Homework 7

Karan Ashar

4/7/2021

## Introduction

We have been given a dataset where each row represents a digit (0-9). The columns represent each pixel and the value in the column represent the intensity of each pixel. Our goal is to build a model which could predict the digit based on the pixel values. This is a classification problem. To be specific it is a multiClass classification problem. We will build three Machine Learning models – KNN, Support Vector Machine & its variants, and Random Forest. We will then compare the results to see which performs better. We will also compare them to last week’s homework as well.

## Loading Required Packages

set.seed(608)  
setwd("C:/Users/karan/Desktop/IST707/data")  
library(e1071)

library(e1071)

library(dplyr)

library(rpart.plot)

library(rpart)

library(splitTools)

library(ranger)

library(caret)

library(MLmetrics)

library(kernlab)

library(e1071)

library(stringr)

library(randomForest)

library(rsample)

library(plyr)

## Preprocessing

We load both the Training Set and as well as the Test set.  
After loading the data, I had to carry out one pre-processing step. I had to convert our output label to the words they correspond to. For example, “1” would convert to “One”. We had to do this because of package dependencies

data=read.csv("digit-train.csv")  
data$label <- mapvalues(data$label, from=c("1","2","3","4","5","6","7","8","9","0"), to=c("One","Two","Three","Four", "Five","Six","Seven","Eight","Nine","Zero"))  
  
  
test=read.csv("digit-test.csv")  
test$label <- mapvalues(test$label, from=c("1","2","3","4","5","6","7","8","9","0"), to=c("One","Two","Three","Four", "Five","Six","Seven","Eight","Nine","Zero"))

**Note – [ We will use K fold Cross validation to train our models. The K we use for all our models is ‘3’]**

## KNN

The first model we will run is K-Nearest Neighbors. We will try a few values of the hyperparameter – ‘k’.   
While training we receive an accuracy of 0.925. It takes 6.678 mins to run.  
While prediction we get an accuracy of 0.9366.

grid\_search=expand.grid(k=c(3,5,7,9,11,13,15))  
  
train\_control <- trainControl(method = "cv",   
 number = 3,  
 classProbs=TRUE,  
 summaryFunction=multiClassSummary  
 )  
  
start\_time <- Sys.time()  
knn <- train(label~.,data=data,   
 method = "knn",   
 trControl = train\_control,  
 tuneGrid = grid\_search  
 )  
  
end\_time <- Sys.time()  
end\_time - start\_time

## Time difference of 6.678906 mins

knn$results %>% top\_n(3, wt = Accuracy) %>%arrange(desc(Accuracy))

## k logLoss AUC prAUC Accuracy Kappa Mean\_F1  
## 1 3 1.0768011 0.9833919 0.1747730 0.9252033 0.9168512 0.9242914  
## 2 5 0.8253198 0.9877232 0.2762900 0.9233006 0.9147321 0.9226780  
## 3 7 0.6869132 0.9900041 0.3497544 0.9159164 0.9065227 0.9153928  
## Mean\_Sensitivity Mean\_Specificity Mean\_Pos\_Pred\_Value Mean\_Neg\_Pred\_Value  
## 1 0.9242256 0.9916875 0.9260892 0.9917274  
## 2 0.9223741 0.9914707 0.9257674 0.9915261  
## 3 0.9150160 0.9906491 0.9191007 0.9907111  
## Mean\_Precision Mean\_Recall Mean\_Detection\_Rate Mean\_Balanced\_Accuracy  
## 1 0.9260892 0.9242256 0.09252033 0.9579566  
## 2 0.9257674 0.9223741 0.09233006 0.9569224  
## 3 0.9191007 0.9150160 0.09159164 0.9528325  
## logLossSD AUCSD prAUCSD AccuracySD KappaSD Mean\_F1SD  
## 1 0.2699451 0.004394821 0.0052197091 0.010332133 0.011477500 0.010900009  
## 2 0.1669994 0.002766249 0.0009091183 0.007128317 0.007916015 0.007697179  
## 3 0.1469554 0.002432915 0.0064732484 0.010409054 0.011562753 0.011403055  
## Mean\_SensitivitySD Mean\_SpecificitySD Mean\_Pos\_Pred\_ValueSD  
## 1 0.010388762 0.0011366791 0.011259918  
## 2 0.007222741 0.0007798829 0.007913358  
## 3 0.010822563 0.0011410282 0.011742819  
## Mean\_Neg\_Pred\_ValueSD Mean\_PrecisionSD Mean\_RecallSD Mean\_Detection\_RateSD  
## 1 0.0011440067 0.011259918 0.010388762 0.0010332133  
## 2 0.0007867515 0.007913358 0.007222741 0.0007128317  
## 3 0.0011472431 0.011742819 0.010822563 0.0010409054  
## Mean\_Balanced\_AccuracySD  
## 1 0.005762149  
## 2 0.004000853  
## 3 0.005981775

