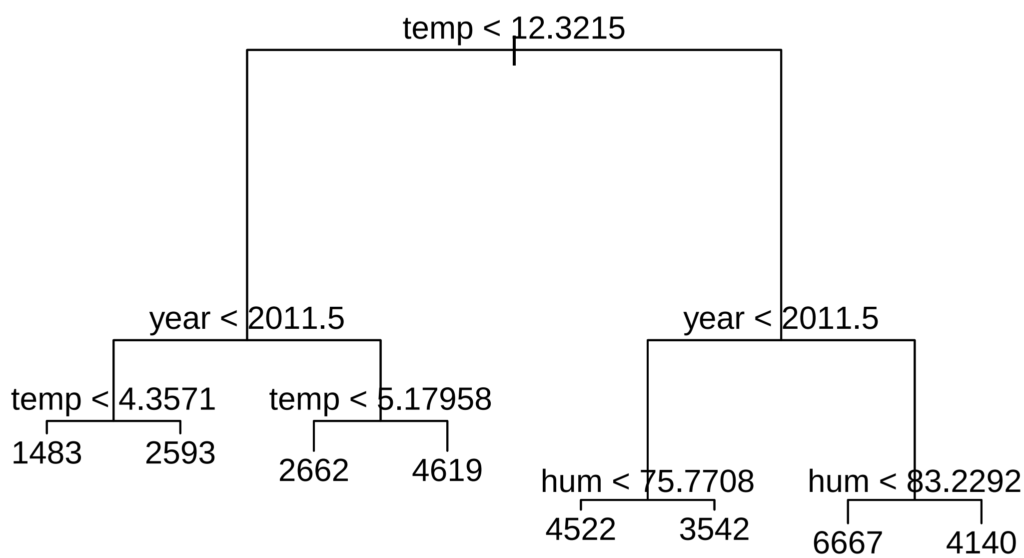
2. Tree-Based Models and Fine-Tuning

(1) Decision tree I plant a big tree, then pruning it.

```r
set.seed(1)
tree.bike <- tree::tree(count ~ ., train)
summary(tree.bike)
```

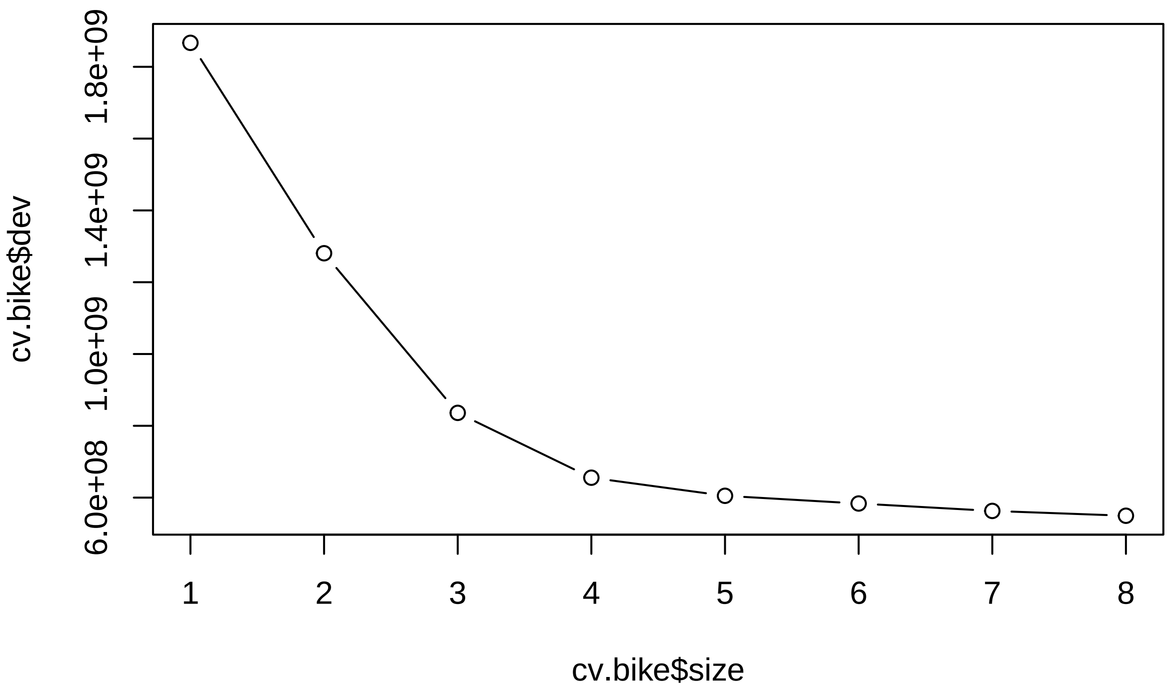
```
##
## Regression tree:
## tree::tree(formula = count ~ ., data = train)
## Variables actually used in tree construction:
## [1] "temp" "year" "hum"
## Number of terminal nodes:  8
## Residual mean deviance:  885200 = 445300000 / 503
## Distribution of residuals:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -4118.0  -530.4    72.7     0.0   675.1  2713.0
```

```r
plot(tree.bike)
text(tree.bike, pretty = 0)
```

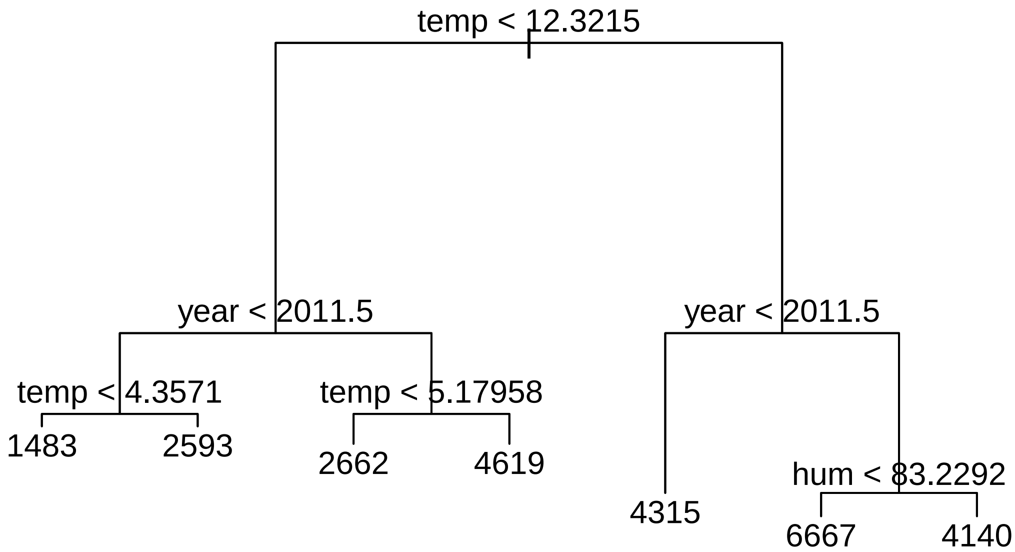


The unpruned tree shows that the only important features are `temp`, `hum` and year. In winter (perhaps start with October), the weather becomes cold. Also, when it is raining (`hum` is high), people do not want to ride a bike. Using cross validation, I found that pruning a tree to size 7 improves the performance.

```r
cv.bike <- cv.tree(tree.bike)
plot(cv.bike$size, cv.bike$dev, type = "b")
```



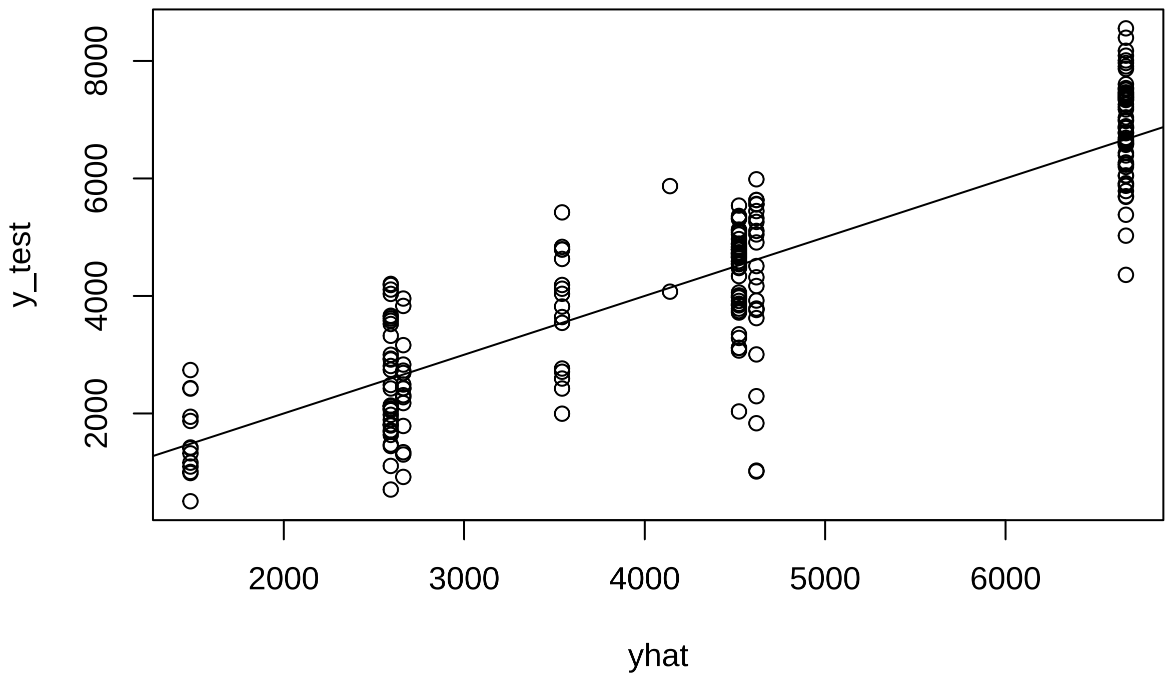

```r
prune.bike <- prune.tree(tree.bike, best = 7)
plot(prune.bike)
text(prune.bike, pretty = 0)
```



