Stats 503 Project: Predicting the Gestures

Chen Xie, Xun Wang, Xinye, Jiang

04/28/2019

1. Introduction

Artificial Intelligence is extremely popular in our new technology age. It is widely applied in every aspect of our daily life, for example, business, industry and healthcare. Combining artificial intelligence and machine learning techniques, our project is going to delve into an dataset which collects 64 recordings of 8 consecutive readings from 8 human arm muscle sensors to predict associated gestures for an artificial arm.

The dataset has about 11678 recordings of human arm muscle activity corresponding to four different hand gestures from a prosthetic control system. It has four classes of motions which are "rock", "scissors", "paper" and "ok" as the response variable. And the rest 64 predictor variables showing readings from sensors are the explanatory variables.

In addition, we explored the dataset numerically and graphically, done variable selection based on the importance of the variables, and utilized various classification methods including logistic regression, LDA, QDA, KNN, SVM, classification tree and random forest to fit the model and calculated the prediction accuracy respectively. In this process, we also discussed the connection between sensors, readings and hand gestures.

2. Data Exploration

In this dataset, "gestures", i.e. `V65`, is our response variable and `V1-V64` are all independent variables of 64 readings. Table 1 shows the specific data descriptions.

Table 1: Data Description

Variables names	Type	Description
V65	categorical	Response (rock:0, scissors:1, paper:2, ok:3)
V1-V8	continuous	Reading 1 Sensor 1-8
V9-V16	continuous	Reading 2 Sensor 1-8
V17-V24	continuous	Reading 3 Sensor 1-8
V25-V32	continuous	Reading 4 Sensor 1-8
V33-V40	continuous	Reading 5 Sensor 1-8
V41-V48	continuous	Reading 6 Sensor 1-8
V49-V56	continuous	Reading 7 Sensor 1-8
V57-V64	continuous	Reading 8 Sensor 1-8

To have an intuitive sense of the dataset, first look at the barchart of the response variable "gesture" in Figure 1. We saw that the data was balanced, as each class of the response variable all had around 2900 observations.

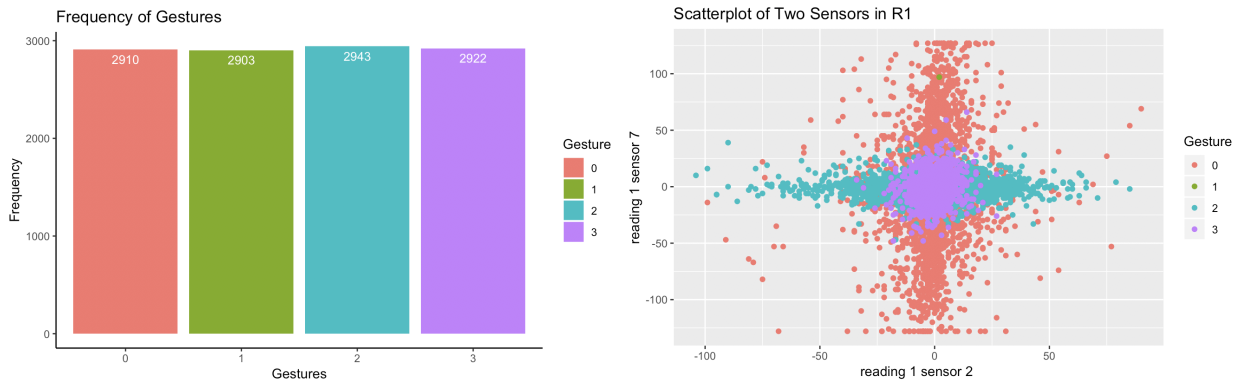

Figure 1: Barchart of Gestures and Scatterplot of Sensor 2 and Sensor 7 from Reading 1

Because our dataset had 64 predictor variables, which was quite high dimensional, we tried pairwise scatterplots of specific pairs to get some sense of the dataset. The scatterplot of two sensors in reading 1 is displayed in Figure 1. We could see that there is no obvious structure in the data. Combined with the numeric summaries of the predictor variables in Table 2 (note that we only show part of it), all the predictors were distributing around 0 and had long symmetric tails. As no specific pattern was detected, PCA won't be a satistfying way to reduce dimension. As our expectation, the first 35 principal components merely covered 80% of the variance, which meant that the effect of this dimension reduction method to this dataset was terrible.

