hw5.R

asahikuroki222

2021-12-07

# import packages  
library(GGally)

## Loading required package: ggplot2

## Registered S3 method overwritten by 'GGally':  
## method from   
## +.gg ggplot2

library(caret)

## Loading required package: lattice

library(rpart.plot)

## Loading required package: rpart

library(gridExtra)  
library(labelVector)  
library(tidyverse)

## ── Attaching packages ───────────────────────────────────────────────────────────────────────────────────────────── tidyverse 1.3.0 ──

## ✓ tibble 3.0.3 ✓ dplyr 1.0.2  
## ✓ tidyr 1.1.0 ✓ stringr 1.4.0  
## ✓ readr 1.3.1 ✓ forcats 0.5.0  
## ✓ purrr 0.3.4

## ── Conflicts ──────────────────────────────────────────────────────────────────────────────────────────────── tidyverse\_conflicts() ──  
## x dplyr::combine() masks gridExtra::combine()  
## x dplyr::filter() masks stats::filter()  
## x dplyr::lag() masks stats::lag()  
## x purrr::lift() masks caret::lift()

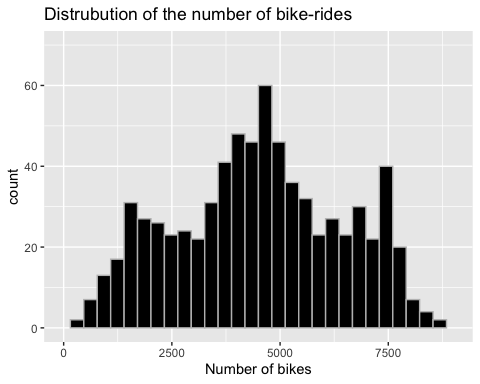
# Analysis   
# Processing and Visualizing data   
# a) Load the data. Get a summary of the data, report it. Use ggplot to plot a histogram for the distribution of the number of bike-rides.  
df <- read.csv("bike\_day.csv")  
summary(df)

## cnt\_bike atemp hum windspeed temp   
## Min. : 22 Min. : 3.95 Min. : 0.00 Min. : 1.50 Min. : 2.42   
## 1st Qu.:3152 1st Qu.:16.89 1st Qu.:52.00 1st Qu.: 9.04 1st Qu.:13.82   
## Median :4548 Median :24.34 Median :62.67 Median :12.13 Median :20.43   
## Mean :4504 Mean :23.72 Mean :62.79 Mean :12.76 Mean :20.31   
## 3rd Qu.:5956 3rd Qu.:30.43 3rd Qu.:73.02 3rd Qu.:15.62 3rd Qu.:26.88   
## Max. :8714 Max. :42.04 Max. :97.25 Max. :34.00 Max. :35.33   
## holiday workingday   
## Min. :0.00000 Min. :0.000   
## 1st Qu.:0.00000 1st Qu.:0.000   
## Median :0.00000 Median :1.000   
## Mean :0.02873 Mean :0.684   
## 3rd Qu.:0.00000 3rd Qu.:1.000   
## Max. :1.00000 Max. :1.000

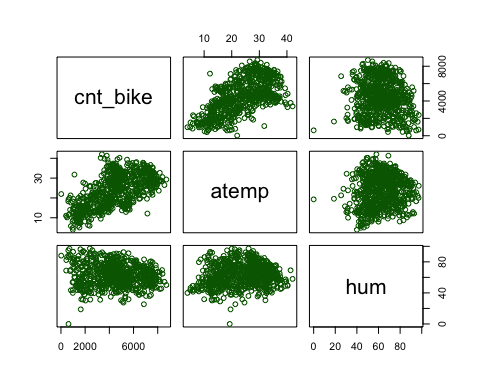
# histogram  
ggplot(data = df, aes(cnt\_bike)) +   
 xlim(0, 9000) +   
 ylim(0, 70) +   
 geom\_histogram(colour = "grey", fill = "black") +   
 ggtitle("Distrubution of the number of bike-rides") +   
 labs(x = "Number of bikes")

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.

## Warning: Removed 2 rows containing missing values (geom\_bar).



# The highest cnt is 8714 and the lowest is 22. It has the most count   
# around the center of the graph (3750 - 6000). Also the highest count   
# is around 60 and most counts are less than 40.   
  