This plot shows that two leaves are pruned. Under some conditions (`temp` > 12.3215 and in year 2012), we need not to consider humidity (`hum`).

Now we predict on the test set.

```
yhat <- predict(tree.bike, newdata = test[,1:10])
y_test <- test[,"count"]
plot(yhat, y_test)
abline(0, 1)
```




```
mean((yhat - y_test)^2)
```

```
## [1] 865038.3
```

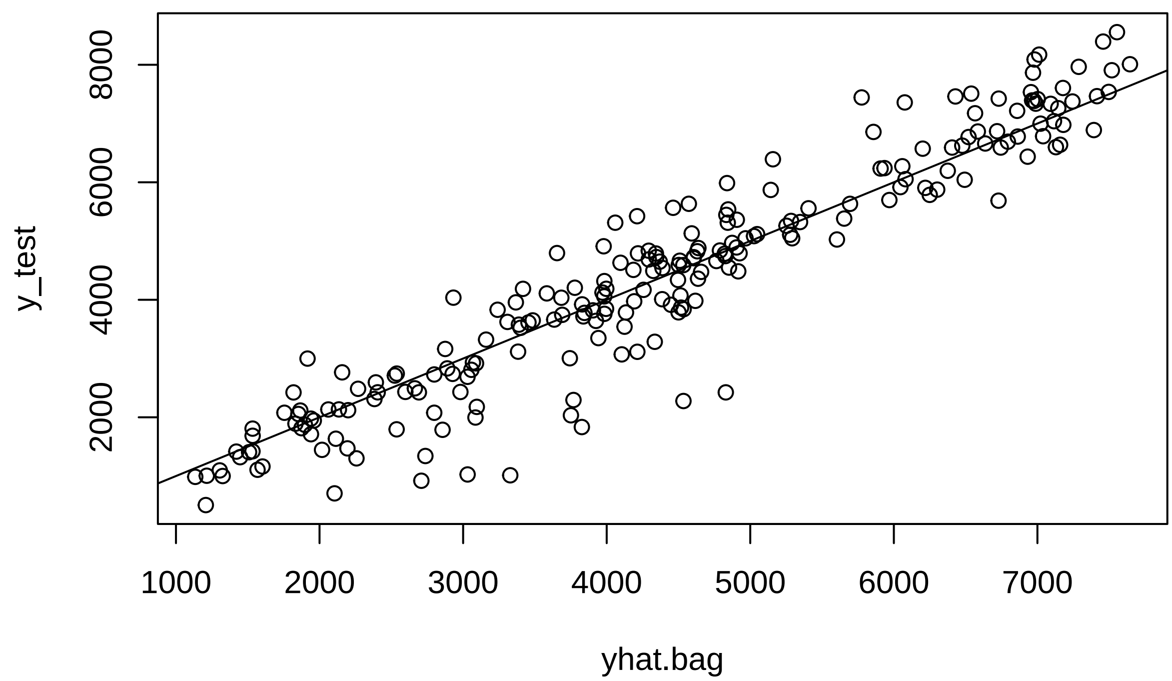
The RMSE of the tree is 865038.3.

(2) Random Forest Since there are 10 features, I take `mtry` $= \lceil \sqrt{10} \rceil = 4$.

```
set.seed(1)
bag.bike <- randomForest(count ~ .,
                         data = train,
                         mtry = 4,
                         ntree = 500,
                         importance = TRUE)
bag.bike
```

```
##
## Call:
##  randomForest(formula = count ~ ., data = train, mtry = 4, ntree = 500,      importance = TRUE)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 4
##
##           Mean of squared residuals: 503916.9
##                     % Var explained: 86.15
```

```r
yhat.bag <- predict(bag.bike, newdata = test[,1:10])
plot(yhat.bag, y_test)
abline(0, 1)
```

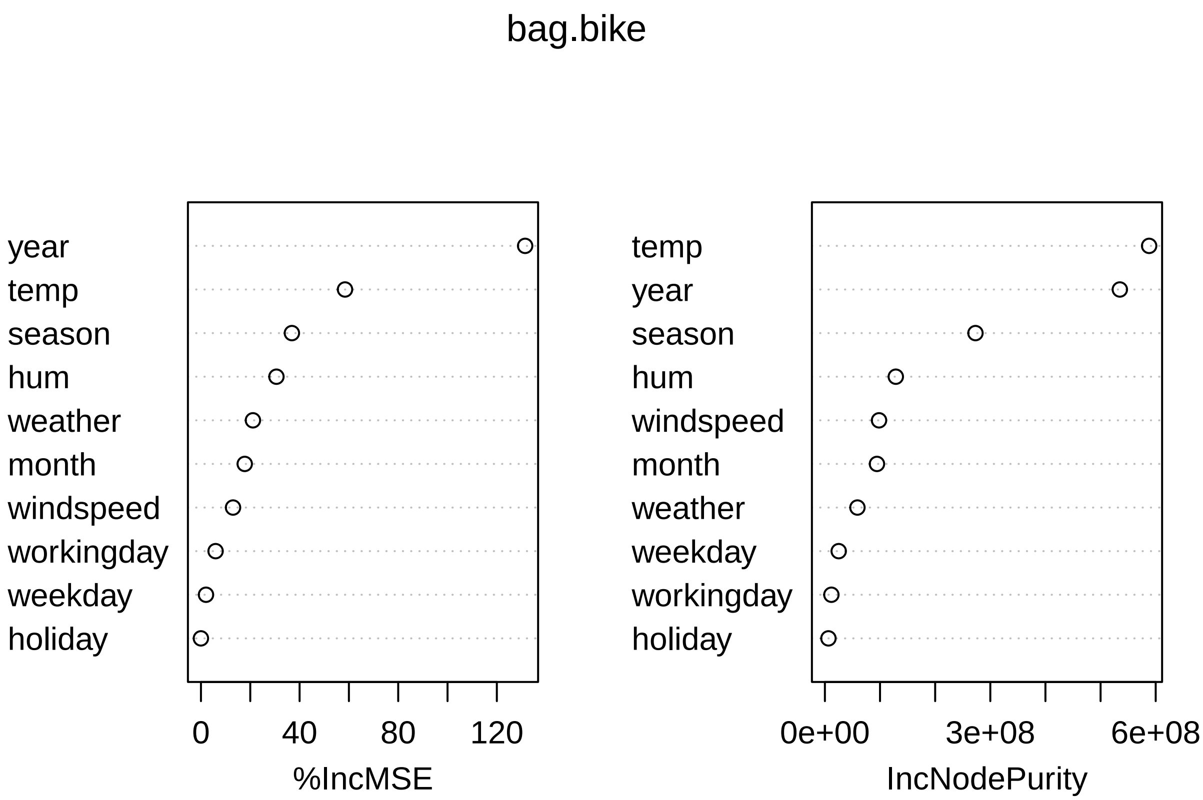



```r
mean((yhat.bag - y_test)^2)
```

```
## [1] 421652.6
```

The random forest gives the RMSE 421652.6 on the test set, which is better than a pruned tree. Next, I show the important features

```r
varImpPlot(bag.bike)
```



If we use the elbow rule to select the important feature by MSE, it would be `year`, `temp`, `season`, `hum` and `weather`. This meets the analysis in EDA and tree algorithm.

```
#dummy
train_dummy_X <- model.matrix(~.,train)[,-1]
train_dummy_X <- train_dummy_X[, colnames(train_dummy_X) != "count"]
test_dummy_X <- model.matrix(~.,test)[,-1]
test_dummy_X <- test_dummy_X[, colnames(test_dummy_X) != "count"]
train_dummy_y <- as.numeric(y_train)
test_dummy_y <- as.numeric(y_test)
```