Table 2: Numerical Summaries (only part is shown)

Summary	V1	V2	V3	V4	V5	V6	V7	V8
Min.	-116.00	-104.00	-33.00	-75.00	-121.00	-122.00	-128.00	-128.00
1st Qu.	-9.00	-4.00	-3.00	-4.00	-10.00	-15.00	-6.00	-8.00
Median	-1.00	-1.00	-1.00	-1.00	0.00	-1.00	-1.00	-1.00
Mean	-0.52	-0.73	-0.74	-0.73	-0.16	-0.55	-1.27	-0.66
3rd Qu.	7.00	3.00	2.00	3.00	10.00	13.00	4.00	6.00
Max.	111.00	90.00	34.00	55.00	92.00	127.00	127.00	126.00

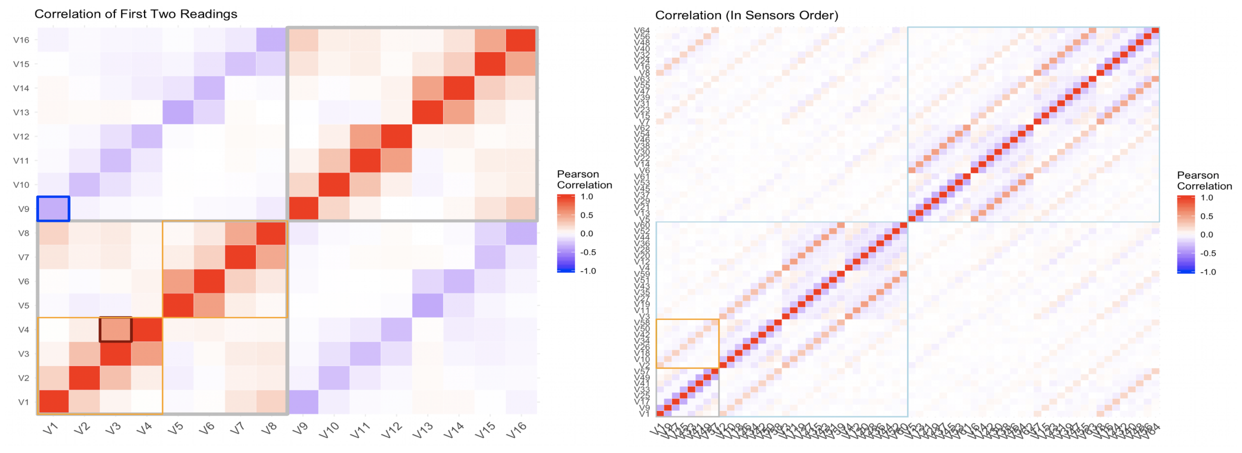

Figure 2: Correlation Plots

To figure out the correlations between the predictors, look at the correlation plots in Figure 2. From the left plot of the correlations of first two readings, we saw that the same sensors in the adjacent readings, for example, `V1` and `V9`, `V2` and `V10`, were negatively correlated. Pairwise adjacent sensors in each reading, like `V5` and `V6`, `V7` and `V8`, were positively correlated, while the first 4 and last 4 sensors in each reading were nearly uncorrelated. The right plot of the correlations in sensors order reconfirmed our conclusions before.