# b) Use the function pairs() to produce a plot of the relationships among count, atemp and hum.   
pairs(df[,1:3], col = "darkgreen")



# bike count (cnt\_bike) and atemp seems to have a positive relationship, while atemp and hum, and cnt\_bike and hum do not have relationship  
  
  
# c) (0.2) Split the data into 80% training and 20% testing.   
trainRows <- createDataPartition(y = df$cnt\_bike, p = 0.8, list = FALSE)  
train\_set <- df[trainRows,]  
test\_set <- df[-trainRows,]  
  
# Train a K-NN model   
# a) Decide whether you need to standardize the data or not   
# A. Yes. We need to standardize the data in K-NN.   
# We will standardize all the attributes besides cnt\_bike because that is our y-value.   
  
train\_set\_stand <- train\_set  
test\_set\_stand <- test\_set  
library(standardize)

##   
## \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   
## Loading standardize package version 0.2.2   
## Call standardize.news() to see new features/changes   
## \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#Apply the standardization  
train\_set\_stand[,2:7] <- apply(train\_set\_stand[,2:7], MARGIN = 2, FUN = scale)  
test\_set\_stand[,2:7]<- apply(test\_set\_stand[,2:7], MARGIN = 2, FUN = scale)  
  
  
# b) Train a k-NN model on the appropriate attributes.  
  
knn\_model <- train(cnt\_bike~., train\_set\_stand, method = "knn")  
knn\_model

## k-Nearest Neighbors   
##   
## 587 samples  
## 6 predictor  
##   
## No pre-processing  
## Resampling: Bootstrapped (25 reps)   
## Summary of sample sizes: 587, 587, 587, 587, 587, 587, ...   
## Resampling results across tuning parameters:  
##   
## k RMSE Rsquared MAE   
## 5 1415.673 0.4864054 1155.578  
## 7 1385.400 0.5029210 1141.224  
## 9 1365.367 0.5158304 1135.913  
##   
## RMSE was used to select the optimal model using the smallest value.  
## The final value used for the model was k = 9.

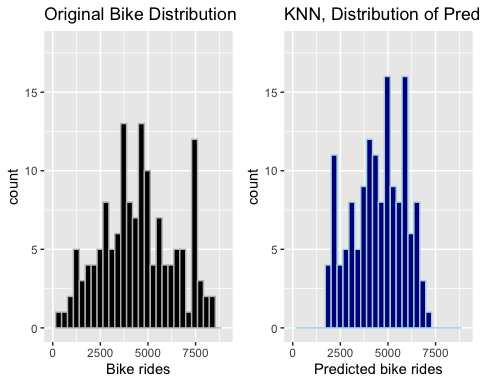
# The algorithm used k = 5, 7, 9   
  
# c) Get the predictions from k-NN model  
knnPred <- predict(knn\_model, test\_set\_stand)  
# create a histogram of the distribution of bike rides  
h\_pred\_knn <- ggplot(data= test\_set\_stand, aes(x = knnPred)) +   
 xlim(0, 9000) +   
 ylim(0, 18) +  
 geom\_histogram(colour = "lightblue", fill = "darkblue") +  
 ggtitle("KNN, Distribution of Predicted bike rides") +  
 labs(x = "Predicted bike rides")  
  
bike\_dist<- ggplot(data=test\_set\_stand, aes(x = cnt\_bike)) +   
 geom\_histogram(colour = "grey", fill = "black") +  
 xlim (0,9000) +   
 ylim (0,18) +   
 ggtitle("Original Bike Distribution") +  
 labs(x = "Bike rides")  
  
grid.arrange(bike\_dist, h\_pred\_knn, nrow=1)

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.

## Warning: Removed 2 rows containing missing values (geom\_bar).

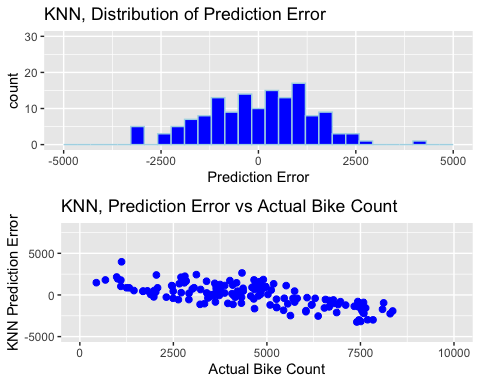
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.

## Warning: Removed 2 rows containing missing values (geom\_bar).



# I think the performance of K-NN is OK. Even though each counts are not   
# the same, the shape of overall graph looks kind of similar.   
  