(3) XGBoost Using gridsearch to find the best parameter of XGBoost. The code is commented or else it would take too long to knitting to PDF files.

```
# create hyperparameter grid
hyper_grid <- expand.grid(
  eta = seq(0, 1, by=0.2),
  max_depth = c(2:5),
  min_child_weight = 2*c(1:5),
  subsample = c(0.6,0.7,0.8,0.9),
  colsample_bytree = c(0.6,0.7,0.8,0.9),
  optimal_trees = 0,               # a place to dump results
  min_error = 0)                   # a place to dump results
```

```
# # grid search
# for(i in 1:nrow(hyper_grid)){
#
#   # create parameter list
#   params <- list(
#     eta = hyper_grid$eta[i],
#     max_depth = hyper_grid$max_depth[i],
#     min_child_weight = hyper_grid$min_child_weight[i],
#     subsample = hyper_grid$subsample[i],
#     colsample_bytree = hyper_grid$colsample_bytree[i])
#
#   # reproducibility
#   set.seed(123)
#
#   # train model
#   xgb.tune <- xgb.cv(
#     params = params,
#     data = train_dummy_X,
#     label = train_dummy_y,
#     nrounds = 500,
#     nfold = 5,
#     objective='reg:squarederror',
#     eval_metric = "rmse",
#     verbose = 0,                # silent,
#     early_stopping_rounds = 10) # stop if no improvement for 10 consecutive trees
#
#
#   # add min training error and trees to grid
#   hyper_grid$optimal_trees[i] <- which.min(xgb.tune$evaluation_log$test_rmse_mean)
#   hyper_grid$min_error[i] <- min(xgb.tune$evaluation_log$test_rmse_mean)
# }
#
# best_para <- hyper_grid %>%
#   dplyr::arrange(min_error) %>%
#   head(1)
```

Obtaining the best parameter as follow.

```
# #(best)
# eta=best_para$eta=0.2
# max_depth=best_para$max_depth=4
# min_child_weight=best_para$min_child_weight=2
# subsample=best_para$subsample=0.8
# colsample_bytree=best_para$colsample_bytree=0.8
# optimal_trees=best_para$optimal_trees=73
```

Now feed it into the XGBoost model.

```
# parameter list
params <- list(
  eta = 0.2,
  max_depth = 4,
  min_child_weight = 2,
```

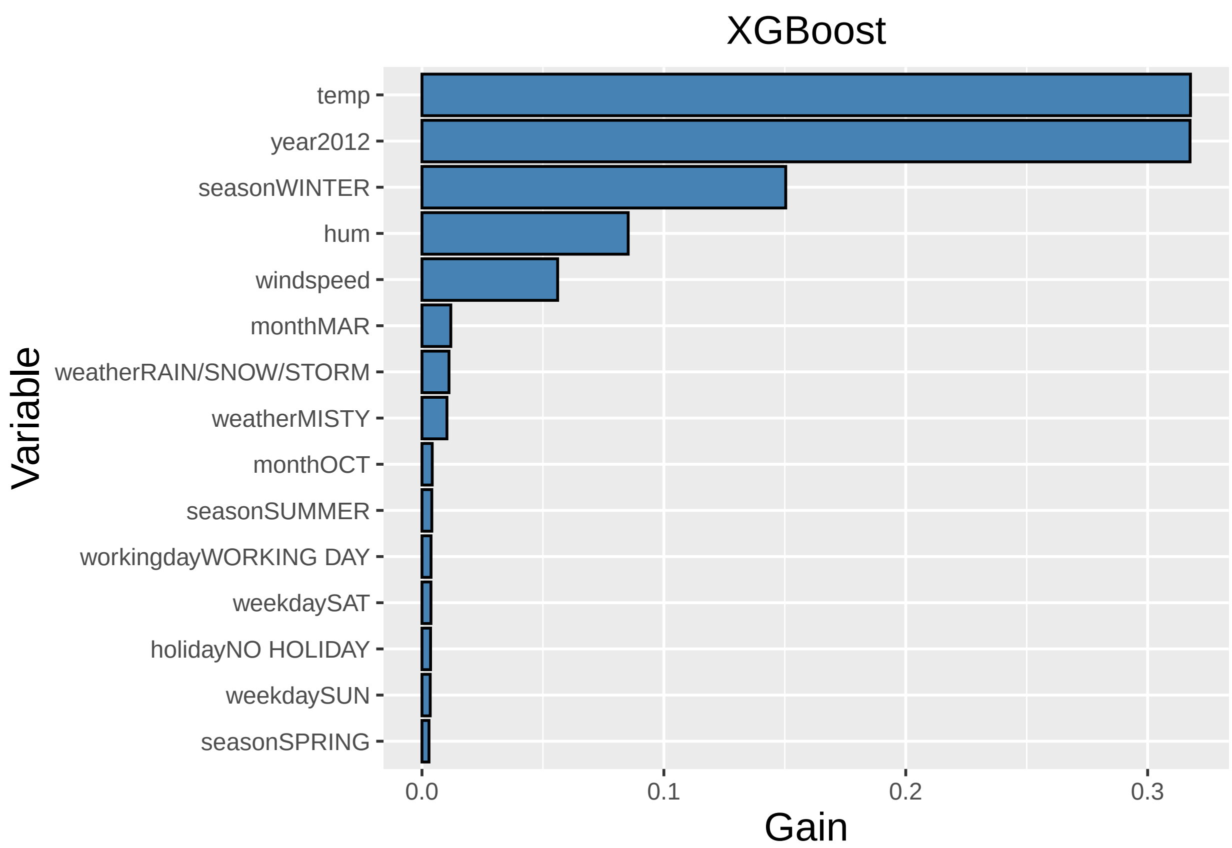
```r
  subsample = 0.8,
  colsample_bytree = 0.8)

set.seed(1)
# train final model
modfinal_xgb <- xgboost(
  params = params,
  data = train_dummy_X,
  label = train_dummy_y,
  nrounds = 100,
  objective='reg:squarederror',
  eval_metric = "rmse",
  verbose = 0)
```

```r
pred_y = predict(modfinal_xgb, test_dummy_X)
mean((y_test - pred_y)^2) #mse
```

```
## [1] 393987.6
```

This gives a very good 393987.6, outperforms than tree and Random forest! Perhaps this is why some people say XGBoost is "secret weapon of Kaggle"

```r
xgb.importance(model = modfinal_xgb) %>%
  as.data.frame() %>%
  `colnames<-`(c("Feature",'Gain','Cover',"Frequency")) %>%
  arrange(desc(Gain)) %>%
  top_n(15,wt = Gain) %>%
  ggplot(aes(x = reorder(Feature, Gain),y = Gain)) +
  geom_col(fill = 'steelblue', color = 'black') +
  coord_flip() +
  ggtitle(label = "XGBoost") +
  xlab('Variable')+
  ylab('Gain')+
  theme(plot.title=element_text(hjust=0.5,size=15),
        axis.title=element_text(size=15))
```

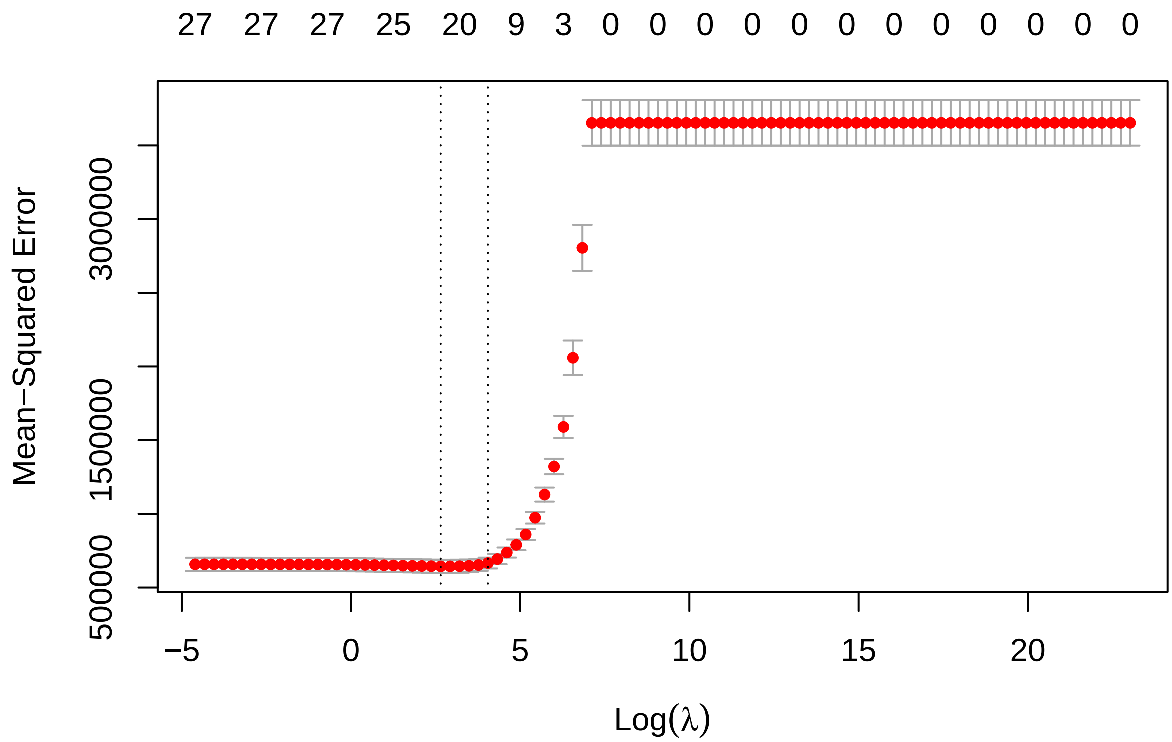


XGBoost selects `temp`, `year`, `season` (whether it is winter or not), `hum`, `windspeed` as the most important features. The difference of XGBoost and the other algorithms is that XGBoost takes wind speed into consideration.