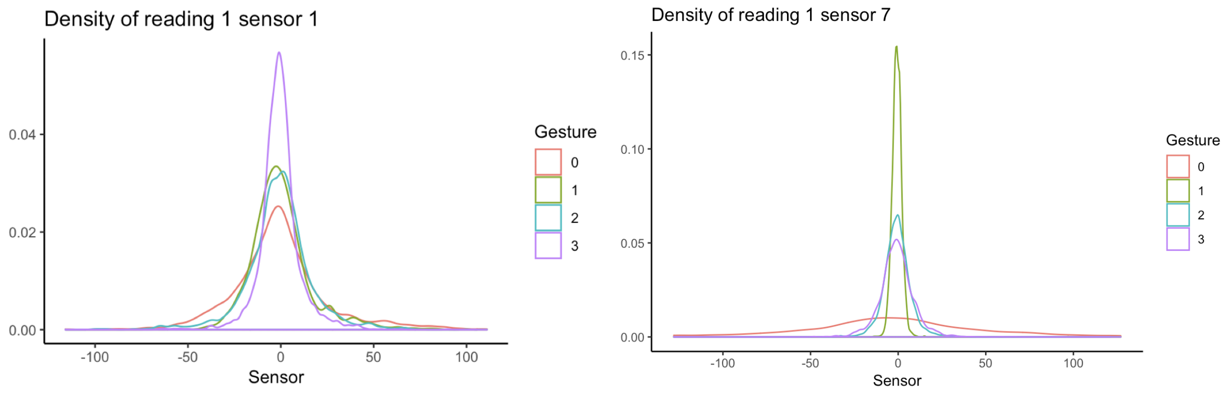

Figure 3: Density Plots

Using the density plots in Figure 3, we addressed the different significance of each predictor variable in prediction for gesture. For example, the densities of sensor 1 in reading 1 for 4 gesture classes were quite similar, while the densities of sensor 7 in reading 1 were quite different. In this case, the sensor 7 should be put on more weights for prediction than the sensor 1. The importance diversity encouraged us to do model selection to reduce dimensions.

To visualize the data more comprehensively, look at the contour plot of density of `V2` and `V7` in Figure 4. The data looked to have normal distribution with unequal variances. Based on this plot, we expected the decision boundary between classes to be non-linear and thus those classifiers generating linear boundaries such as LDA, logistic regression would perform poorly in this case.

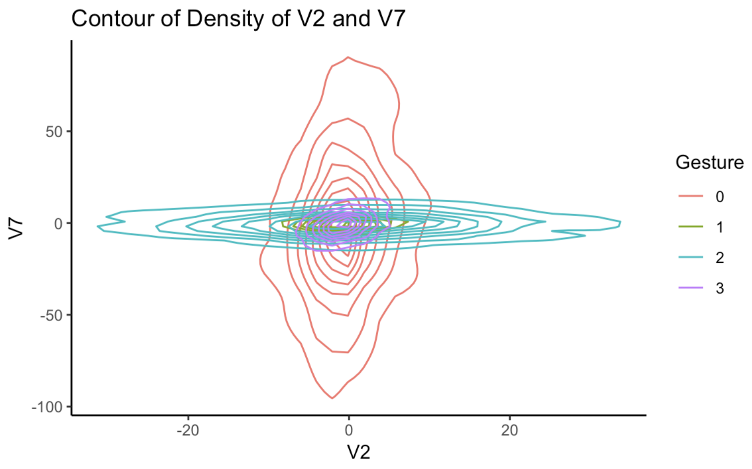

Figure 4: Contour of Density of V2 and V7

Variable Selection

As mentioned before, we had 64 predictor variables and some were more important than others. Therefore, we tried two methods to do variable selection to reduce redundance and better the prediction performance and interpretability.

The first method was to use random forest with mtry=8 to fit the whole dataset and check the importance of the variables. We found that the last but one sensor in each reading was most important and the second sensor in each reading was next most important, which can be seen in Figure 5.

The second method was to utilize stepwise forward selection based on AIC and BIC values to choose the corresponding optimal logistic regression models. The important variables selected by this method were basically the same as the ones we found using random forest.

Finally we separately checked the overall prediction accuracy using 8 predictors, i.e., the recordings of 7th sensors in 8 readings, and the accuracy using 16 predictors, i.e., the 7th and the 2nd sensors in 8 readings. As a result, we reached the conclusion that the latter one had overall better performance than the former one. Thus, we finally decided to carry on the remaining analysis based on 16 important predictors.

