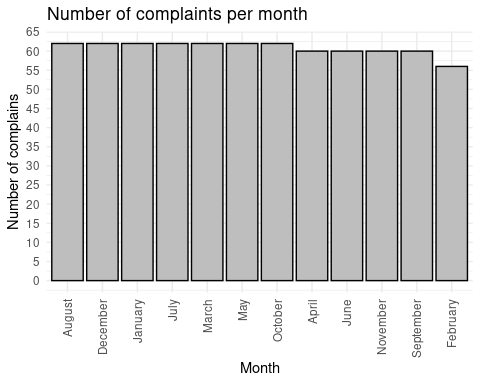
Data Minning Coursework

Boluwatife Sulaimon 2114578

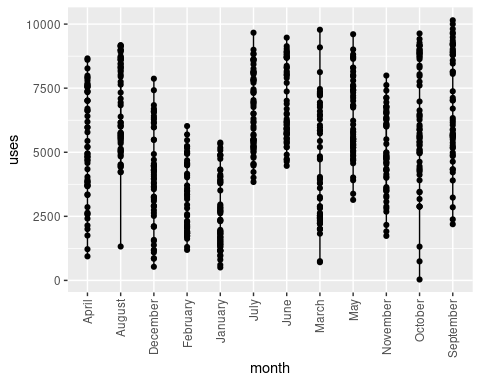
11/17/2021

set.seed(123)  
# read csv file  
ride4U <- read.csv("ride4U.csv")  
  
# convert data.frame to tibble  
ride4U <- tibble::as\_tibble(ride4U)

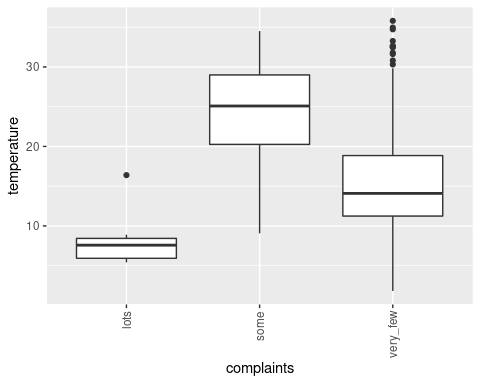
## Univariate analysis  
ggplot(ride4U) + geom\_bar(aes(fct\_infreq(month)), color = "black", fill = "grey") + labs(title = "Number of complaints per month", x = "Month", y = "Number of complains") + scale\_y\_continuous(breaks = seq(0, 65, 5)) + theme\_minimal() + theme(axis.text.x = element\_text(angle = 90, vjust = 0.5, hjust=1))



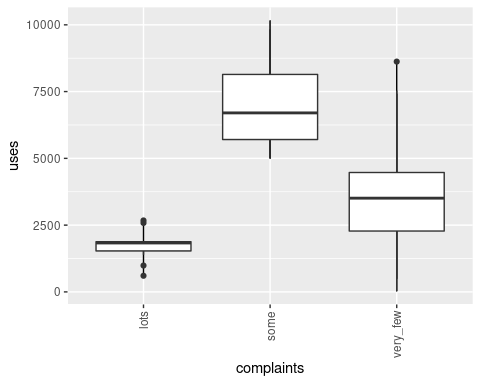
# compare information for extreme months  
compr4U <- ride4U[ride4U$month == "February" | ride4U$month == "August",]  
ride4U %>% ggplot(aes(month, uses)) + geom\_line() + geom\_point()+ theme(axis.text.x = element\_text(angle = 90, vjust = 0.5, hjust=1))

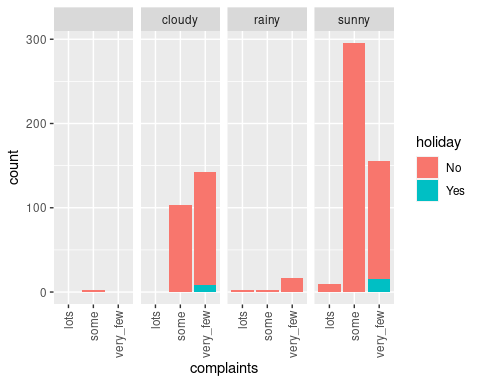


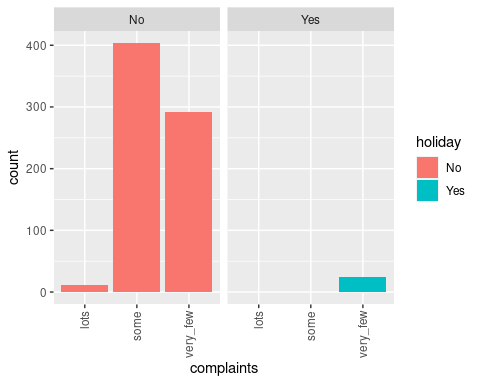
ride4U %>% ggplot(aes(complaints, temperature)) + geom\_boxplot()+ theme(axis.text.x = element\_text(angle = 90, vjust = 0.5, hjust=1))



