Analysing the efficacy of machine learning with glass data

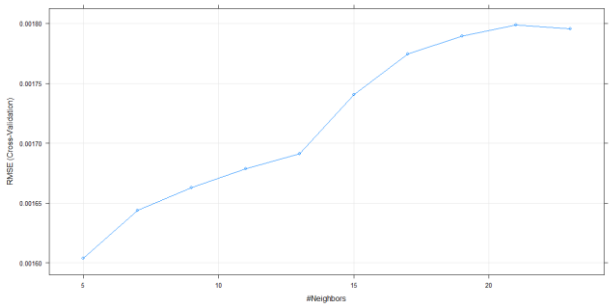
Introduction

Glass is a crucial material which has many applications such as building construction and windows. Due to the sheer versatility of the material, many kinds of glass are designed to fulfil different purposes. The purpose of this project is to use the data set compiled from the percentage metal composition of different glass to predict refractive index and to classify them into different types of glass. More specifically, the project aims to investigate the accuracy of machine learning in identifying glass as Type 1 or 2, and to predict the refractive index based on metal composition with reasonable accuracy. The techniques that will be used are kNN regression and linear regression for prediction, logistic regression and kNN classification for classification, and K-means clustering.

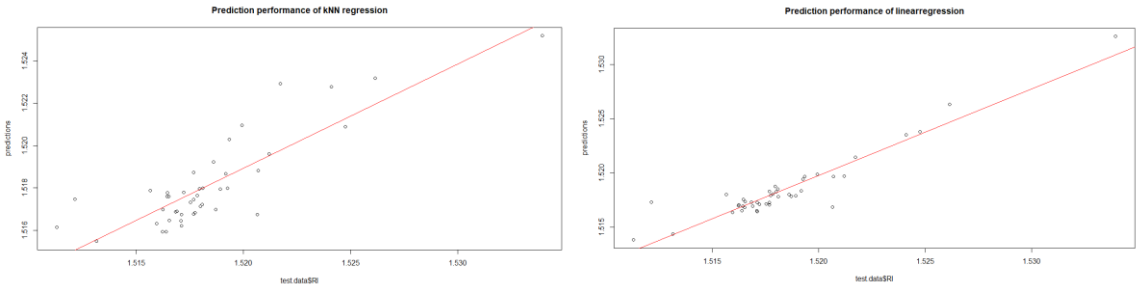
Analysis

Glass from mlbench package contains 214 samples of glass which are categorised into 6 types of glass based on the percentage metal composition (Na, Mg, Al, Si, K, Ca, Ba, Fe). The refractive index (RI) of the glass is also influenced by the metal composition. To prepare the data set for prediction, we first make another data frame Glass2 which does not have the Type variable and predict the RI based on the predictor variables (Na, Mg, Al, Si, K, Ca, Ba, Fe).