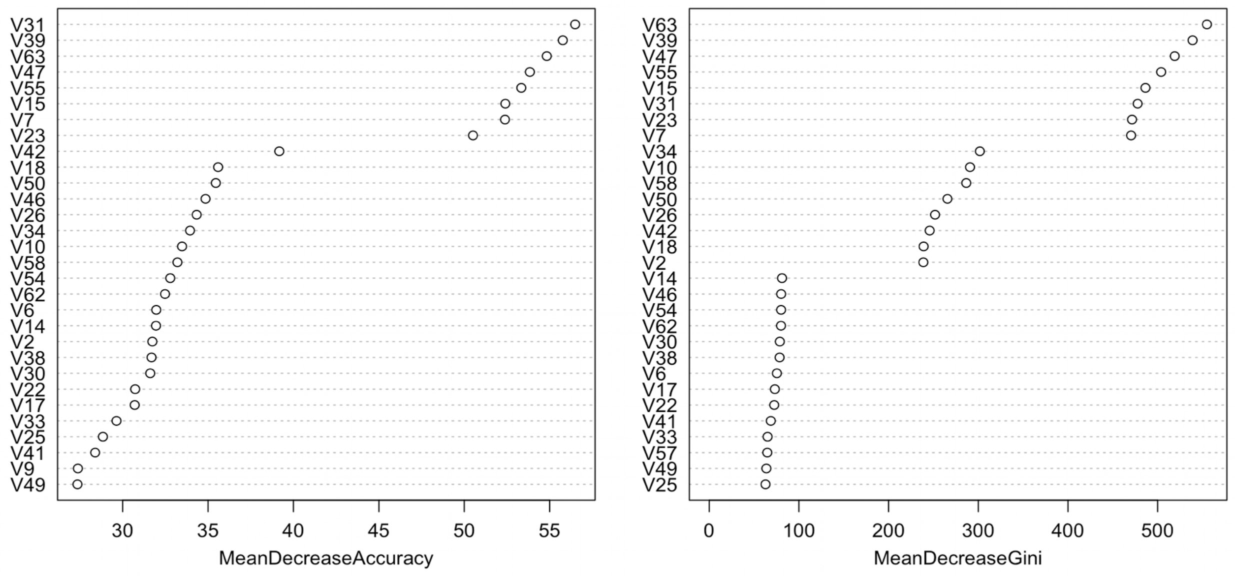

Figure 5: The Importance of the Variables

3. Classification Methods

In this part, we were about to fit various classification models using 16 significant variables. In order to compare their prediction performances, we first randomly split the dataset into training and testing parts at a ratio about 8:2.

3.1 Logistic Regression

The first model we tried was multinomial logistic regression. It is a generalization of original logistic regression when the response has multiple classes. We fitted the a multinomial logistic regression with all terms linear and also fitted another logistic model which included quadratic terms.

Table 3: Logistic Regression (Linear & Quadratic) Prediction

Truth vs Logistic Prediction	Linear 0	1	2	3	Quadratic 0	1	2	3
0	317	60	90	135	560	1	7	34
1	18	22	200	329	0	556	4	9
2	84	158	205	153	13	38	523	26
3	119	33	152	261	15	103	12	435

From the confusion matrix for both models in Table 3, we saw that adding quadratic terms in the logistic regression model greatly increased the prediction accuracy, from 34.5% to 88.8%. This confirmed our expectation in data exploration that the decision boundary between classes was not linear.

3.2 LDA & QDA

LDA and QDA are also powerful approaches to identify the type of decision boundary, i.e., linear or non-linear. In this step, we fitted LDA and QDA classifiers on this dataset to verify our conclusion in logistic regression.

Table 4: LDA & QDA Prediction

Truth vs	LDA 0	1	2	3	QDA 0	1	2	3
0	294	63	108	137	559	1	4	38
1	4	17	204	344	0	544	8	17
2	54	164	222	160	9	20	559	12
3	84	28	172	281	21	72	10	462

Here we could see that the classifier QDA with quadratic classification boundary had an overwhelming advantage over the classifier LDA with linear boundary regarding the prediction accuracy. QDA had a very good prediction performance with prediction accuracy over 90.9%. In contrast, LDA performed quite poorly with an accuracy reaching only 34.8%, proving that a linear classification boundary could be excluded from the solutions for this dataset.

3.3 KNN

As the decision boundary was non-linear, we wanted to try more flexible models with less assumptions on distribution of the dataset. Therefore, we tried KNN method on this dataset. First, we performed 5-fold cross validation on training set to choose the corresponding K with the smallest cross-validation error. From Figure 6, we could see that K equals 3 was the optimal choie. After that, we fitted KNN model with K=3 again and obtained the confusion matrix as Table 5.

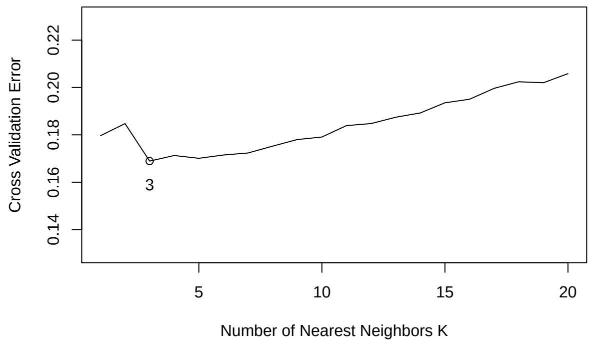

Figure 6: The 5-fold Cross Validation Errors for Each Choice of K.

Table 5 indicated that KNN (K=3) also performed well on this dataset, although its performance was a little worse than the logistic regression with quadratic terms and QDA.