## compare effect of variables on level of complaint  
  
ride4U %>% ggplot(aes(complaints, uses)) + geom\_line() + geom\_boxplot()+ theme(axis.text.x = element\_text(angle = 90, vjust = 0.5, hjust=1))

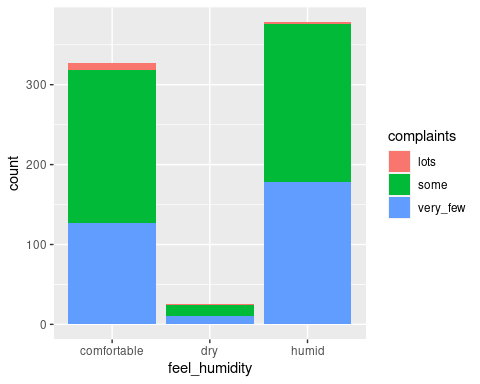


ride4U %>% ggplot(aes(complaints)) + geom\_bar(aes(fill = holiday)) + facet\_grid(~holiday)+ theme(axis.text.x = element\_text(angle = 90, vjust = 0.5, hjust=1))

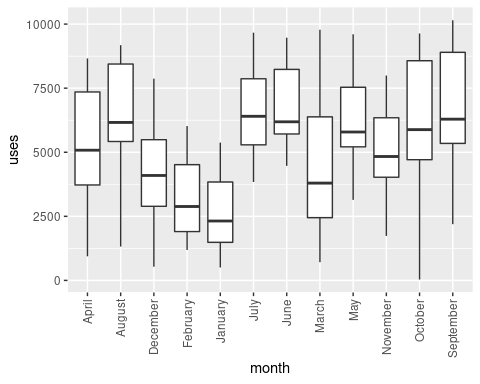


ride4U %>% ggplot(aes(complaints)) + geom\_bar(aes(fill = holiday)) + facet\_grid(~outlook)+ theme(axis.text.x = element\_text(angle = 90, vjust = 0.5, hjust=1))

ride4U %>% ggplot(aes(feel\_humidity)) + geom\_bar(aes(fill = complaints))



ride4U %>% ggplot(aes(x=month, y=uses)) + geom\_boxplot() + theme(axis.text.x = element\_text(angle = 90, vjust = 0.5, hjust=1))



**1. Univariante and Bivariant Analysis**

Exploratory data analysis reveals that the level of complaints across most months is around the same level, with a similar level of complaints across seven months. The least amount of complaints was recorded in February while the most complaints were recorded in August; this information informed further **univariate analysis**.

**Bivariate analysis** reveals that fewer complaints are lodged on holidays compared to days that aren’t holidays. It was also noted that a sunny outlook resulted in more complaints. Generally, lower temperature results in a lot of complaints.

Temperatures in February(Average value of 12.46) are generally much lower than temperature ranges in August(Average value of 29.08). It was also observed that most of the bikes used by customers in August were bikes with more usage which has been observed to impact the complaint level.

## Using Ride4U to obtain three further datasets  
ride4U40 <- sample\_frac(ride4U, 0.4)  
ride4U20 <- sample\_frac(ride4U40, 0.5)

1. **Advantages and disadvantages of using a reduced dataset Subsampling** is a technique used to reduce the amount of population to a fraction that is ideally representative of the population; samples can then be obtained from the sample population to obtain characteristics that could be used to estimate the population’s parameter, reducing computational expenses. One disadvantage is that in situations where the sub-sample is not representative of the population, insight obtained could be incorrect when used to estimate population parameters.