#d) Computer the prediction error for the k-NN model  
# create ggolit histrogram for the error  
knn\_error <-knnPred - test\_set\_stand$cnt\_bike  
  
#Visualize the prediction error  
#Histogram of the distribution of the prediction error  
h\_error\_knn = ggplot(data= test\_set\_stand, aes(x = knn\_error)) +   
 geom\_histogram(colour = "lightblue", fill = "blue") +  
 xlim (-5000, 5000) +   
 ylim (0, 30) +   
 ggtitle("KNN, Distribution of Prediction Error") +  
 labs(x = "Prediction Error")  
  
  
#Plot prediction error vs actual price  
p\_error\_knn<- ggplot(data = test\_set\_stand, aes(x=cnt\_bike, y=knn\_error)) +  
 geom\_point(size=2, color = "blue") +  
 ylim (-5000, 8000) +  
 xlim (0, 10000) +  
 ggtitle("KNN, Prediction Error vs Actual Bike Count") +  
 labs(x = "Actual Bike Count", y = "KNN Prediction Error")  
  
grid.arrange(h\_error\_knn, p\_error\_knn)

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



# It seems like there are more negative errors by looking at   
# the distribution of Prediction Error. Also, KNN have more positive   
# error when the actual bike count is < 5000 and negative error when   
# actual bike count is > 5000  
# We could say K-NN is under-predicting.   
  
#e)  
knnME <- mean(knn\_error)  
knnME

## [1] -20.2392

knnRMSE<- RMSE(pred = knnPred, obs = test\_set\_stand$cnt\_bike)  
knnRMSE

## [1] 1359.406

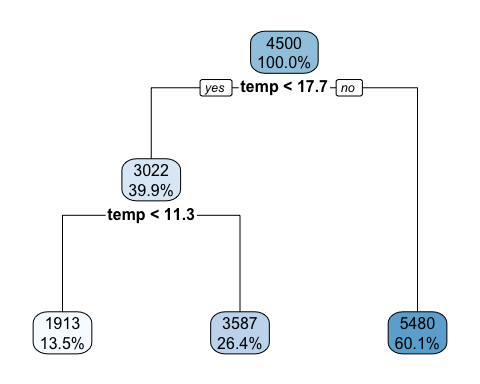
# ME of -20 tells us that on average we are under-predicting by about 20.   
# RMSE of 1359 tells us that on average our prediction is off by 1359 bike counts  
  
# Train a Regression Tree  
#f) Decide whether to standardize.   
# No. We do not have to standarize the data in regression tree.   
  
#g) Train a regression tree  
rtree <- train(cnt\_bike~., train\_set, method = "rpart")

## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainInfo, :  
## There were missing values in resampled performance measures.

rtree

## CART   
##   
## 587 samples  
## 6 predictor  
##   
## No pre-processing  
## Resampling: Bootstrapped (25 reps)   
## Summary of sample sizes: 587, 587, 587, 587, 587, 587, ...   
## Resampling results across tuning parameters:  
##   
## cp RMSE Rsquared MAE   
## 0.05290532 1480.938 0.4183129 1233.859  
## 0.06659281 1512.564 0.3928347 1267.285  
## 0.38629891 1703.105 0.3450741 1419.709  
##   
## RMSE was used to select the optimal model using the smallest value.  
## The final value used for the model was cp = 0.05290532.

# Plot the final tree   
rpart.plot(rtree$finalModel, digits=-3)



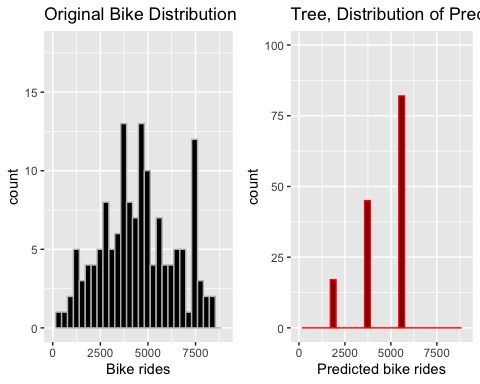
# The algorithm picked temp < 17.7, temp < 11.3 for the attributes.   
  