Table 5: KNN (K=3) Prediction

Truth vs KNN	0	1	2	3
0	517	1	12	72
1	0	553	3	13
2	2	112	435	51
3	7	78	8	472

3.4 SVM

Fit the SVM classifiers with radial kernel and polynomial kernel to the dataset. The optimal parameters such as `cost`, `gamma`, `degree` were chosen by minimizing the validation errors. Note that here we used validation instead of cross validation under the consideration of computational cost. For the SVM with radial kernel, the optimal parameters chosen were cost = 1, gamma = 0.5 . For the SVM with polynomial kernel, the optimal cost is 100 and degree is 2.

Table 6: SVM with Radial & Polynomial Kernel Prediction

Truth vs SVM	Radial 0	1	2	3	Polynomial 0	1	2	3
0	571	0	3	28	552	1	15	34
1	0	549	12	8	0	548	9	12
2	208	20	364	8	7	38	537	18
3	25	56	20	464	16	80	11	458

Table 6 showed that SVM with these two kernels both performed well, with the one with polynomial kernel having higher accuracy rate than the one with the radial kernel. We saw that the SVM with polynomial kernel with optimal degree 2 had relatively better prediction performance, indicating that the decision boundary between classes probably did have quadratic curves.

3.5 Classification Tree & Random Forest

Finally we fitted classification tree and random forest to the dataset.

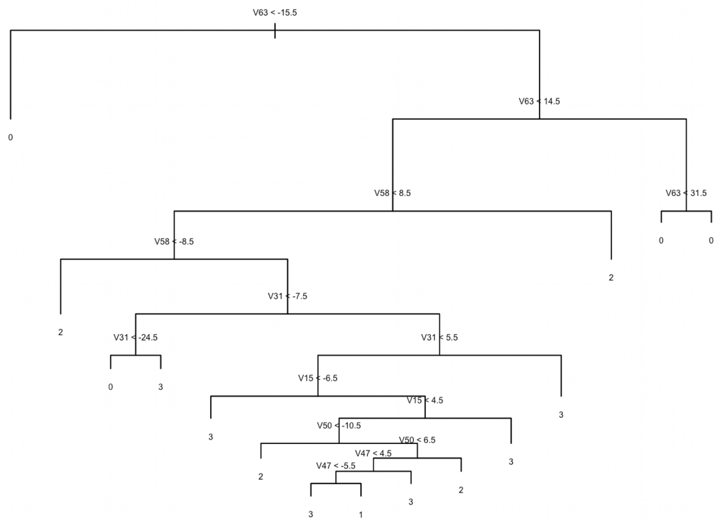

Figure 7: Decision Tree Plot

The ultimate decision tree model was shown in Figure 7. We saw that it used the variables `V63`, `V58`, `V31`, `V15`, `V50` and `V47`. The decision tree had accuracy slightly larger than 70%, which was hardly regarded as good performance. This intrigued us to do random forest to better the prediction performance.

The random forest had the best prediction accuracy so far which was over 91.1%. Becuase partial plots of 8 readings from each sensor displayed a similar pattern, we just showed the partial plots of `V7` and `V18` in Figure 8 as the representatives of 7th and 2nd sensor. Keeping other predictors unchangeable, the influence of the 7th and 2nd sensors to the gesture classes varied from time to time. It was not linear nor constant, so these senors produced a significant and non-linear influence to response. In general, the effects of sensors on response exhibited different patterns when they were closed to 0 and away from 0.

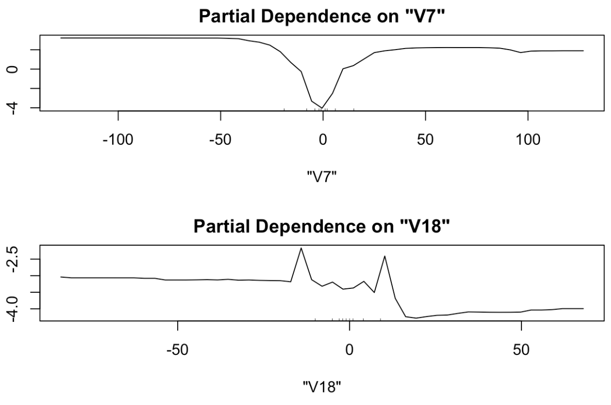

Figure 8: Random Forest Partial Plots

Table 7 showed the prediction results of the classification tree and random forest. As we could see, random forest outperformed the classification tree model for each class category.