## Corrupt 15% of datasets  
dodgyride4U <- ride4U  
  
# chose an average of 15% to corrupt at random  
corrupt <- rbinom(nrow(ride4U), 1, 0.15)  
corrupt <- as.logical(corrupt)  
  
# generate the noise to add to outlook  
vals <- levels(ride4U$outlook)  
noise <- sample(ride4U$outlook, length(ride4U$outlook) - 1, replace = TRUE)  
dodgyride4U$outlook[corrupt] <- noise[corrupt]  
  
# generate noise for selected instances for temperature  
noise <- rnorm(corrupt, median(ride4U$temperature), sd(ride4U$temperature))

## Warning in rnorm(corrupt, median(ride4U$temperature), sd(ride4U$temperature)):  
## NAs produced

dodgyride4U$temperature[corrupt] <- as.integer(noise[corrupt])

1. Insight Obtainable from Using a dataset with Noise means all unreadable data at the percentage given should be generated

preprocess\_dataframe <- function(df) {  
 df <- na.omit(df)  
 df$label <- case\_when(  
 df$complaints %in% c('very\_few') ~ '0',  
 df$complaints %in% c('some') ~ '1',  
 df$complaints %in% c('lots') ~ '2',  
 )  
 df <- subset(df, select= -c(complaints, country))  
 df <- as.data.frame(unclass(df), stringsAsFactors = TRUE)  
   
 return (df)  
}  
  
## Preparing the datasets for classification  
prepare\_dataframe <- function(df) {  
 intrain <- createDataPartition(y = df$label, p=0.7, list = FALSE)  
 return (intrain)  
}

run\_tree\_classifier <- function(df, ctrl) {  
 tree <- train(df[,1:12], df$label, method = "rpart", tuneLength = 12, trControl = ctrl)  
 return (tree)  
}  
  
  
run\_instance\_classifier <- function(df, ctrl) {  
 knn <- train(label ~ ., data = df, method = "knn", trControl = ctrl, preProcess = c("center", "scale"), tuneLength = 20)  
 return (knn)  
}

ride4U <- preprocess\_dataframe(ride4U)  
ride4U20 <- preprocess\_dataframe(ride4U20)  
ride4U40 <- preprocess\_dataframe(ride4U40)  
dodgyride4U <- preprocess\_dataframe(dodgyride4U)  
  
r4uout <- prepare\_dataframe(ride4U)  
r4u20out <- prepare\_dataframe(ride4U20)  
r4u40out <- prepare\_dataframe(ride4U40)  
r4udodgyout <- prepare\_dataframe(dodgyride4U)  
  
r4ut <- as.data.frame(ride4U[r4uout,])  
r4ute <- as.data.frame(ride4U[-r4uout, ])  
r4u20t <- as.data.frame(ride4U20[r4u20out,])  
r4u20te <- as.data.frame(ride4U20[-r4u20out,])  
r4u40t <- as.data.frame(ride4U40[r4u40out,])  
r4u40te <- as.data.frame(ride4U40[-r4u40out,])  
r4udodgyt <- as.data.frame(dodgyride4U[r4udodgyout,])  
r4udodgyte <- as.data.frame(dodgyride4U[-r4udodgyout, ])  
  
  
ctrl <- trainControl(method = "repeatedcv", number = 3, repeats = 3, verboseIter=TRUE)

r4uc <- confusionMatrix.train(ride4Uclfr, norm="none")  
print("ride4U confusion Matrix")

## [1] "ride4U confusion Matrix"

r4uc

## Cross-Validated (3 fold, repeated 3 times) Confusion Matrix   
##   
## (entries are un-normalized aggregated counts)  
##   
## Reference  
## Prediction 0 1 2  
## 0 663 0 21  
## 1 0 840 0  
## 2 0 0 3  
##   
## Accuracy (average) : 0.9862

r4u20c <- confusionMatrix.train(ride4U20clfr, norm="none")  
print("ride4U20 confusion Matrix")

## [1] "ride4U20 confusion Matrix"

r4u20c

## Cross-Validated (3 fold, repeated 3 times) Confusion Matrix   
##   
## (entries are un-normalized aggregated counts)  
##   
## Reference  
## Prediction 0 1 2  
## 0 125 3 6  
## 1 1 174 0  
## 2 0 0 0  
##   
## Accuracy (average) : 0.9676

r4u40c <- confusionMatrix.train(ride4U40clfr, norm="none")  
print("ride4U40 confusio Matrix")