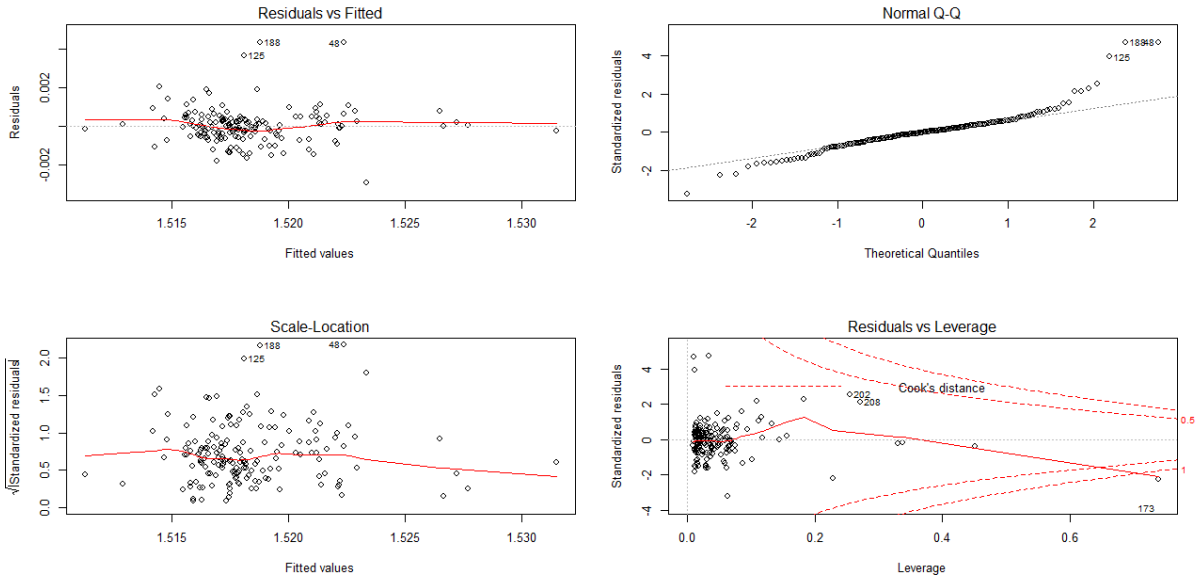
Prediction



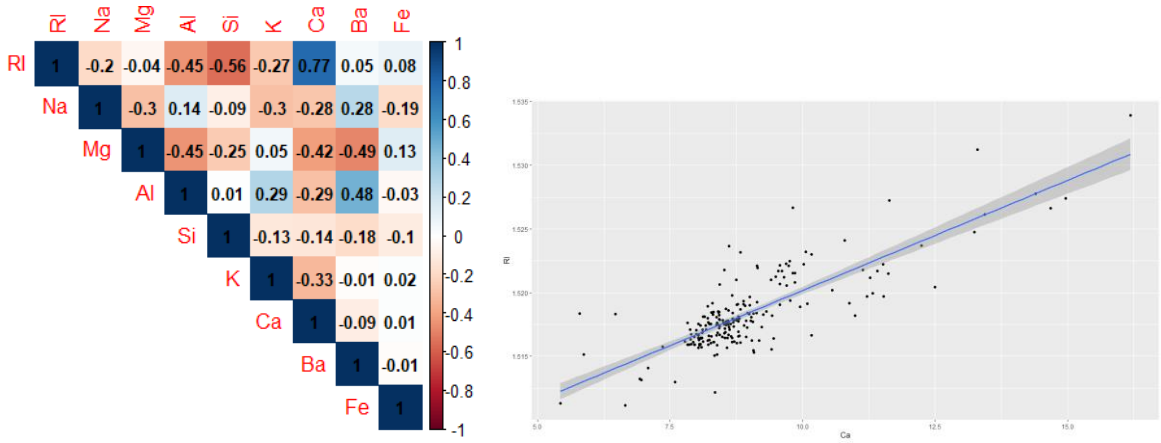
As above in the graph, the prediction error RMSE for kNN regression is minimised to 0.002189306 when k is set at k = 5.



For kNN regression, the graph shows that most of the data points are near the best fit linear line, so this is a decent prediction. However, linear regression has a far lower RMSE error of 0.001283655 and in the graph above the data points are far more compact to the best fit linear line. Thus, linear regression proves to be a better model for predicting the RI of glass based on metal composition.



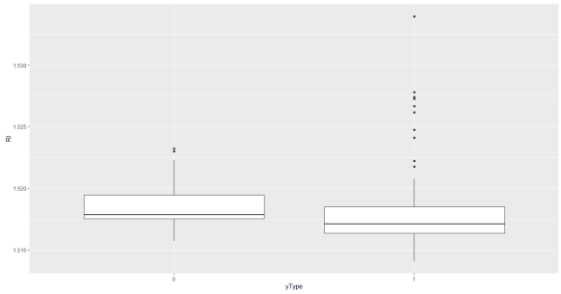
As above the outliers are samples 48,125,188,173,202,208. The plot points on the fitted values and theoretical quantities show a linear relation and can be further improved by removing the outliers.



The correlation coefficient plot above shows that Si and Ca are highly related to RI due to having coefficient value of |r| >= 0.5. As above, the graph of Ca to RI also shows that Ca is directly proportionate to RI.

Classification

We make use of the Glass1 data frame, which contains only Type 1 and 2 glass categorized into a new header yType with factor levels 0 and 1 respectively.