3. Linear Regression and a Nonlinear Regression Model

(1) Linear Regression First, First, I find the best λ value of LASSO. Then, I use Lasso to select features.

```
set.seed(1)
grid <- 10^seq(10, -2, length = 100) # use grid search to find lambda
cv.out <- cv.glmnet(train_dummy_X, train_dummy_y, alpha = 1, nfolds=5, lambda=grid) # LASSO
plot(cv.out)
```




```r
bestlam_lasso <- cv.out$lambda.min # best lambda
bestlam_lasso
```

```
## [1] 14.17474
```

```r
# retrain the model with the best lambda
lasso.bike <- glmnet(train_dummy_X, train_dummy_y, alpha = 1, lambda = grid)

# training performance
lasso.pred <- predict(lasso.bike, s = bestlam_lasso, newx = train_dummy_X)
MSE_train <- mean((lasso.pred - train_dummy_y)^2)

# testing performance
lasso.pred <- predict(lasso.bike, s = bestlam_lasso, newx = test_dummy_X)
MSE_test <- mean((lasso.pred - test_dummy_y)^2)

# results
MSE_train
```

```
## [1] 576650.1
```

```r
MSE_test
```

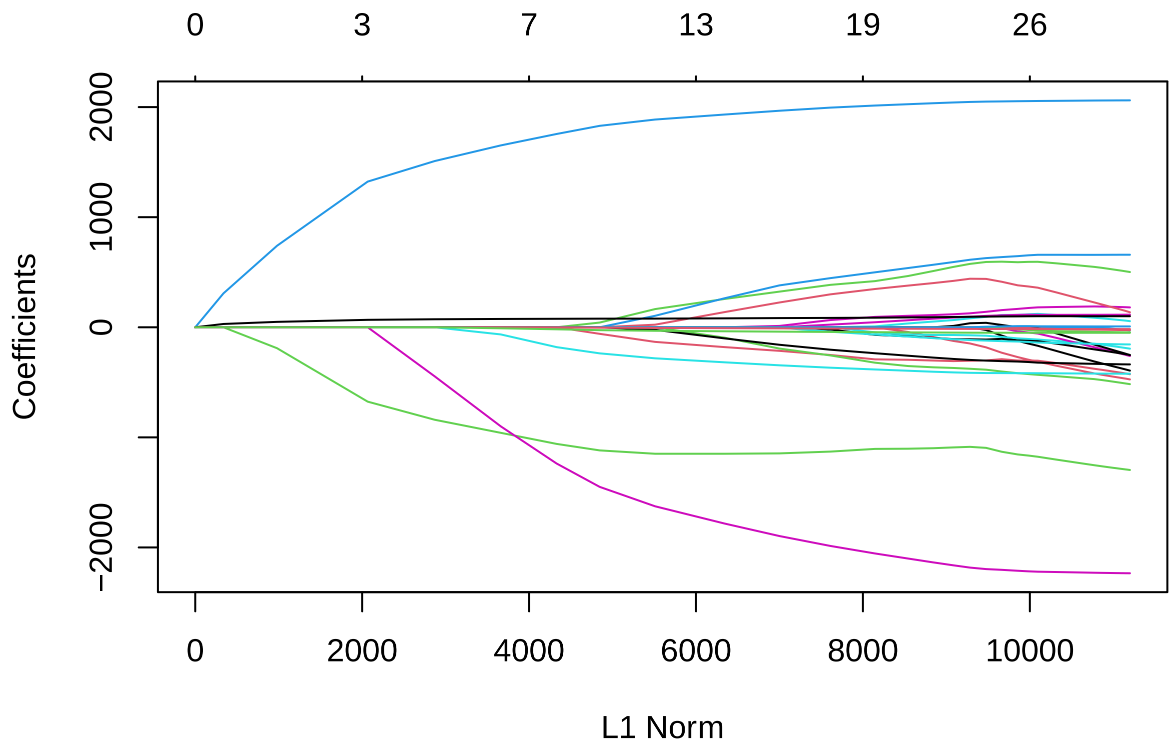
```
## [1] 640451
```

```r
lasso_coeff <- predict(lasso.bike,
                       s = bestlam_lasso,
                       exact = T,
                       type = "coefficients",
                       x = train_dummy_X,
                       y = train_dummy_y)
lasso_coeff[1:29,][which(lasso_coeff!=0)]
```

```
##           (Intercept)          seasonSUMMER          seasonWINTER 
##            3536.99872            -124.34025           -1091.28864 
##              year2012              monthAUG              monthFEB 
##            2041.88208             -70.25065            -107.63891 
##              monthJAN              monthJUL              monthMAR 
##            -307.28419            -369.63595              65.48380 
##              monthMAY              monthNOV              monthOCT 
##             118.15184              12.25287             421.34724 
##              monthSEP     holidayNO HOLIDAY            weekdayMON 
##             548.22761             592.35602            -108.88048 
##            weekdaySAT            weekdaySUN          weatherMISTY 
##              88.15895            -287.92436            -410.00483 
## weatherRAIN/SNOW/STORM                  temp                   hum 
##           -2162.51810              93.48235             -14.72513 
##             windspeed 
##             -48.48235
```

This shows that LASSO takes `season`, `weekday`, `month`, `year`, and the weather condition into consideration. On top of that, by the following graph, the most important five features are `year`, `weather`, `season`, `month`, and `holiday`.

```r
plot(lasso.bike)
```

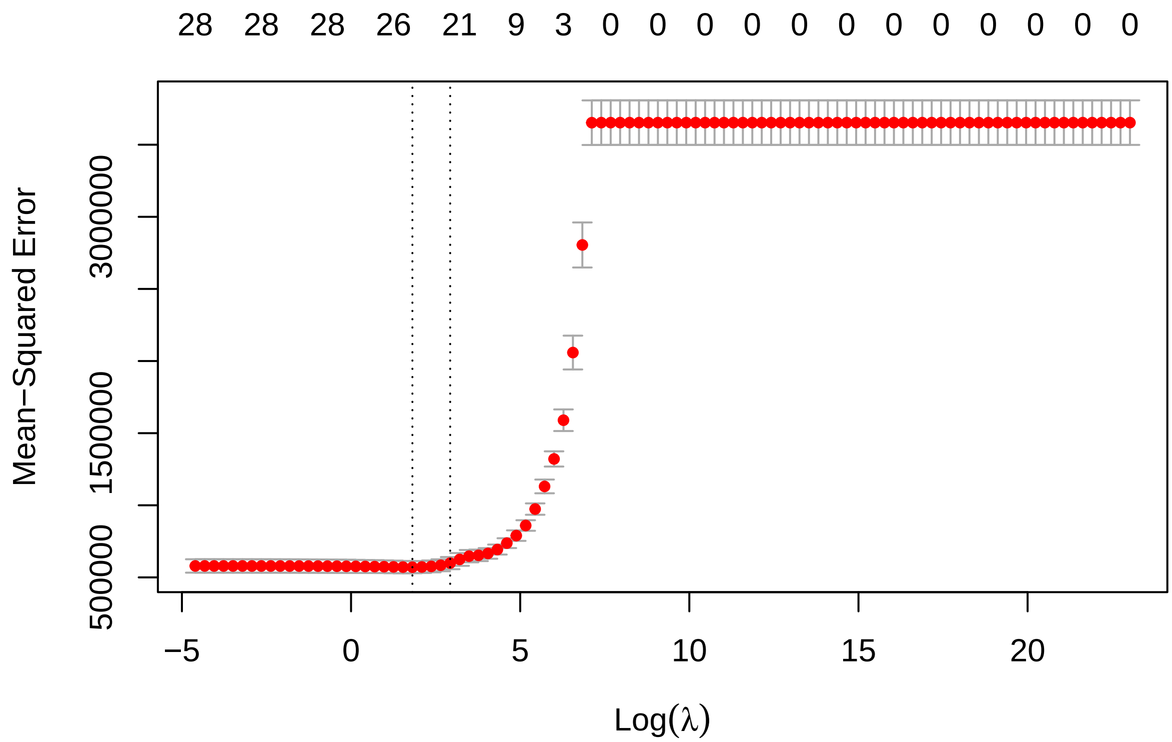


(2) Nonlinear Regression From EDA, we found that count and temp has a pattern. Hence, I add the square of `temp` into the dataset to see if the performence improves.

```
train_dummy_X2 <- cbind(train_dummy_X, train_dummy_X[,"temp"]^2)
test_dummy_X2 <- cbind(test_dummy_X, test_dummy_X[,"temp"]^2)
```