Prediction:-

start\_time <- Sys.time()  
prediction=predict(knn,test[,-1])  
end\_time <- Sys.time()  
end\_time - start\_time

## Time difference of 40.34712 secs

confusionMatrix(table(test$label,prediction))

## Confusion Matrix and Statistics  
##   
## prediction  
## Eight Five Four Nine One Seven Six Three Two Zero  
## Eight 339 20 2 9 9 1 3 7 1 2  
## Five 1 364 0 4 2 0 7 9 0 1  
## Four 0 0 363 32 6 2 1 0 0 0  
## Nine 1 1 5 354 3 12 2 7 0 1  
## One 0 0 0 0 473 1 1 0 3 0  
## Seven 0 0 5 12 10 437 0 0 1 0  
## Six 2 4 0 0 0 0 392 0 0 6  
## Three 4 9 0 6 7 7 1 409 1 2  
## Two 2 1 0 0 10 7 1 3 392 4  
## Zero 0 0 0 0 0 0 4 0 1 409  
##   
## Overall Statistics  
##   
## Accuracy : 0.9366   
## 95% CI : (0.9288, 0.9438)  
## No Information Rate : 0.1239   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.9295   
##   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: Eight Class: Five Class: Four Class: Nine  
## Sensitivity 0.97135 0.91228 0.96800 0.84892  
## Specificity 0.98597 0.99368 0.98928 0.99154  
## Pos Pred Value 0.86260 0.93814 0.89851 0.91710  
## Neg Pred Value 0.99737 0.99081 0.99684 0.98347  
## Prevalence 0.08313 0.09505 0.08933 0.09933  
## Detection Rate 0.08075 0.08671 0.08647 0.08433  
## Detection Prevalence 0.09362 0.09242 0.09624 0.09195  
## Balanced Accuracy 0.97866 0.95298 0.97864 0.92023  
## Class: One Class: Seven Class: Six Class: Three Class: Two  
## Sensitivity 0.9096 0.9358 0.95146 0.94023 0.98246  
## Specificity 0.9986 0.9925 0.99683 0.99017 0.99263  
## Pos Pred Value 0.9895 0.9398 0.97030 0.91704 0.93333  
## Neg Pred Value 0.9874 0.9920 0.99473 0.99307 0.99815  
## Prevalence 0.1239 0.1112 0.09814 0.10362 0.09505  
## Detection Rate 0.1127 0.1041 0.09338 0.09743 0.09338  
## Detection Prevalence 0.1139 0.1108 0.09624 0.10624 0.10005  
## Balanced Accuracy 0.9541 0.9641 0.97414 0.96520 0.98754  
## Class: Zero  
## Sensitivity 0.96235  
## Specificity 0.99867  
## Pos Pred Value 0.98792  
## Neg Pred Value 0.99577  
## Prevalence 0.10124  
## Detection Rate 0.09743  
## Detection Prevalence 0.09862  
## Balanced Accuracy 0.98051