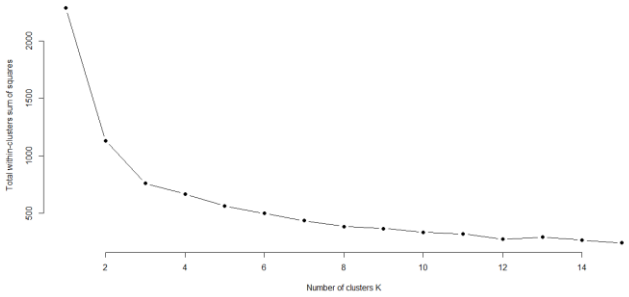
As above, the boxplot shows that Type 1 glass (shown under 0) has a higher mean RI but has a lower spread of RI. Type 2 glass (shown under 1 has lower RI but has very wide spread of RI.

Confusion matrix

knn1	0	1		y_pred (logistic	0	1
0	7	1		0	10	3
1	4	17		1	3	13

From the confusion matrix above, kNN classification has a superior accuracy of 82.8% compared to logistic regression which yields about 79.3%. Thus, kNN classification is a better model for categorising glass into Type 1 and 2. Accuracy is also highest when k=5.

Clustering



We attempt to identify clusters in Glass. The graph suggests that k=5 is the optimal number of clusters as it is the bend. The 5 clusters are listed below (ignore Type).

Cluster	RI	Na	Mg	Al	Si	K	Ca	Ba	Fe	Type
<int>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
1	1.52	13.1	3.50	1.38	72.8	0.584	8.35	0.0277	0.0642	NA
2	1.52	12.7	0.259	1.32	72.5	0.256	12.5	0.166	0.0695	NA
3	1.52	13.8	3.21	1.11	71.8	0.227	9.58	0.0767	0.0631	NA
4	1.52	14.5	0.154	2.08	73.4	0.197	8.65	0.916	0.0139	NA
5	1.51	13.3	0.893	3.19	70.4	4.7	6.59	0.733	0	NA

Conclusion

For this dataset, linear regression is a more accurate model than kNN regression for predicting the RI of glass samples. Interestingly, both methods also have very minute error RMSE values, which would imply that machine learning can predict RI of glass with very high levels of accuracy.

For classifying glass samples into Type 1 and Type 2, kNN classification model proved to be more reliable compared to logistic regression model. Both models have good accuracy in classifying into different types of glass, and thus machine learning can also reliably classify glass into Type 1 and 2. However, another alternative method would be needed if there is a need to classify into all 6 types of glass as the two models used only function well with two different classification variable levels.

Overall, machine learning is very accurate in predicting and classifying this dataset.