## [1] "ride4U40 confusio Matrix"

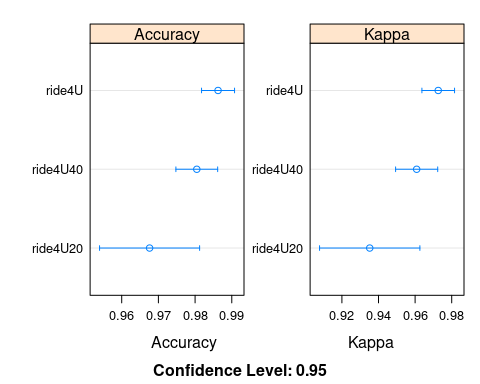
r4u40c

## Cross-Validated (3 fold, repeated 3 times) Confusion Matrix   
##   
## (entries are un-normalized aggregated counts)  
##   
## Reference  
## Prediction 0 1 2  
## 0 261 0 9  
## 1 3 342 0  
## 2 0 0 0  
##   
## Accuracy (average) : 0.9805

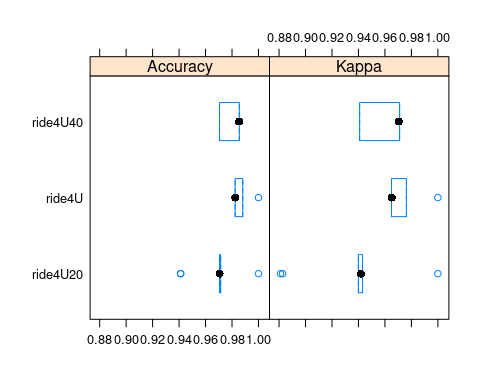
# collect resamples  
results <- resamples(list(ride4U = ride4Uclfr, ride4U20 = ride4U20clfr, ride4U40 = ride4U40clfr))

summary(results)

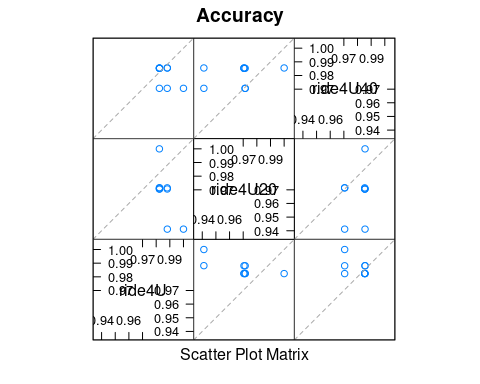
scales <- list(x=list(relation="free"), y=list(relation="free"))  
dotplot(results, scales=scales, conf.level = 0.95)



bwplot(results)



splom(results)



1. Tree classifier discussion Using confusion matrix on the model reveals that models generated using ride4U, ride4U20 and ride4U40 dataset did not show variation in terms of accuracy, with the model obtained with the ride4U dataset having an accuracy of 98.6% and ride4U40 with an accuracy of 98.1% and ride4U20 with an accuracy of 96.7%, The difference between the models are statistically significant as their confidence intervals do not overall. The model obtained from using the ride4U20 dataset appears to be the best model as it classifies the complaint more accurately, only misclassifying 10 instance, compared to the other classes ride4U and ride4U40 misclassifying 24 and 12 instances respectively.

# building instance classifier  
ride4Uinstclf <- run\_instance\_classifier(r4ut, ctrl)

dodgyride4Uinstclf <- run\_instance\_classifier(r4udodgyt, ctrl)

## [1] "ride4U tree model confusion matrix"

## Cross-Validated (3 fold, repeated 3 times) Confusion Matrix   
##   
## (entries are un-normalized aggregated counts)  
##   
## Reference  
## Prediction 0 1 2  
## 0 663 0 21  
## 1 0 840 0  
## 2 0 0 3  
##   
## Accuracy (average) : 0.9862

## [1] "dodgyride4U tree model confusion matrix"

## Cross-Validated (3 fold, repeated 3 times) Confusion Matrix   
##   
## (entries are un-normalized aggregated counts)  
##   
## Reference  
## Prediction 0 1 2  
## 0 581 0 24  
## 1 4 738 0  
## 2 0 0 0  
##   
## Accuracy (average) : 0.9792