# h) Get the predictions from the regression tree and use ggplot   
# to create a histogram of the distribution of the predicted bike rides  
# compare it to the histogram of the true count  
treePred <- predict(rtree, test\_set)  
h\_pred\_tree<- ggplot(data= test\_set, aes(x = treePred)) +   
 geom\_histogram(colour = "red", fill = "darkred") +  
 xlim (0,9000) +   
 ylim (0, 100) +   
 ggtitle("Tree, Distribution of Predictions") +  
 labs(x = "Predicted bike rides")  
  
#compare to the actual price distribution we created above  
grid.arrange(bike\_dist,h\_pred\_tree, nrow=1)

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.

## Warning: Removed 2 rows containing missing values (geom\_bar).

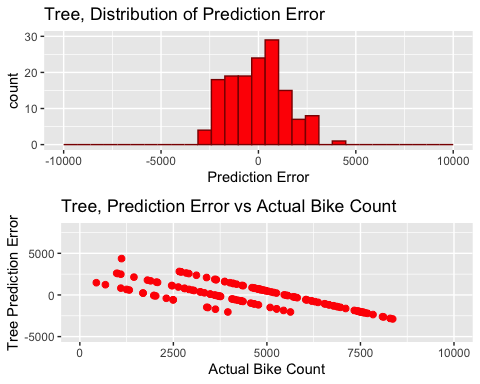
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.

## Warning: Removed 2 rows containing missing values (geom\_bar).



# Regression tree only produced three bars in the graph.   
# Just by looking at the graph, regression tree does not seem to be performing well because it only has three bars while the original one has a lot. It does not capture all the details.   
  
# i) Compute the prediction error for the regression tree and   
# create a ggplot histogram for the prediction error  
  
# Prediction error  
tree\_error <-treePred - test\_set$cnt\_bike  
h\_error\_tree<- ggplot(data= test\_set, aes(x = tree\_error)) +   
 geom\_histogram(colour = "darkred", fill = "red") +  
 xlim (-10000, 10000) +   
 ylim (0, 30) +   
 ggtitle("Tree, Distribution of Prediction Error") +  
 labs(x = "Prediction Error")  
  
#Plot prediction error vs actual price  
p\_error\_tree<- ggplot(data = test\_set, aes(x=cnt\_bike, y=tree\_error)) +  
 geom\_point(size=2, color = "red") +  
 ylim (-5000, 8000) +  
 xlim (0, 10000) +  
 ggtitle("Tree, Prediction Error vs Actual Bike Count") +  
 labs(x = "Actual Bike Count", y = "Tree Prediction Error")  
  
grid.arrange(h\_error\_tree, p\_error\_tree)

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



# The distribution of Prediction Error looks like there might be a little more error on the left side.  
# It is interesting how we could draw three lines in this scatter plot, and those would represent the three bars in “Distribution of prediction” graph. I think the algorithm is doing a good job not over-Estimating nor under-estimating too much.   
  
# i,b) Compute the ME and RMSE for the regression tree  
ME\_tree <- mean(tree\_error)  
treeRMSE <- RMSE(pred = treePred, obs = test\_set$cnt\_bike)  
ME\_tree

## [1] -53.61088

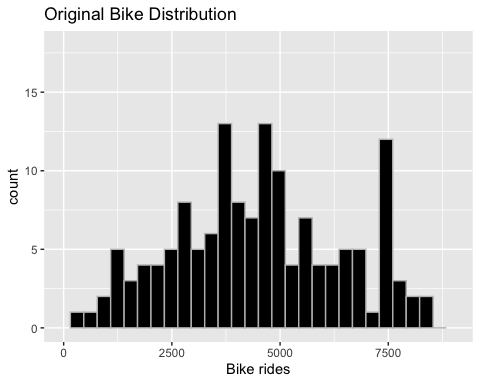
treeRMSE

## [1] 1435.095

# ME of -53 means that on average RegressionTree under predicts  
# by 53   
  
# RMSE of 1435 meants that on average RegressionTree's prediction   
# is off by 1435 bike counts  
  
  
  
  
# Train a Linear Regression  
#a) Decide whether you need to standardize the data  
A. No. I do not have to use standardized data in Linear Regression  
  
#b) Check and comment on whether using the attributes used for the prediction.   
bike\_dist

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.