## SVM – Linear

The second model we will try is Support Vector Machine – Linear. This version does not use the kernel trick.   
The model has one hyperparameter – ‘C’. We will try a few values of ‘C’.   
While training we get an accuracy of

modelLookup('svmLinear')

## model parameter label forReg forClass probModel  
## 1 svmLinear C Cost TRUE TRUE TRUE

grid\_search=expand.grid(C=c(seq(0.01,1,length=5)))  
  
  
train\_control <- trainControl(method = "cv",   
 number = 3,  
 summaryFunction=multiClassSummary  
)  
  
start\_time <- Sys.time()  
svm\_l <- train(label~.,data=data,   
 method = "svmLinear",   
 trControl = train\_control,  
 tuneGrid = grid\_search,  
)

end\_time <- Sys.time()  
end\_time - start\_time

## Time difference of 59.69698 secs

svm\_l$results %>% top\_n(3, wt = Accuracy) %>%arrange(desc(Accuracy))

## C Accuracy Kappa Mean\_F1 Mean\_Sensitivity Mean\_Specificity  
## 1 0.0100 0.9044775 0.8938218 0.9033975 0.9035477 0.9893957  
## 2 0.2575 0.9044775 0.8938218 0.9033975 0.9035477 0.9893957  
## 3 0.5050 0.9044775 0.8938218 0.9033975 0.9035477 0.9893957  
## 4 0.7525 0.9044775 0.8938218 0.9033975 0.9035477 0.9893957  
## 5 1.0000 0.9044775 0.8938218 0.9033975 0.9035477 0.9893957  
## Mean\_Pos\_Pred\_Value Mean\_Neg\_Pred\_Value Mean\_Precision Mean\_Recall  
## 1 0.9039721 0.9894046 0.9039721 0.9035477  
## 2 0.9039721 0.9894046 0.9039721 0.9035477  
## 3 0.9039721 0.9894046 0.9039721 0.9035477  
## 4 0.9039721 0.9894046 0.9039721 0.9035477  
## 5 0.9039721 0.9894046 0.9039721 0.9035477  
## Mean\_Detection\_Rate Mean\_Balanced\_Accuracy AccuracySD KappaSD  
## 1 0.09044775 0.9464717 0.004619496 0.005128387  
## 2 0.09044775 0.9464717 0.004619496 0.005128387  
## 3 0.09044775 0.9464717 0.004619496 0.005128387  
## 4 0.09044775 0.9464717 0.004619496 0.005128387  
## 5 0.09044775 0.9464717 0.004619496 0.005128387  
## Mean\_F1SD Mean\_SensitivitySD Mean\_SpecificitySD Mean\_Pos\_Pred\_ValueSD  
## 1 0.004374561 0.004549233 0.0005090543 0.004588896  
## 2 0.004374561 0.004549233 0.0005090543 0.004588896  
## 3 0.004374561 0.004549233 0.0005090543 0.004588896  
## 4 0.004374561 0.004549233 0.0005090543 0.004588896  
## 5 0.004374561 0.004549233 0.0005090543 0.004588896  
## Mean\_Neg\_Pred\_ValueSD Mean\_PrecisionSD Mean\_RecallSD Mean\_Detection\_RateSD  
## 1 0.0005276383 0.004588896 0.004549233 0.0004619496  
## 2 0.0005276383 0.004588896 0.004549233 0.0004619496  
## 3 0.0005276383 0.004588896 0.004549233 0.0004619496  
## 4 0.0005276383 0.004588896 0.004549233 0.0004619496  
## 5 0.0005276383 0.004588896 0.004549233 0.0004619496  
## Mean\_Balanced\_AccuracySD  
## 1 0.002528087  
## 2 0.002528087  
## 3 0.002528087  
## 4 0.002528087  
## 5 0.002528087

start\_time <- Sys.time()  
prediction=predict(svm\_l,test[,-1])  
end\_time <- Sys.time()  
end\_time - start\_time