First, I find the best λ value of LASSO.

```
set.seed(1)
grid <- 10^seq(10, -2, length = 100) # use grid search to find lambda
cv.out <- cv.glmnet(train_dummy_X2, train_dummy_y, alpha = 1, nfolds=5, lambda=grid) # LASSO
plot(cv.out)
```




```
bestlam_lasso <- cv.out$lambda.min # best lambda
bestlam_lasso
```

```
## [1] 6.135907
```

```
# retrain the model with the best lambda
lasso.bike2 <- glmnet(train_dummy_X2, train_dummy_y, alpha = 1, lambda = grid)

# training performance
lasso.pred2 <- predict(lasso.bike2, s = bestlam_lasso, newx = train_dummy_X2)
MSE_train <- mean((lasso.pred2 - train_dummy_y)^2)

# testing performance
lasso.pred2 <- predict(lasso.bike2, s = bestlam_lasso, newx = test_dummy_X2)
MSE_test <- mean((lasso.pred2 - test_dummy_y)^2)

# results
MSE_train
```

```
## [1] 504586.7
```

```
MSE_test
```

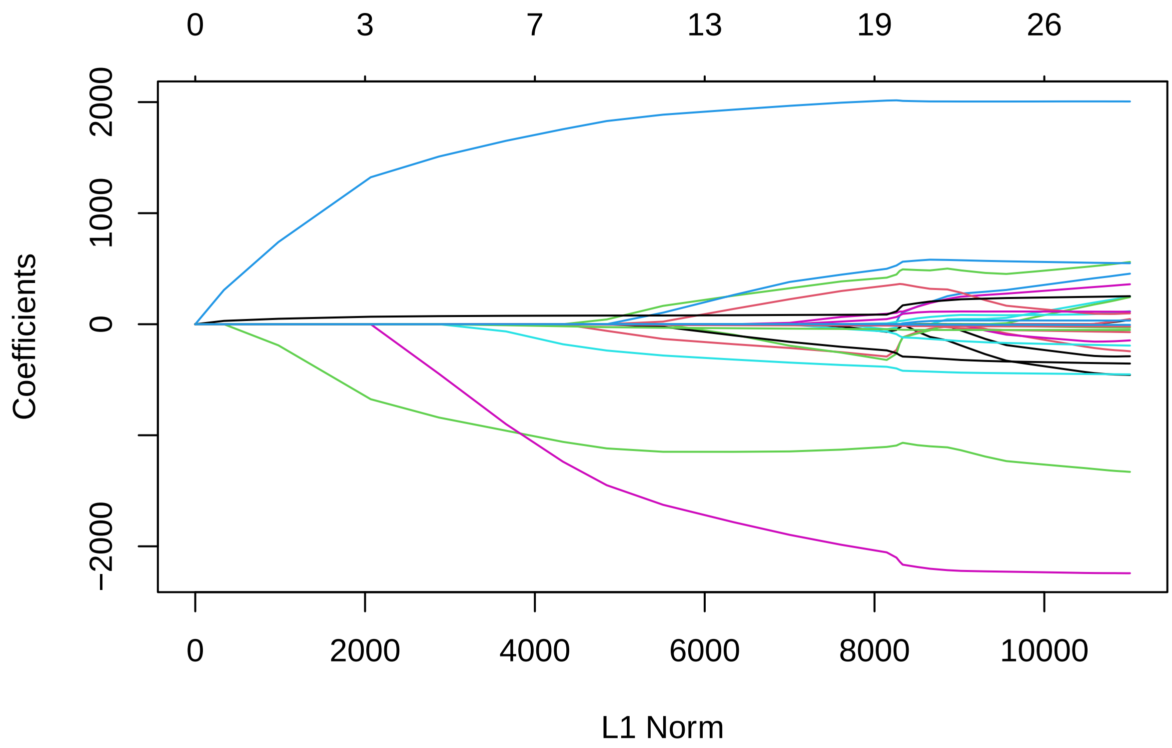
```
## [1] 562421.6
```

The testing RMSE is 562421.6, which is improved comparing without the quadratic terms of `temp`.

```
lasso_coeff <- predict(lasso.bike2,
                       s = bestlam_lasso,
                       exact = T,
                       type = "coefficients",
                       x = train_dummy_X,
                       y = train_dummy_y)
lasso_coeff[1:29,][which(lasso_coeff!=0)]
```

```
##           (Intercept)           seasonSPRING           seasonWINTER
##            3192.94147             -146.47739            -1108.85280
##              year2012               monthAUG               monthJAN
##            2005.92752               44.26225              -20.38373
##              monthJUN               monthMAR               monthMAY
##             252.34360               76.83980              227.13169
##              monthNOV               monthOCT               monthSEP
##             -17.38618              312.83439              500.82502
##     holidayNO HOLIDAY             weekdayMON             weekdaySAT
##             578.64123             -143.58872              113.66755
##            weekdaySUN             weekdayTHU             weekdayWED
##            -314.11814              -26.31573               32.57870
##          weatherMISTY weatherRAIN/SNOW/STORM                   temp
##            -432.26805            -2214.99472              215.30359
##                   hum              windspeed                   <NA>
##             -18.00903              -51.83234                     NA
```

```
plot(lasso.bike2)
```



4. Summary

The RMSE on testing set and selected features are presented in the following tables.

```r
knitr::include_graphics("1.png")
```

model	RMSE
tree	865038
Random Forest	421653
XGBoost	393988
LASSO	640451
LASSO (adapted)	562422

```r
knitr::include_graphics("2.png")
```

Top 5 important features				
Tree	Random Forest	XGBoost	LASSO	LASSO (adapted)
temp	temp	temp	year	year
year	year	year	weather	season
hum	season	season	season	holiday
	hum	hum	month	month
	weather	windspeed	holiday	weather

Among all the models, XGBoost gives the best regression performance. To sum up, from the features selected by each model, we may conduct the following inference:

(1) In year 2012, either more people knows the sharing bikes, or more bikes are available. Hence the number of people renting a bike increases.

(2) When the weather condition is bad, such as raining (high humidity), or it is too cold (in winter months), people are not willing to rent a bike.

(3) During the break, such as weekend or holiday, people are more likely to rent a bike. Perhaps people drive or take an MRT to work since it is too far for a bike.

Problem 2: Airline Customer Satisfaction

This dataset contains an airline passenger satisfaction survey. The data collect the demographic information about the customers, their feedback regarding the flight experiences and some flight information.

The goal of the analysis has two folds:

- Find a good model for the classification of customer satisfaction, and summarize the performance.
- Identify the important aspects of the flight experience related to customer satisfaction (or dissatisfaction). Based on your findings, give some specific suggestions (provided with the data evidence) to the airline company for achieving a higher customer satisfaction.
- Add some noise factors (both continuous variable and categorical variable) in your data set (call these added variables z1-z?), and check the ranking of variable importance of these z-variables relative to the original meaningful input variables. Make your comments regarding this experiment.