## Warning: Removed 2 rows containing missing values (geom\_bar).



# A. Yes. There are some outliers, but it is somewhat normally distributed  
  
#c) Create a correlation matrix using the attributes used for the prediction,   
cor(df[,c(2:7)])

## atemp hum windspeed temp holiday  
## atemp 1.00000000 0.14000432 -0.183668766 0.99169797 -0.032502593  
## hum 0.14000432 1.00000000 -0.248509797 0.12695001 -0.015927598  
## windspeed -0.18366877 -0.24850980 1.000000000 -0.15792514 0.006288675  
## temp 0.99169797 0.12695001 -0.157925143 1.00000000 -0.028556898  
## holiday -0.03250259 -0.01592760 0.006288675 -0.02855690 1.000000000  
## workingday 0.05215699 0.02432579 -0.018791911 0.05267624 -0.253022700  
## workingday  
## atemp 0.05215699  
## hum 0.02432579  
## windspeed -0.01879191  
## temp 0.05267624  
## holiday -0.25302270  
## workingday 1.00000000

# I will exclude atemp from the attributes because it is highly correlated   
# with temp (0.99169).   
train\_set\_lr <- train\_set %>% select(1:1, 3:7)  
test\_set\_lr <- test\_set %>% select(1:1, 3:7)  
  
# d) Train a linear regression model   
lin\_reg <- train(cnt\_bike~., train\_set\_lr, method = "lm")  
lin\_reg

## Linear Regression   
##   
## 587 samples  
## 5 predictor  
##   
## No pre-processing  
## Resampling: Bootstrapped (25 reps)   
## Summary of sample sizes: 587, 587, 587, 587, 587, 587, ...   
## Resampling results:  
##   
## RMSE Rsquared MAE   
## 1437.873 0.4553024 1184.21  
##   
## Tuning parameter 'intercept' was held constant at a value of TRUE

#Summarize final model  
fit <- lin\_reg$finalModel  
options(scipen = 999) #this is to avoid scientific notation  
summary(fit)

##   
## Call:  
## lm(formula = .outcome ~ ., data = dat)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -4917.8 -1075.7 -96.7 1070.7 3635.1   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 4053.062 392.155 10.335 < 0.0000000000000002 \*\*\*  
## hum -32.192 4.338 -7.422 0.000000000000413 \*\*\*  
## windspeed -67.318 11.863 -5.674 0.000000021967255 \*\*\*  
## temp 163.722 8.058 20.318 < 0.0000000000000002 \*\*\*  
## holiday -727.827 371.818 -1.957 0.0508 .   
## workingday 57.799 129.667 0.446 0.6559   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 1420 on 581 degrees of freedom  
## Multiple R-squared: 0.4681, Adjusted R-squared: 0.4636   
## F-statistic: 102.3 on 5 and 581 DF, p-value: < 0.00000000000000022

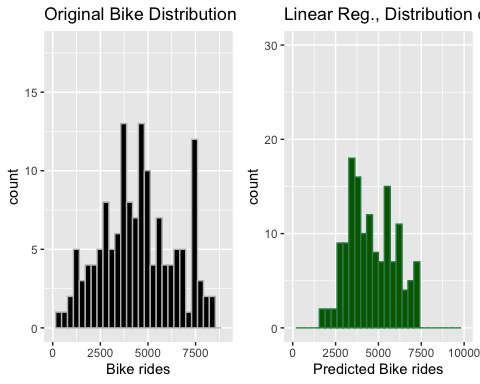
# j) Get the predictions from the linear regression model and   
# use ggplot to create a histogram of the distribution of the predicted   
# bike rides  
  
lin\_pred <- predict(lin\_reg, newdata = test\_set\_lr)  
  
#Visualize the predictions  
#Create a histogram for the distribution of predicted prices  
h\_pred\_lm <- ggplot(data= test\_set\_lr, aes(x = lin\_pred)) +   
 geom\_histogram(colour = "seagreen", fill = "darkgreen") +  
 xlim (0,10000) +   
 ylim (0, 30) +   
 ggtitle("Linear Reg., Distribution of Predictions") +  
 labs(x = "Predicted Bike rides")  
  
#compare to the actual price distribution  
grid.arrange(bike\_dist, h\_pred\_lm, nrow = 1)

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.

## Warning: Removed 2 rows containing missing values (geom\_bar).

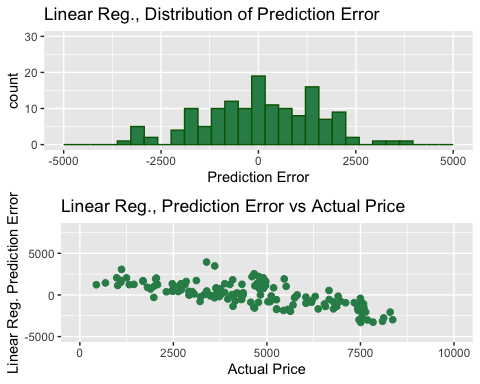
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.