## [1] "ride4U instance model confusion matrix"

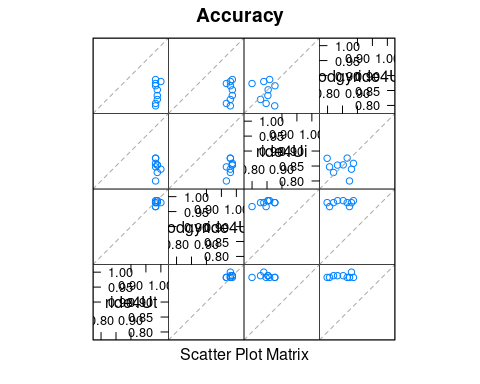
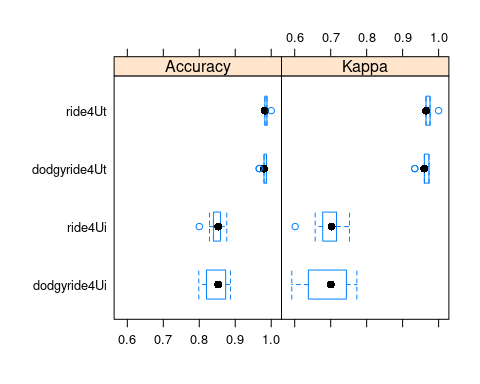
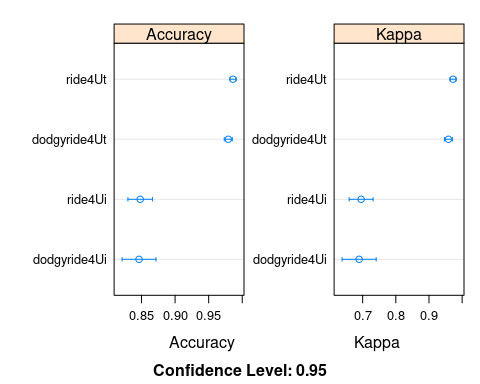
## Cross-Validated (3 fold, repeated 3 times) Confusion Matrix   
##   
## (entries are un-normalized aggregated counts)  
##   
## Reference  
## Prediction 0 1 2  
## 0 537 82 24  
## 1 126 758 0  
## 2 0 0 0  
##   
## Accuracy (average) : 0.8481

## [1] "dodgyride4U instance model confusion matrix"

## Cross-Validated (3 fold, repeated 3 times) Confusion Matrix   
##   
## (entries are un-normalized aggregated counts)  
##   
## Reference  
## Prediction 0 1 2  
## 0 443 41 24  
## 1 142 697 0  
## 2 0 0 0  
##   
## Accuracy (average) : 0.8463

##   
## Call:  
## resamples.default(x = list(ride4Ut = ride4Uclfr, dodgyride4Ut  
## = dodgyride4Uclfr, ride4Ui = ride4Uinstclf, dodgyride4Ui = dodgyride4Uinstclf))  
##   
## Models: ride4Ut, dodgyride4Ut, ride4Ui, dodgyride4Ui   
## Number of resamples: 9   
## Performance metrics: Accuracy, Kappa   
## Time estimates for: everything, final model fit

##   
## Call:  
## summary.resamples(object = results)  
##   
## Models: ride4Ut, dodgyride4Ut, ride4Ui, dodgyride4Ui   
## Number of resamples: 9   
##   
## Accuracy   
## Min. 1st Qu. Median Mean 3rd Qu. Max. NA's  
## ride4Ut 0.9823529 0.9823529 0.9824561 0.9862586 0.9880952 1.0000000 0  
## dodgyride4Ut 0.9666667 0.9800000 0.9800000 0.9792294 0.9865772 0.9865772 0  
## ride4Ui 0.8000000 0.8392857 0.8529412 0.8480581 0.8588235 0.8764706 0  
## dodgyride4Ui 0.7986577 0.8200000 0.8523490 0.8463087 0.8733333 0.8866667 0  
##   
## Kappa   
## Min. 1st Qu. Median Mean 3rd Qu. Max. NA's  
## ride4Ut 0.9649557 0.9649557 0.9651187 0.9726362 0.9761702 1.0000000 0  
## dodgyride4Ut 0.9337573 0.9603734 0.9603734 0.9587363 0.9732063 0.9732063 0  
## ride4Ui 0.6012418 0.6778409 0.7025268 0.6954355 0.7163121 0.7524272 0  
## dodgyride4Ui 0.5918554 0.6379403 0.7006939 0.6893615 0.7436589 0.7727273 0



ride4Uinstclfp <- predict(ride4Uinstclf, newdata = r4ute)  
r4Uic <- confusionMatrix(ride4Uinstclfp, r4ute$label)  
print("confusion matrix for ride4U knn")