Appendix

Key code used:

```
library(mlbench)
library(dplyr)
library(caret)
library(ggplot2)
library(corrplot)
library(class)
data(Glass)
str(Glass)
summary(Glass)
#convert the Type variable into yType factor with only 2 levels and select only Type 1 and 2
Glass1 <- subset(Glass, Type==1 | Type ==2)
Glass1 <- Glass1%>%mutate(yType=factor(ifelse(Type=="2", 1,0)))%>%select(RI:Fe, yType)

########## Prediction ###############
##########1. kNN regression ###############
#form Glass2 for finding the correlation matrix between RI and other variables
Glass2 <- Glass%>%select(RI:Fe)
#split data into training and data sets
set.seed(100)
training.idx <- sample(1: nrow(Glass2), size=nrow(Glass2)*0.8)
train.data <-Glass2[training.idx, ]
test.data <- Glass2[-training.idx, ]
# Fit the model on the training set
set.seed(101)
model <- train(
  RI~., data = train.data, method = "knn",
  trControl = trainControl("cv", number = 10),
  preProcess = c("center","scale"),
  tuneLength = 10
)
# Plot model error RMSE vs different values of k
plot(model)
model$bestTune
predictions <-predict(model, test.data)
head(predictions)
#1.516316 1.518658 1.517638 1.517798 1.517446 1.516818
```

```
#Compute the prediction error RMSE
RMSE(predictions, test.data$RI)
#0.002189306
#visualize the performance of kNN reg,we plot predictedRI vs RI in the test data
par(mfrow=c(1,1))
plot(test.data$RI, predictions,main="Prediction performance of kNN regression")
#add a reference line
fit <- lm(predictions~test.data$RI,data = Glass2)
abline(fit, col="red")

##########2. Linear regression ###############
lmodel<-lm(RI~., data = train.data)
#print details of model fitting
summary(lmodel)
# Make predictions on the test data
predictions <-predict(lmodel, test.data)
RMSE(predictions, test.data$RI)
#0.001283655
plot(test.data$RI, predictions,main="Prediction performance of linearregression")
#add a reference line
fit <- lm(predictions~test.data$RI,data = Glass2)
abline(fit, col="red")
#calculate residuals
residuals(lmodel)
#create multiple plots on the same page
par(mfrow=c(2,2))
plot(lmodel)
#Visualize the correlation between the outcome RI and each predictor
par(mfrow=c(1,1))
corrplot(cor(train.data),type="upper",method="color",addCoef.col="black",number.cex=0.8)
#remove outliers from the training data
Glass1<-Glass1[-c(48,125,188,173,202,208),]
set.seed(100)
training.idx <-sample(1: nrow(Glass1), size=nrow(Glass1)*0.8)
train.data  <-Glass1[-training.idx,]
test.data <-Glass1[-training.idx, ]
p2model<-lm(RI~Na+Mg+Al+Si+K+Ca+Ba+Fe+I(Ca^2)+I(Si^2),data = train.data)
#print details of model fitting
summary(p2model)
```

```
# Make predictions on the test data
predictions <-predict(p2model, test.data)
RMSE(predictions, test.data$RI)
#0.0002637537
#create multiple residual plots on the same page
par(mfrow=c(2,2))
plot(p2model)

########## Data Visualisation with boxplot ###############
ggplot(Glass2, aes(x=Ca, y=RI))+geom_point()+geom_smooth(method='lm')
ggplot(Glass1, aes(x=yType, y=RI)) +geom_boxplot()

########## Classification ###############
##########1. Logistic regression ###############
set.seed(100)
training.idx <- sample(1: nrow(Glass1), size=nrow(Glass1)*0.8)
train.data <-Glass1[training.idx, ]
test.data <- Glass1[-training.idx, ]
mlogit <- glm(yType~RI+Na+Mg+Al+Si+K+Ca+Ba+Fe, data = train.data,family = "binomial")
summary(mlogit)
Pred.p <-predict(mlogit, newdata =test.data, type = "response")
y_pred_num <-ifelse(Pred.p > 0.5, 1, 0)
y_pred <-factor(y_pred_num, levels=c(0,1))
#Accuracy of the classification
mean(y_pred ==test.data$yType )
#0.7931034
#Create the confusion matrix with row-y_pred col=y
table(y_pred,test.data$yType)

##########2. kNN classification ###############
#Normalize numeric variables
nor <-function(x) { (x -min(x))/(max(x)-min(x)) }
Glass1[,1:9] <- sapply(Glass1[,1:9], nor)
#split data
set.seed(100)
training.idx <- sample(1: nrow(Glass1), size=nrow(Glass1)*0.8)
train.data <-Glass1[training.idx, ]
test.data <- Glass1[-training.idx, ]
#kNN classification
```

```
set.seed(101)
knn1<-knn(train.data[,1:9], test.data[,1:9], cl=train.data$yType, k=5)
mean(knn1 ==test.data$yType)
#0.8275862
table(knn1,test.data$y)
#try different k to find the best classfier
ac<-rep(0, 30)
for(i in 1:30){
  set.seed(101)
  knn.i<-knn(train.data[,1:9], test.data[,1:9], cl=train.data$yType, k=i)
  ac[i]<-mean(knn.i ==test.data$yType)
  cat("k=", i, " accuracy=", ac[i], "\n")
}
#Accuracy plot
par(mfrow=c(1,1))
plot(ac, type="b", xlab="K",ylab="Accuracy")
#k=5 best

########## Clustering ###############
##########1. K-means clustering ###############
k2 <- kmeans(Glass, centers = 2, nstart = 25)
str(k2)
k2
# function to compute total within-cluster sum of square
wcss <- function(k) {
  kmeans(Glass, k, nstart = 10 )$tot.withinss
}
# Compute and plot wss for k = 1 to k = 15
k.values <- 1:15
set.seed(100)
# apply wcss to all k values
wcss_k<-sapply(k.values, wcss)
plot(k.values, wcss_k, type="b", pch = 19, frame = FALSE,xlab="Number of clusters K",ylab="Total within-clusters sum of squares")
# 5
#final clustering with k=5
set.seed(100)
k5.final <- kmeans(Glass2, 5, nstart = 25)
k5.final
Glass %>% mutate(Cluster = k5.final$cluster) %>% group_by(Cluster) %>% summarise_all("mean")
```