Table 7: Decision Tree and Random Forest Prediction

Truth vs	Tree 0	1	2	3	Random Forest 0	1	2	3
0	463	1	49	89	569	0	9	24
1	2	441	52	74	0	503	18	48
2	37	35	365	163	2	15	569	14
3	62	92	23	388	33	24	20	488

3.6 Prediction Accuracy Comparison

Table 8: Prediction Accuracy of Various Classification Methods

Method	Prediction Accuracy %
Logistic Regression	34.46
Logistic Regression with quadratic terms	88.78
LDA	34.85
QDA	90.92
KNN	84.63
SVM (radial)	83.39
SVM (polynomial)	89.68
Classification Tree	70.93
Random Forest	91.14

Table 8 collected the prediction accuracy rates of all the classification methods we went through. We can see that random forest had the best prediction performance, QDA, SVM with degree-2 polynomial kernel and logistic regression with quadratic terms of the predictors had the next best performance. In comparsion, LDA and linear multinomial logistic regression performed quite badly.

4. Conclusion and Discussion

In conclusion, this specific dataset was balanced and clearly had non-linear, probably quadratic decision boundary between gesture classes according to the prediction accuracy of different classification methods.

In order to reduce dimensionality, we used the random forest and stepwise forward selection method to do variable selection. We found that the motion of the muscles corresponding to the 2nd and 7th sensors are almost decisive for these 4 gestures. Then we mainly used these 16 predictors to make preditions on gestures classes.

Among all the classification methods we tried, random forest performed the best, other complicated classifiers with quadratic decision boundary such as QDA, SVM with degree-2 polynomial kernel and logistic regression with quadratic terms all performed quite well. We finally reached a prediction accuracy around 90%, which is very satisfying for this dataset.

5. Reference

Yashuk, K. (2019). Classify gestures by reading muscle activity. https://www.kaggle.com/kyr7plus/emg-4#0.csv

Molnar, C. (2019). Interpretable Machine Learning. https://christophm.github.io/interpretable-ml-book/pdp.html