## Warning: Removed 2 rows containing missing values (geom\_bar).



# By looking at the graph it seems like linear regression   
# model is doing a good job because the graph has similar   
# overall shape. But it does not capture all the details.   
  
#k) Compute the prediction error for the linear regression model and create   
# a ggplot histogram for the distribute of the prediction error  
#Compute Prediction error  
lm\_error <- lin\_pred - test\_set\_lr$cnt\_bike  
#Visualize the prediction error  
#Histogram of the distribution of prediction errors  
h\_error\_lm <- ggplot(data= test\_set\_lr, aes(x = lm\_error)) +   
 geom\_histogram(colour = "darkgreen", fill = "seagreen") +  
 xlim (-5000, 5000) +   
 ylim (0, 30) +   
 ggtitle("Linear Reg., Distribution of Prediction Error") +  
 labs(x = "Prediction Error")  
  
#Plot of the Prediction Error vs Actual Price  
p\_error\_lm<- ggplot(data = test\_set\_lr, aes(x=cnt\_bike, y=lm\_error)) +  
 geom\_point(size=2, color = "seagreen") +  
 ylim (-5000, 8000) +  
 xlim (0, 10000) +  
 ggtitle("Linear Reg., Prediction Error vs Actual Price") +  
 labs(x = "Actual Price", y = "Linear Reg. Prediction Error")  
  
grid.arrange(h\_error\_lm, p\_error\_lm)

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



# By looking at these two graphs we can say that linear   
# regression is doing a good job not over predicting nor   
# under-predicting.   
# Also the model tends to over predict when actual price is < 5000 and under-predict when actual price is > 5000.   
#e)   
ME\_lin <- mean(lm\_error)  
#RMSE  
lin\_RMSE <- RMSE(pred = lin\_pred, obs = test\_set\_lr$cnt\_bike)  
ME\_lin

## [1] 61.8804

lin\_RMSE

## [1] 1440.201

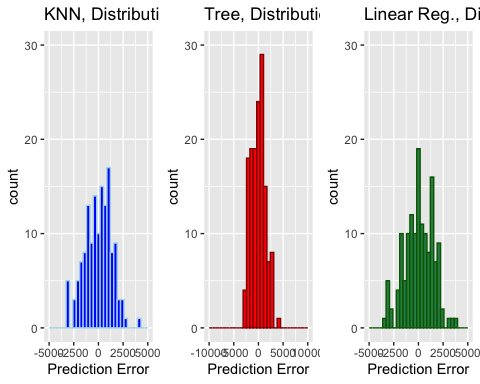
# ME of 61 means that on average Linear Regression model is   
# underpredicting by 61.   
# RMSE of 1440 means that on average the model is off by   
# 1440 bike counts  
######  
# Product Insights  
#Put together the error metrics  
  
error\_table <- c(knnME, knnRMSE, ME\_tree, treeRMSE, ME\_lin, lin\_RMSE)  
names(error\_table) <- c("KNN ME", "KNN RMSE", "TREE ME", "TREE RMSE", "LR ME", "LR RMSE")  
error\_table <- set\_label(error\_table, "Error table")  
error\_table

## Error table  
## KNN ME KNN RMSE TREE ME TREE RMSE LR ME LR RMSE   
## -20.23920 1359.40552 -53.61088 1435.09505 61.88040 1440.20087

# Report the histogram for the distribution of the prediction errors  
grid.arrange(h\_error\_knn,h\_error\_tree,h\_error\_lm, nrow = 1)

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



# I would suggest the company to implement the K-NN model. This is because the K-NN model has lowest abs (ME) and lowest RMSE. This means K-NN model did the best job not overpredicting nor underpredicting as well as minimizing the overall error. I would not suggest the company to use Regression Tree model and Linear Regression model because ME and RMSE are higher than K-NN’s ME and RMSE.

Also in the histogram, Regression tree has higher Prediction Error count because the model only has 3 values to return. We could see that from Regression Tree’s distribution of prediction.

K-NN might did a better job than Regression Tree and Linear Regression because the data points were pretty close to each other and there weren’t that many outliers.