More data descriptions can be found at https://www.kaggle.com/teejmahal20/airline-passenger-satisfaction

```
survey <- read.csv(file="airline.csv") #read data
head(survey)
```

```
##       id Gender     Customer.Type Age   Type.of.Travel    Class Flight.Distance
## 1  70172   Male    Loyal Customer  13 Personal Travel Eco Plus             460
## 2   5047   Male disloyal Customer  25 Business travel Business             235
## 3 110028 Female    Loyal Customer  26 Business travel Business            1142
## 4  24026 Female    Loyal Customer  25 Business travel Business             562
## 5 119299   Male    Loyal Customer  61 Business travel Business             214
## 6 111157 Female    Loyal Customer  26 Personal Travel      Eco            1180
```

```
##   Inflight.wifi.service Departure.Arrival.time.convenient
## 1                     3                                 4
## 2                     3                                 2
## 3                     2                                 2
## 4                     2                                 5
## 5                     3                                 3
## 6                     3                                 4
##   Ease.of.Online.booking Gate.location Food.and.drink Online.boarding
## 1                      3             1              5               3
## 2                      3             3              1               3
## 3                      2             2              5               5
## 4                      5             5              2               2
## 5                      3             3              4               5
## 6                      2             1              1               2
##   Seat.comfort Inflight.entertainment On.board.service Leg.room.service
## 1            5                      5                4                3
## 2            1                      1                1                5
## 3            5                      5                4                3
## 4            2                      2                2                5
## 5            5                      3                3                4
## 6            1                      1                3                4
##   Baggage.handling Checkin.service Inflight.service Cleanliness
## 1                4               4                5           5
## 2                3               1                4           1
## 3                4               4                4           5
## 4                3               1                4           2
## 5                4               3                3           3
## 6                4               4                4           1
##   Departure.Delay.in.Minutes Arrival.Delay.in.Minutes            satisfaction
## 1                         25                       18 neutral or dissatisfied
## 2                          1                        6 neutral or dissatisfied
## 3                          0                        0               satisfied
## 4                         11                        9 neutral or dissatisfied
## 5                          0                        0               satisfied
## 6                          0                        0 neutral or dissatisfied
```

```
survey1 <- na.omit(survey) #remove missing data
```

1. Classification Model

```
# train-test split (70%-30%)
set.seed(48763)
train_index <- sample(1:nrow(survey1),(0.7*nrow(survey1)))
train_survey <- survey1[train_index,2:24]
test_survey <- survey1[-train_index,2:24]
x_train_survey <- train_survey[,1:22]
y_train_survey <- train_survey[,23]
x_test_survey <- test_survey[,1:22]
y_test_survey <- test_survey[,23]
train_dummy_X_survey <- model.matrix(~.,train_survey)[,2:23]
train_dummy_X_survey <- train_dummy_X_survey[, colnames(train_dummy_X_survey) != "satisfaction"]
test_dummy_X_survey <- model.matrix(~.,test_survey)[,2:23]
```

```r
test_dummy_X_survey <- test_dummy_X_survey[, colnames(test_dummy_X_survey) != "satisfaction"]
train_dummy_y_survey <- as.numeric(as.factor(y_train_survey))-1 # change label to 0 and 1
test_dummy_y_survey <- as.numeric(as.factor(y_test_survey))-1
```

(1) RandomForest Since random forest always performs better than a tree, I do not take tree into consideration in this problem.

```r
set.seed(1)
bag.surv <- randomForest(x = x_train_survey,
                         y = as.factor(y_train_survey),
                         mtry = 5,
                         ntree = 500,
                         importance = TRUE)
bag.surv
```

```
##
## Call:
##  randomForest(x = x_train_survey, y = as.factor(y_train_survey),      ntree = 500, mtry = 5, importa
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 5
##
##         OOB estimate of  error rate: 3.72%
## Confusion matrix:
##                         neutral or dissatisfied satisfied class.error
## neutral or dissatisfied                   40237       878  0.02135474
## satisfied                                  1822     29578  0.05802548
```

```r
y_pred <- predict(bag.surv, newdata = x_test_survey)
# Confusion Matrix
confusion_mtx = table(as.factor(y_test_survey), y_pred)
confusion_mtx
```

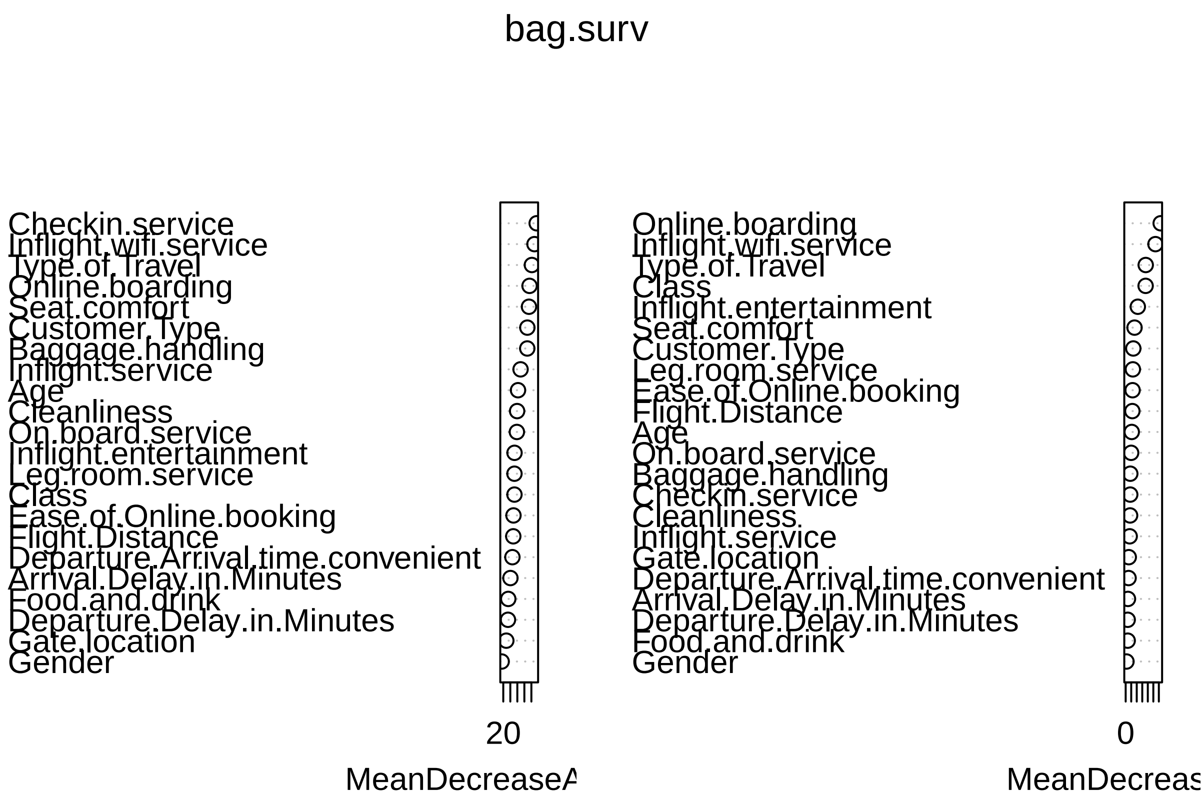
```
##                            y_pred
##                             neutral or dissatisfied satisfied
##   neutral or dissatisfied                     17199       383
##   satisfied                                     771     12726
```

```r
(17199+12726)/(17199+12726+771+383)
```

```
## [1] 0.9628688
```

The accuracy is 96.29%.

```r
varImpPlot(bag.surv)
```



In the left panel (i.e., considering the accuracy), it seems like some of the features ranked after 7 share same importance. The top 7 importance feature (choose 7 since there is a significant gap) are `Checkin service`, `Inflight wifi service`, `Type of Travel`, `Online boarding`, `Seat comfort`, `Customer Type`, and `Baggage handling`.

(2) XGBoost Using gridsearch to find the best parameter of XGBoost. The code is commented or else it would take too long to knitting to PDF files.

```
# # grid search
# for(i in 1:nrow(hyper_grid)){
#
#   # create parameter list
#   params <- list(
#     eta = hyper_grid$eta[i],
#     max_depth = hyper_grid$max_depth[i],
#     min_child_weight = hyper_grid$min_child_weight[i],
#     subsample = hyper_grid$subsample[i],
#     colsample_bytree = hyper_grid$colsample_bytree[i])
#
#   # reproducibility
#   set.seed(123)
#
#   # train model
#   xgb.tune <- xgb.cv(
#     params = params,
#     data = train_dummy_X_survey,
```

```
#     label = train_dummy_y_survey,
#     nrounds = 500,
#     nfold = 5,
#     objective='binary:logistic',
#     eval_metric = "error",
#     verbose = 0,               # silent,
#     early_stopping_rounds = 10) # stop if no improvement for 10 consecutive trees
#
#
#   # add min training error and trees to grid
#   hyper_grid$optimal_trees[i] <- which.min(xgb.tune$evaluation_log$test_error_mean)
#   hyper_grid$min_error[i] <- min(xgb.tune$evaluation_log$test_error_mean)
# }
#
# best_para <- hyper_grid %>%
#   dplyr::arrange(min_error) %>%
#   head(1)
```

Obtaining the best parameter as follow. This takes me about 16 hours!

```
# #(best)
# eta=best_para$eta=0.2
# max_depth=best_para$max_depth=5
# min_child_weight=best_para$min_child_weight=2
# subsample=best_para$subsample=0.9
# colsample_bytree=best_para$colsample_bytree=0.9
# optimal_trees=best_para$optimal_trees=125
```