## [1] "confusion matrix for ride4U knn"

print(r4Uic)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1 2  
## 0 74 13 3  
## 1 20 107 0  
## 2 0 0 0  
##   
## Overall Statistics  
##   
## Accuracy : 0.8341   
## 95% CI : (0.7778, 0.881)  
## No Information Rate : 0.553   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.666   
##   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: 0 Class: 1 Class: 2  
## Sensitivity 0.7872 0.8917 0.00000  
## Specificity 0.8699 0.7938 1.00000  
## Pos Pred Value 0.8222 0.8425 NaN  
## Neg Pred Value 0.8425 0.8556 0.98618  
## Prevalence 0.4332 0.5530 0.01382  
## Detection Rate 0.3410 0.4931 0.00000  
## Detection Prevalence 0.4147 0.5853 0.00000  
## Balanced Accuracy 0.8286 0.8427 0.50000

dodgyride4Uinstclfp <- predict(dodgyride4Uinstclf, newdata = r4udodgyte)  
r4udic <- confusionMatrix(dodgyride4Uinstclfp, r4udodgyte$label)  
print("confusion Matrix for dodgy ride4U knn")

## [1] "confusion Matrix for dodgy ride4U knn"

print(r4udic)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1 2  
## 0 61 3 3  
## 1 22 102 0  
## 2 0 0 0  
##   
## Overall Statistics  
##   
## Accuracy : 0.8534   
## 95% CI : (0.7951, 0.9003)  
## No Information Rate : 0.5497   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.7012   
##   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: 0 Class: 1 Class: 2  
## Sensitivity 0.7349 0.9714 0.00000  
## Specificity 0.9444 0.7442 1.00000  
## Pos Pred Value 0.9104 0.8226 NaN  
## Neg Pred Value 0.8226 0.9552 0.98429  
## Prevalence 0.4346 0.5497 0.01571  
## Detection Rate 0.3194 0.5340 0.00000  
## Detection Prevalence 0.3508 0.6492 0.00000  
## Balanced Accuracy 0.8397 0.8578 0.50000

1. Evaluation of effect of introduction of noise to classifier

Using a confusion matrix for the tree classifier model, it was observed that the introduction of noise to the ride4U dataset reduces the performance of the tree classifier, as it is observed that the accuracy of the decision tree model reduces the accuracy from 98.6% to 97.9%. It was also observed that compared to the model created with the ride4U dataset, the noisy dataset causes the model to misclassify a total of 30 instances compared to 26 misclassification by the ride4U model. The differences between the tree models isn’t significant as the confidence interval overlaps Similar effect was also observed in the instance classifier. The accuracy drops to 84.6% from 84.8%, the noisy model also misclassifying 207 instances compared to 232, making it the more accurate model.The differences between the tree models isn’t significant as the confidence interval overlaps

## [1] "Instance"

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1 2  
## 0 75 51 3  
## 1 12 222 0  
## 2 0 0 0  
##   
## Overall Statistics  
##   
## Accuracy : 0.8182   
## 95% CI : (0.7746, 0.8565)  
## No Information Rate : 0.7521   
## P-Value [Acc > NIR] : 0.001632   
##   
## Kappa : 0.5772   
##   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: 0 Class: 1 Class: 2  
## Sensitivity 0.8621 0.8132 0.000000  
## Specificity 0.8043 0.8667 1.000000  
## Pos Pred Value 0.5814 0.9487 NaN  
## Neg Pred Value 0.9487 0.6047 0.991736  
## Prevalence 0.2397 0.7521 0.008264  
## Detection Rate 0.2066 0.6116 0.000000  
## Detection Prevalence 0.3554 0.6446 0.000000  
## Balanced Accuracy 0.8332 0.8399 0.500000

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1 2  
## 0 75 38 3  
## 1 12 235 0  
## 2 0 0 0  
##   
## Overall Statistics  
##   
## Accuracy : 0.854   
## 95% CI : (0.8134, 0.8887)  
## No Information Rate : 0.7521   
## P-Value [Acc > NIR] : 1.429e-06   
##   
## Kappa : 0.6453   
##   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: 0 Class: 1 Class: 2  
## Sensitivity 0.8621 0.8608 0.000000  
## Specificity 0.8514 0.8667 1.000000  
## Pos Pred Value 0.6466 0.9514 NaN  
## Neg Pred Value 0.9514 0.6724 0.991736  
## Prevalence 0.2397 0.7521 0.008264  
## Detection Rate 0.2066 0.6474 0.000000  
## Detection Prevalence 0.3196 0.6804 0.000000  
## Balanced Accuracy 0.8568 0.8637 0.500000