Appendix

```
## Table 1
t1=data.frame(name=c("V65","V1-V8","V9-V16","V17-V24","V25-V32","V33-V40","V41-V48","V49-V56","V57-V64"
              type=c("categorical",rep("continuous",8)),des=c("Response (rock:0, scissors:1, paper:2, o
cap="Data Description"
knitr::kable(t1,format='pandoc',caption=cap,align='l',
             col.names=c('Variables names','Type','Description'))
## Read the Data
dat0 = read.csv("0.csv", header = F)
dat1 = read.csv("1.csv", header = F)
dat2 = read.csv("2.csv", header = F)
dat3 = read.csv("3.csv", header = F)
g = rbind(dat0, dat1, dat2, dat3)
g$V65=as.factor(g$V65)
## Libraries
library(ggplot2);library(GGally);library(gbm);library(reshape2);library(gplots)
library(dplyr);library(e1071);library(tree);library(class);library(MASS);
library(nnet);library(randomForest);library(foreign);
# Response barchart & Scatterplot
knitr::include_graphics(c("response.png","scatterplot.png"))
## Table 2: Summary
summary_col = function(x)
  {c(unname(quantile(x,c(0,0.25,0.5))),mean(x),unname(quantile(x,c(0.75,1))))}
t2=data.frame(Summary=c("Min.","1st Qu.","Median","Mean","3rd Qu.","Max."),
              V1=summary_col(g[,1]),V2=summary_col(g[,2]),
              V3=summary_col(g[,3]),V4=summary_col(g[,4]),
              V5=summary_col(g[,5]),V6=summary_col(g[,6]),
              V7=summary_col(g[,7]),V8=summary_col(g[,8]))
cap="Numerical Summaries (only part is shown)"
knitr::kable(t2,format='pandoc',caption=cap,align='l',digits = 2)
# Correlation plots
knitr::include_graphics(c("correlation1.png","correlation2.png"))
#Histograms of reading 1 of sensor 1 and sensor 7
knitr::include_graphics(c("r1sensor1.png","r1sensor7.png"))
# Contour of density
knitr::include_graphics(c("contour.png"))
# The Importance of the Variables
## Perform Variable Selection
## by random forest and check the importance of variables
#set.seed(77)
#rf=randomForest(V65~.,data=g,mtry=8,importance=TRUE)
#varImpPlot(rf)
knitr::include_graphics(c("VarImport.png"))
## Variable Selection by stepwise logistic regression based on AIC/BIC
#min.model = multinom(V65~1, data=g[train,])
#biggest = multinom(V65~., data=g[train,])
#fwd1.model = step(min.model, scope=formula(biggest), direction="forward")
#summary(fwd1.model)
#pred_lr1 = predict(fwd1.model, g[-train,])
#mean(pred_lr1 == g[-train,65])
#fwd2.model = step(min.model, scope=formula(biggest), direction="forward", k=log(length(train)))
#summary(fwd2.model)
## Split the dataset into training and test sets
```

```r
set.seed(77)
g_new=g[,c(seq(from=2,to=58,by=8),seq(from=7,to=63,by=8),65)]
train=sample(1:nrow(g_new),size=nrow(g_new)*0.8,replace=FALSE)
## Multinomial Logistic Regression
set.seed(3)
lr_fit1 = multinom(V65~., data=g_new[train,], trace=FALSE)
lr_fit2 = multinom(V65~poly(V2,2)+poly(V10,2)+poly(V18,2)+poly(V26,2)+poly(V34,2)+poly(V42,2)+poly(V50,
lr_pred1 = predict(lr_fit1, g_new[-train,])
accuracy_lr1 = mean(lr_pred1 == g_new[-train,'V65'])
lr_pred2 = predict(lr_fit2, g_new[-train,])
accuracy_lr2 = mean(lr_pred2 == g_new[-train,'V65'])
# Table 3: Logistic Regression Prediction
lr_pred = unname(cbind(table(g_new[-train,'V65'], lr_pred1), table(g_new[-train,'V65'], lr_pred2)))
colnames(lr_pred) = c("Linear 0","1","2","3","Quadratic 0","1","2","3")
knitr::kable(data.frame("Truth"=0:3, lr_pred), align = "c",
             col.names = c("Truth vs Logistic Prediction", colnames(lr_pred)),
             caption = "Logistic Regression (Linear & Quadratic) Prediction")
## LDA
lda_fit = lda(V65~., data=g_new[train,])
lda_pred = predict(lda_fit, g_new[-train,])$class
accuracy_lda = mean(lda_pred == g_new[-train,'V65'])
## QDA
qda_fit = qda(V65~., data = g_new[train,])
qda_pred = predict(qda_fit, g_new[-train,])$class
accuracy_qda = mean(qda_pred==g_new[-train,'V65'])
# Table 4: LDA, QDA
lqda_pred = unname(cbind(table(g_new[-train,'V65'], lda_pred),
                    table(g_new[-train,'V65'], qda_pred)))
colnames(lqda_pred) = c("LDA 0","1","2","3","QDA 0","1","2","3")
knitr::kable(data.frame("Truth"=0:3, lqda_pred),
             col.names = c("Truth vs", colnames(lqda_pred)), align = "c",
             caption = "LDA & QDA Prediction")
## Initialize
knn_cv_error = NULL
## Possible k's
k_list = 1:20
set.seed(3)
## Cross Validation function for knn
knn.cv = function(k, t, nfolds=5) {
  n_train = nrow(t)
  ## Split training and validation sets
  s = split(sample(n_train), rep(1:nfolds, length=n_train))
  cv_error = 0
  for(i in seq(nfolds)){
    ## Computing validation errors