Now feed it into the XGBoost model.

```
# parameter list
params <- list(eta = 0.2,
               max_depth = 5,
               min_child_weight = 2,
               subsample = 0.9,
               colsample_bytree = 0.9)

set.seed(1)
# train final model
modfinal_xgb_survey <- xgboost(params = params,
                               data = train_dummy_X_survey,
                               label = train_dummy_y_survey,
                               nrounds = 100,
                               objective='binary:logistic',
                               eval_metric = "error",
                               verbose = 0)
```

```
y_pred <- predict(modfinal_xgb_survey, newdata = test_dummy_X_survey)
y_pred <- as.numeric(y_pred > 0.5)
# Confusion Matrix
confusion_mtx = table(test_dummy_y_survey, y_pred)
confusion_mtx
```

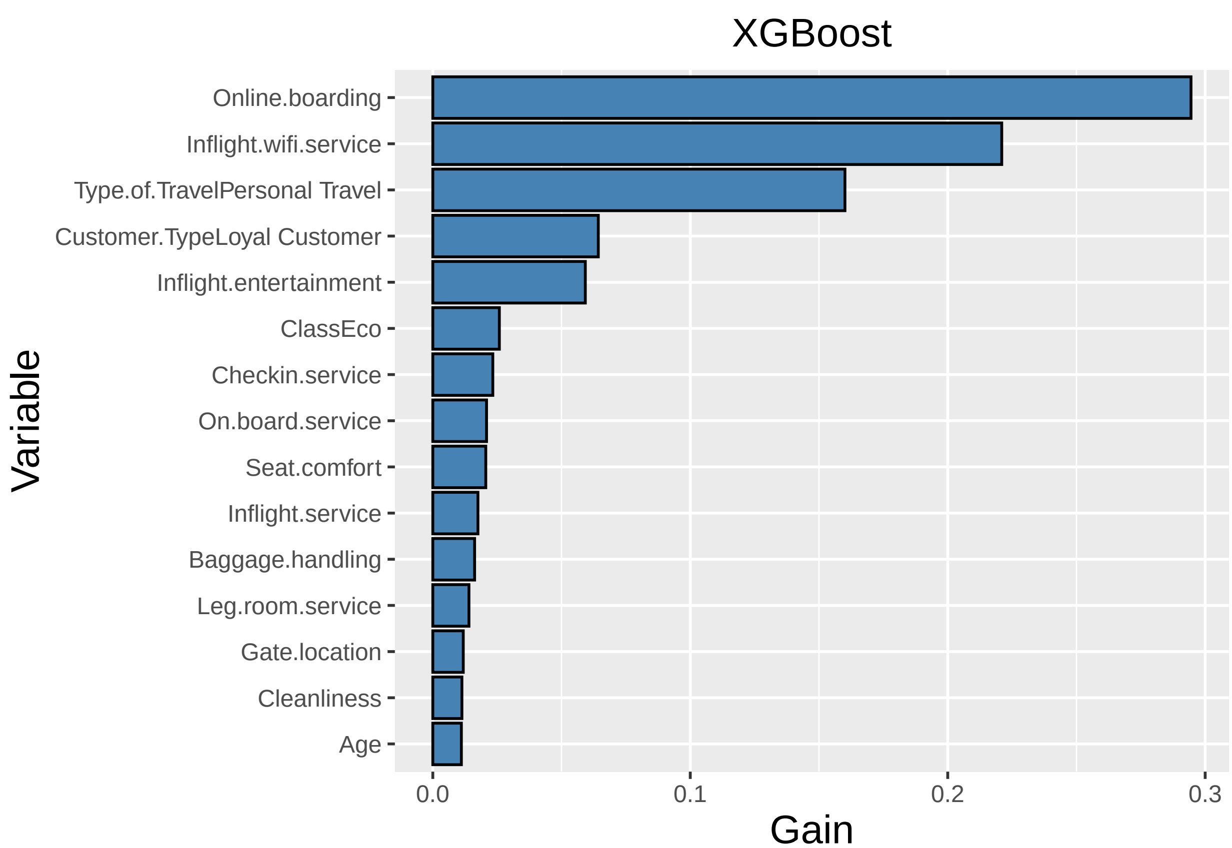
```
##                    y_pred
## test_dummy_y_survey     0     1
##                   0 17180   402
##                   1   837 12660
```

```r
(17180+12660)/(17180+12660+402+837)
```

```
## [1] 0.9601339
```

The accuracy is 96.01%.

```r
xgb.importance(model = modfinal_xgb_survey) %>%
  as.data.frame() %>%
  `colnames<-`(c("Feature",'Gain','Cover',"Frequency")) %>%
  arrange(desc(Gain)) %>%
  top_n(15,wt = Gain) %>%
  ggplot(aes(x = reorder(Feature, Gain),y = Gain)) +
  geom_col(fill = 'steelblue', color = 'black') +
  coord_flip() +
  ggtitle(label = "XGBoost") +
  xlab('Variable')+
  ylab('Gain')+
  theme(plot.title=element_text(hjust=0.5,size=15),
        axis.title=element_text(size=15))
```



The top 5 importance feature are `Online boarding`, `Inflight wifi service`, `Type of Travel`, `Customer Type`, `Inflight entertainment`.

3. Noise factors Experiment

```
X_train_noise <- cbind(train_dummy_X_survey,
                       rnorm(nrow(train_dummy_X_survey)),
                       rbinom(n=nrow(train_dummy_X_survey),size=1,prob=0.5))
X_test_noise <- cbind(test_dummy_X_survey,
                      rnorm(nrow(test_dummy_X_survey)),
                      rbinom(n=nrow(test_dummy_X_survey),size=1,prob=0.5))
# Must specify the column name of the new features,
# or else random forest cannot predict
colnames(X_train_noise)[23:24] <- c("gaussian_noise", "catgorical_noise")
colnames(X_test_noise)[23:24] <- c("gaussian_noise", "catgorical_noise")
```

(1) Adding Noise

```
set.seed(1)
bag.noise <- randomForest(x = X_train_noise,
                          y = as.factor(y_train_survey),
                          mtry = 5,
                          ntree = 500,
                          importance = TRUE)
```

```
y_pred_noise <- predict(bag.noise, newdata = X_test_noise)
confusion_mtx = table(as.factor(y_test_survey), y_pred_noise)
confusion_mtx
```

(2) Repeat Random forest

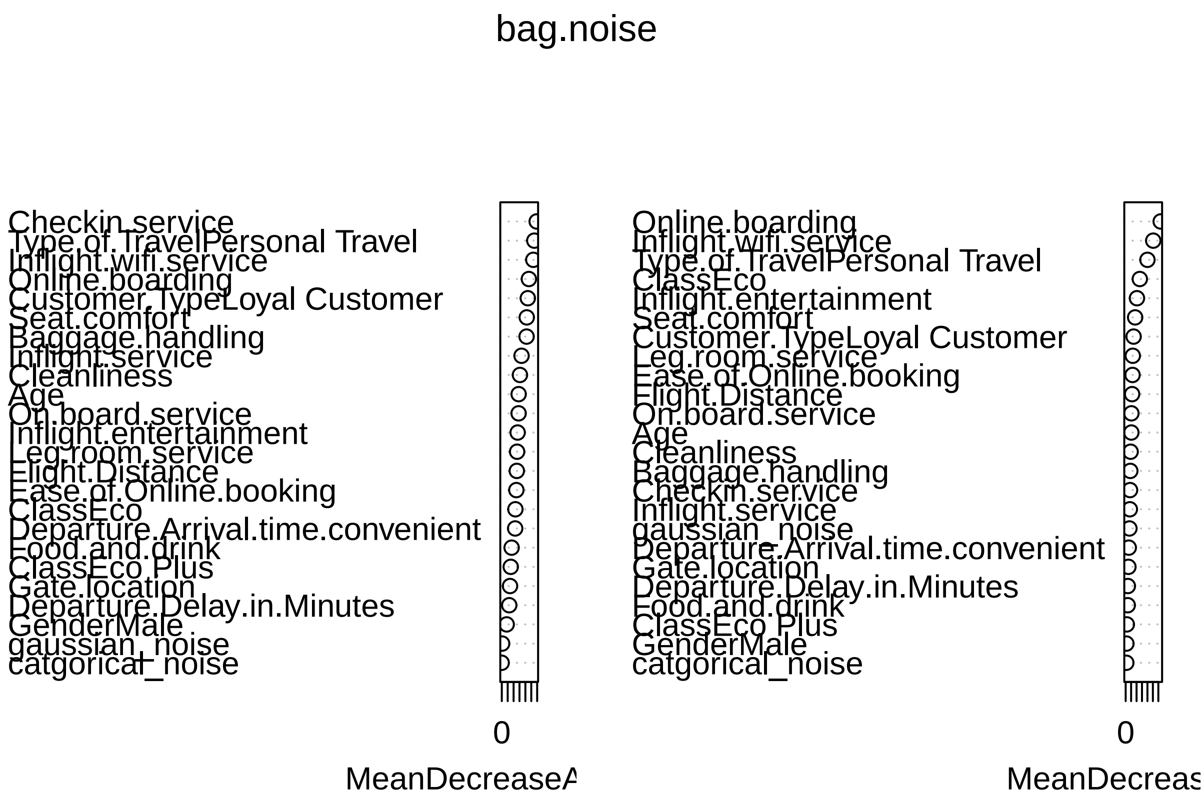
```
##                            y_pred_noise
##                             neutral or dissatisfied satisfied
##   neutral or dissatisfied                     17180       402
##   satisfied                                     792     12705
```

```
(17182+12705)/(17182+12705+792+400)
```

```
## [1] 0.9616461
```

The accuracy is 96.16%.

```
varImpPlot(bag.noise)
```



The top 7 importance feature remains the same, and the noise terms are the least importance, which does not affect the performance of the random forest algorithm.