## [1] "Tree"

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1 2  
## 0 87 0 0  
## 1 0 273 0  
## 2 0 0 3  
##   
## Overall Statistics  
##   
## Accuracy : 1   
## 95% CI : (0.9899, 1)  
## No Information Rate : 0.7521   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 1   
##   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: 0 Class: 1 Class: 2  
## Sensitivity 1.0000 1.0000 1.000000  
## Specificity 1.0000 1.0000 1.000000  
## Pos Pred Value 1.0000 1.0000 1.000000  
## Neg Pred Value 1.0000 1.0000 1.000000  
## Prevalence 0.2397 0.7521 0.008264  
## Detection Rate 0.2397 0.7521 0.008264  
## Detection Prevalence 0.2397 0.7521 0.008264  
## Balanced Accuracy 1.0000 1.0000 1.000000

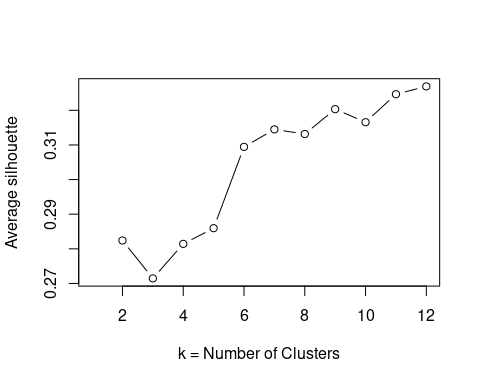
## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1 2  
## 0 87 0 3  
## 1 0 273 0  
## 2 0 0 0  
##   
## Overall Statistics  
##   
## Accuracy : 0.9917   
## 95% CI : (0.976, 0.9983)  
## No Information Rate : 0.7521   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.978   
##   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: 0 Class: 1 Class: 2  
## Sensitivity 1.0000 1.0000 0.000000  
## Specificity 0.9891 1.0000 1.000000  
## Pos Pred Value 0.9667 1.0000 NaN  
## Neg Pred Value 1.0000 1.0000 0.991736  
## Prevalence 0.2397 0.7521 0.008264  
## Detection Rate 0.2397 0.7521 0.000000  
## Detection Prevalence 0.2479 0.7521 0.000000  
## Balanced Accuracy 0.9946 1.0000 0.500000

1. Model validation Result

Off All models from task 5 it is observed that the tree classifier model is the best performing with accuracy of 100% for model generated by the ride4U dataset and an accuracy of 99.1% with the model generated by the noisy ride4U dataset. Both models classify all instances without any error, compared to instances classifier with accuracy of 81.8% for the model generated with the ride4U dataset and 85.4% for the model generated with the noisy ride4U dataset.

# make copy of the dataset  
ride4UC <- ride4U  
# select outcome variable  
label\_outcom <- ride4UC %>% select(label)  
# remove outcome variable from dataset  
ride4UC <- ride4UC %>% select(-label)  
  
# scale numeric variables  
#ride4UC[, c("day", "humidity", "uses", "temperature")] <- scale(ride4UC[, c("day", "humidity", "uses", "temperature")])  
  
# dummy code variables with two levels and are coded 1/0  
ride4UC$holiday <- ifelse(ride4UC$holiday == "yes", 1, 0)  
  
# dummy code variables that have three or more levels  
city <- as.data.frame(dummy.code(ride4UC$city))  
month <- as.data.frame(dummy.code(ride4UC$month))  
day\_of\_week <- as.data.frame(dummy.code(ride4UC$day\_of\_week))  
outlook <- as.data.frame(dummy.code(ride4UC$outlook))  
feel\_humidity <- as.data.frame(dummy.code(ride4UC$feel\_humidity))  
wind <- as.data.frame(dummy.code(ride4UC$wind))  
  
# combine new dummy variables with original dataset0  
ride4UC <- cbind(ride4UC, city, month, day\_of\_week, outlook, feel\_humidity, wind)  
  