    knn_cv_pred = knn(t[-s[[i]],-17], t[s[[i]],-17], t[-s[[i]],17], k=k)
    cv_error = cv_error + mean(knn_cv_pred!=t[s[[i]],17])
  }
  cv_error = cv_error / nfolds
}
## Perform cross validation
## choose k=3
for(k in k_list) {
  knn_cv_error = c(knn_cv_error, knn.cv(k, g_new[train,]))
}
## Plot validation errors
cap = 'The 5-fold Cross Validation Errors for Each Choice of K.'
plot(k_list, knn_cv_error, type='l', ylim=c(0.13, 0.23),
     xlab='Number of Nearest Neighbors K', ylab='Cross Validation Error')
points(k_list[which.min(knn_cv_error)], min(knn_cv_error))
text(k_list[which.min(knn_cv_error)], min(knn_cv_error)-0.01, "3")
## Predicting by knn and k=3
set.seed(3)
knn_pred = knn(g_new[train,-17], g_new[-train,-17], g_new[train,17], k=3)
accuracy_knn = mean(knn_pred == g_new[-train,17])
# Table 5: KNN
k_pred = unname(cbind(table(g_new[-train,'V65'], knn_pred), 1))[,-5]
colnames(k_pred) = c("0","1","2","3")
knitr::kable(data.frame("Truth"=0:3, k_pred),
             col.names = c("Truth vs KNN", colnames(k_pred)), align = "c",
             caption = "KNN (K=3) Prediction")
## SVM radial kernel
#set.seed(77)
## The validation set
#valid=sample(train,size=floor(length(train)*0.2),replace=FALSE)
#train2=setdiff(train,valid)
## Possible parameters
#costv=c(0.01,0.1,1,10,100)
#gammav=c(0.5,1,2,3,4)
#svm_v_error=cbind(expand.grid(costv,gammav),0)
## Perform validation set method to choose parameters
#for (i in 1:nrow(svm_v_error)){
#  svm_v_fit=svm(V65~.,data=g_new[train2,], kernel="radial", cost=svm_v_error[i,1],gamma=svm_v_error[i,
#  svm_v_pred=predict(svm_v_fit,g_new[valid,])
#  svm_v_error[i,3]=mean(svm_v_pred!=g_new[valid,"V65"])}
#svm_v_error[which.min(svm_v_error[,3]),]
## Fit svm with cost=1, gamma=0.5
svm_fit1=svm(V65~.,data=g_new[train,], kernel="radial", cost=1, gamma=0.5)
svm_pred1=predict(svm_fit1,g_new[-train,])
accuracy_svm1=mean(svm_pred1==g_new[-train,"V65"])
## SVM polynomial kernel
#set.seed(77)
## The validation set
#valid=sample(train,size=floor(length(train)*0.2),replace=FALSE)
#train2=setdiff(train,valid)
## Possible parameters
#costv=c(0.01,0.1,1,10,100,1000)
#degreev=c(2,3)
#svm_v_error=cbind(expand.grid(costv,degreev),0)
## Perform validation set method to choose parameters
#for (i in 1:nrow(svm_v_error)){
#  svm_v_fit=svm(V65~.,data=g_new[train2,], kernel="polynomial", cost=svm_v_error[i,1],degree=svm_v_err
#  svm_v_pred=predict(svm_v_fit,g_new[valid,])
#  svm_v_error[i,3]=mean(svm_v_pred!=g_new[valid,"V65"])}
#svm_v_error[which.min(svm_v_error[,3]),]
## Fit svm with cost=100, degree=2
```

```r
svm_fit2=svm(V65~.,data=g_new[train,], kernel="polynomial", cost=100, degree=2)
svm_pred2=predict(svm_fit2,g_new[-train,])
accuracy_svm2=mean(svm_pred2==g_new[-train,"V65"])
# Table 6: SVM
svm_pred = unname(cbind(table(g_new[-train,'V65'], svm_pred1),
                      table(g_new[-train,'V65'], svm_pred2)))
colnames(svm_pred) = c("Radial 0","1","2","3","Polynomial 0","1","2","3")
knitr::kable(data.frame("Truth"=0:3, svm_pred),
             col.names = c("Truth vs SVM", colnames(svm_pred)), align = "c",
             caption = "SVM with Radial & Polynomial Kernel Prediction")
# Decision Tree Plot and Random Forest Partial Plots
knitr::include_graphics(c("tree.png"))
## Classification Tree
tree_fit=tree(V65~.,data=g_new[train,])
tree_pred=predict(tree_fit,g_new[-train,],type="class")
accuracy_tree=mean(tree_pred==g_new$V65[-train])
# Random Forest
set.seed(77)
rf_fit=randomForest(V65~.,data=g_new[train,],mtry=4,importance=TRUE)
rf_pred = predict(rf_fit,newdata=g_new[-train,])
accuracy_rf=mean(rf_pred==g_new$V65[-train])
# Decision Tree Plot and Random Forest Partial Plots
knitr::include_graphics(c("randomforest.png"))
# Table 7: tree & random forest
t_pred = unname(cbind(table(g_new[-train,'V65'], tree_pred),
                      table(g_new[-train,'V65'], rf_pred)))
colnames(t_pred) = c("Tree 0","1","2","3","Random Forest 0","1","2","3")
knitr::kable(data.frame("Truth"=0:3, t_pred),
             col.names = c("Truth vs", colnames(t_pred)), align = "c",
             caption = "Decision Tree and Random Forest Prediction")
t2=data.frame(method=c("Logistic Regression","Logistic Regression with quadratic terms","LDA","QDA","KN
              pred=100*c(accuracy_lr1,accuracy_lr2,accuracy_lda,accuracy_qda,accuracy_knn,accuracy_svm1
cap="Prediction Accuracy of Various Classification Methods"
knitr::kable(t2,format='pandoc',caption=cap,align='c',digits = 2,
             col.names=c('Method','Prediction Accuracy %'))
```