(3) Repeat XGBoost The code is commented or else it would take too long to knitting to PDF files.

```
# # grid search
# for(i in 1:nrow(hyper_grid)){
#
#   # create parameter list
#   params <- list(
#     eta = hyper_grid$eta[i],
#     max_depth = hyper_grid$max_depth[i],
#     min_child_weight = hyper_grid$min_child_weight[i],
#     subsample = hyper_grid$subsample[i],
#     colsample_bytree = hyper_grid$colsample_bytree[i])
#
#   # reproducibility
#   set.seed(123)
#
#   # train model
#   xgb.tune <- xgb.cv(
#     params = params,
#     data = X_train_noise,
#     label = train_dummy_y_survey,
#     nrounds = 500,
#     nfold = 5,
```

```
#     objective='binary:logistic',
#     eval_metric = "error",
#     verbose = 0,                 # silent,
#     early_stopping_rounds = 10) # stop if no improvement for 10 consecutive trees
#
#
#   # add min training error and trees to grid
#   hyper_grid$optimal_trees[i] <- which.min(xgb.tune$evaluation_log$test_error_mean)
#   hyper_grid$min_error[i] <- min(xgb.tune$evaluation_log$test_error_mean)
# }
#
# best_para_noise <- hyper_grid %>%
#   dplyr::arrange(min_error) %>%
#   head(1)
```

```
# #(best)
# eta=best_para$eta=0.2
# max_depth=best_para$max_depth=5
# min_child_weight=best_para$min_child_weight=6
# subsample=best_para$subsample=0.8
# colsample_bytree=best_para$colsample_bytree=0.9
# optimal_trees=best_para$optimal_trees=148
```

Now feed it into the XGBoost model.

```
# parameter list
params <- list(eta = 0.2,
               max_depth = 5,
               min_child_weight = 6,
               subsample = 0.8,
               colsample_bytree = 0.9)

set.seed(1)
# train final model
modfinal_xgb_noise <- xgboost(params = params,
                              data = X_train_noise,
                              label = train_dummy_y_survey,
                              nrounds = 100,
                              objective='binary:logistic',
                              eval_metric = "error",
                              verbose = 0)
```

```
y_pred <- predict(modfinal_xgb_noise, newdata = X_test_noise)
y_pred <- as.numeric(y_pred > 0.5)
# Confusion Matrix
confusion_mtx = table(test_dummy_y_survey, y_pred)
confusion_mtx
```

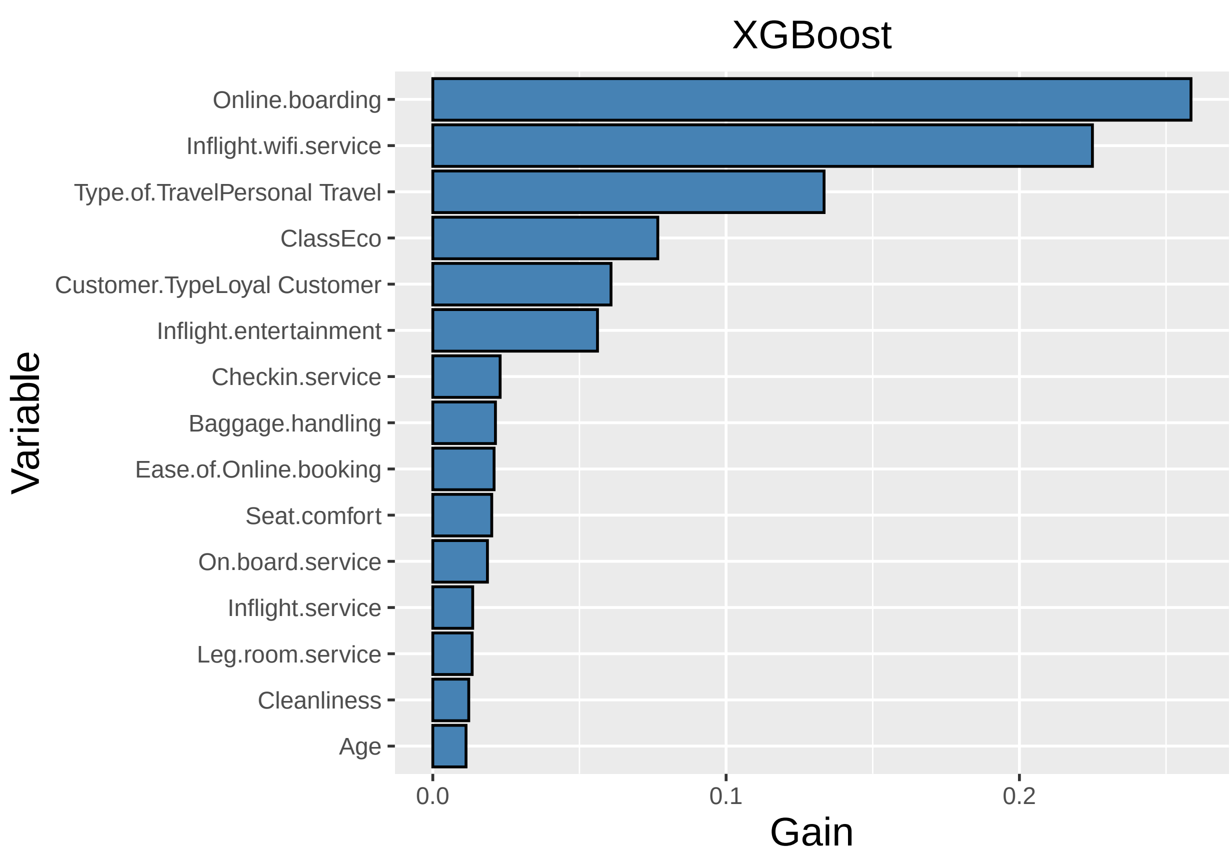
```
##                    y_pred
## test_dummy_y_survey     0     1
##                   0 17165   417
##                   1   840 12657
```

```r
(17165+12657)/(17165+12657+840+417)
```

```
## [1] 0.9595547
```

The accuracy is 95.96%.

```r
xgb.importance(model = modfinal_xgb_noise) %>%
  as.data.frame() %>%
  `colnames<-`(c("Feature",'Gain','Cover',"Frequency")) %>%
  arrange(desc(Gain)) %>%
  top_n(15,wt = Gain) %>%
  ggplot(aes(x = reorder(Feature, Gain),y = Gain)) +
  geom_col(fill = 'steelblue', color = 'black') +
  coord_flip() +
  ggtitle(label = "XGBoost") +
  xlab('Variable')+
  ylab('Gain')+
  theme(plot.title=element_text(hjust=0.5,size=15),
        axis.title=element_text(size=15))
```



The top 6 importance feature are `Online boarding`, `Inflight wifi service`, `Type of Travel`, `Class`, `Customer Type`, `Inflight entertainment`. This is slightly different with the original XGBoost, where `Class` is the 6th most important feature and now becomes 4.

4. Summary

The accuracy of testing set and selected features are presented in the following tables.

```r
knitr::include_graphics("3.png")
```

model	acc
Random Forest	96.29%
XGBoost	96.01%
Random Forest (with noise)	96.16%
XGBoost (with noise)	95.96%

```r
knitr::include_graphics("4.png")
```

Important Features			
Random Forest	XGBoost	Random Forest (with noise)	XGBoost (with noise)
Checkin service	Online boarding	Checkin service	Online boarding
Inflight wifi service	Inflight wifi service	Inflight wifi service	Inflight wifi service
Type of Travel	Type of Travel	Type of Travel	Type of Travel
Online boarding	Customer Type	Online boarding	Class
Seat comfort	Inflight entertainment	Seat comfort	Customer Type
Customer Type		Customer Type	Inflight entertainment
Baggage handling		Baggage handling	

(1) Adding 1 column of Gaussian noise and 1 column of categorical noise does not affect the tree-based model performance much.

(2) To get higher passenger satisfication, the airline company had to improve online boarding, checkin service and wifi service since these three features are the most important. Also, be good to those passengers taking a personal travel. Other measurements such as improve the seat comfort and inflight entertainment would also helps.