#remove original values that had to be dummy coded  
ride4UC <- ride4UC %>% select(-one\_of(c("city", "month", "day\_of\_week", "outlook", "feel\_humidity", "wind")))  
  
# remove zero variance column from dataset  
ride4UC <- ride4UC[ , which(apply(ride4UC, 2, var) != 0)]  
  
  
# preprocessing  
prepride4U <- preProcess(ride4UC, method = c("center", "scale", "pca"), pcaComp = 4)  
ride4UC2 <- predict(prepride4U, newdata = ride4UC)

# find best k using the elbow method  
sil <- NULL  
  
for (i in 2:12) {  
 res <- kmeans(ride4UC2, centers = i, nstart = 25)  
 ss <- silhouette(res$cluster, dist(ride4UC2))  
 sil[i] <- mean(ss[, 3])  
}  
  
plot(1:12, sil, type="b", xlab="k = Number of Clusters", ylab = "Average silhouette")



km <- kmeans(ride4UC2, 3, nstart = 25, iter.max = 1000)  
  
#plotting the result  
plot3d(ride4UC2$PC1, ride4UC2$PC2, col=km$cluster)

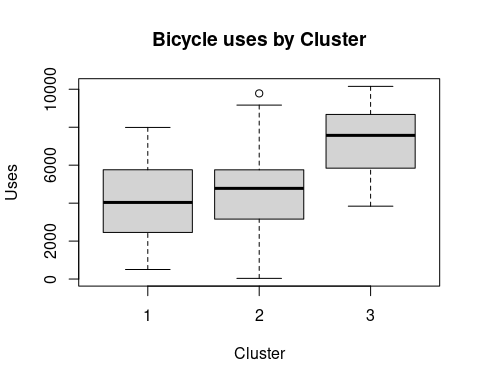
sort(table(km$cluster))

##   
## 1 3 2   
## 207 227 292

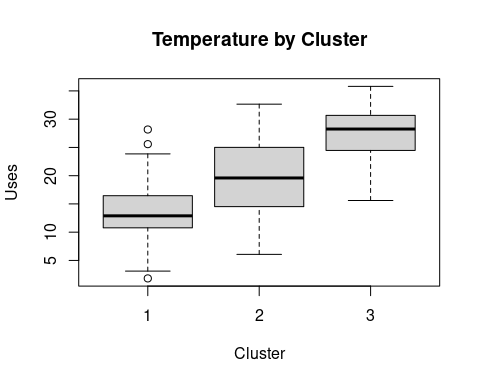
clust <- names(sort(table(km$cluster)))  
  
row.names(ride4UC[km$cluster == clust[1],])

row.names(ride4UC[km$cluster == clust[2],])

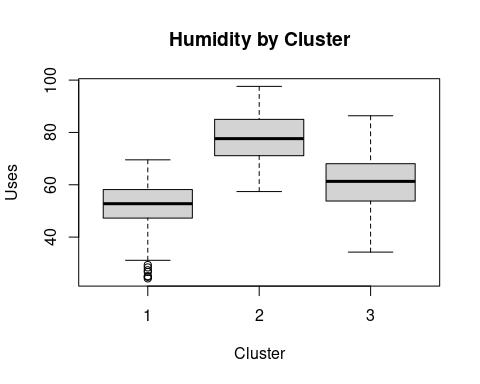
boxplot(ride4UC$uses ~ km$cluster, xlab="Cluster", ylab="Uses", main="Bicycle uses by Cluster")



boxplot(ride4UC$temperature ~ km$cluster, xlab="Cluster", ylab="Uses", main="Temperature by Cluster")



boxplot(ride4UC$humidity ~ km$cluster, xlab="Cluster", ylab="Uses", main="Humidity by Cluster")



ride4UC[km$cluster==clust[1], 1:20]

ride4UC[km$cluster==clust[2], 1:20]

ride4UC[km$cluster==clust[3], 1:22]

1. Clustering Algorithm discussion The silhouette method reveals 12 to be the best value of k for clustering. Given that the dataset has already been classified into three classes based on the level of complaints, it would be ideal to use three clusters. Kmeans is the preferred algorithm as it is easy to implement, and guarantees convergence, and generalizes to clusters of different shapes, while kmeans doesn’t do well with outliers, the dataset is scaled to reduce the impact any outlier would have on the algorithm. The clusters correlate with temperature, bicycle uses, and humidity.