## Time difference of 1.390716 secs

confusionMatrix(table(test$label,prediction))

## Confusion Matrix and Statistics  
##   
## prediction  
## Eight Five Four Nine One Seven Six Three Two Zero  
## Eight 350 14 2 2 5 0 4 9 6 1  
## Five 11 346 0 3 4 0 5 16 2 1  
## Four 1 1 375 21 1 0 1 0 4 0  
## Nine 4 2 15 339 2 17 0 5 1 1  
## One 4 0 0 0 470 0 0 1 2 1  
## Seven 2 3 10 22 3 424 0 0 1 0  
## Six 4 4 4 0 0 1 379 0 7 5  
## Three 18 18 0 0 10 4 2 381 8 5  
## Two 7 4 7 4 10 1 7 11 367 2  
## Zero 5 7 1 0 0 1 1 0 0 399  
##   
## Overall Statistics  
##   
## Accuracy : 0.9123   
## 95% CI : (0.9034, 0.9207)  
## No Information Rate : 0.1203   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.9025   
##   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: Eight Class: Five Class: Four Class: Nine  
## Sensitivity 0.86207 0.86717 0.90580 0.86701  
## Specificity 0.98866 0.98894 0.99234 0.98765  
## Pos Pred Value 0.89059 0.89175 0.92822 0.87824  
## Neg Pred Value 0.98528 0.98609 0.98972 0.98636  
## Prevalence 0.09671 0.09505 0.09862 0.09314  
## Detection Rate 0.08337 0.08242 0.08933 0.08075  
## Detection Prevalence 0.09362 0.09242 0.09624 0.09195  
## Balanced Accuracy 0.92536 0.92806 0.94907 0.92733  
## Class: One Class: Seven Class: Six Class: Three Class: Two  
## Sensitivity 0.9307 0.9464 0.94987 0.90071 0.92211  
## Specificity 0.9978 0.9891 0.99342 0.98278 0.98605  
## Pos Pred Value 0.9833 0.9118 0.93812 0.85426 0.87381  
## Neg Pred Value 0.9906 0.9936 0.99473 0.98881 0.99179  
## Prevalence 0.1203 0.1067 0.09505 0.10076 0.09481  
## Detection Rate 0.1120 0.1010 0.09028 0.09076 0.08742  
## Detection Prevalence 0.1139 0.1108 0.09624 0.10624 0.10005  
## Balanced Accuracy 0.9643 0.9677 0.97165 0.94175 0.95408  
## Class: Zero  
## Sensitivity 0.96145  
## Specificity 0.99603  
## Pos Pred Value 0.96377  
## Neg Pred Value 0.99577  
## Prevalence 0.09886  
## Detection Rate 0.09505  
## Detection Prevalence 0.09862  
## Balanced Accuracy 0.97874

## SVM – Polynomial

This is another variant of the Support Vector Machine. This variant uses the ‘Kernel Trick’   
As we can see below – ‘degree’, ‘scale’,’ C’ are the hyperparameters of the model.   
We try different combinations of the hyperparameters.   
It takes 14.37mins to train the model on different hyperparameters and give us the best answer.  
While training the best accuracy we get is 0.9368.  
The testing accuracy is 0.9509.

modelLookup('svmPoly')

## model parameter label forReg forClass probModel  
## 1 svmPoly degree Polynomial Degree TRUE TRUE TRUE  
## 2 svmPoly scale Scale TRUE TRUE TRUE  
## 3 svmPoly C Cost TRUE TRUE TRUE

grid\_search=expand.grid(degree=c(2,3),  
 C=c(seq(0.01,1,length=3)),  
 scale=c(0.001,0.01,0.1))  
  
  
train\_control <- trainControl(method = "cv",   
 number = 3,  
 summaryFunction=multiClassSummary  
)  
  
start\_time <- Sys.time()  
svm\_poly <- train(label~.,data=data,   
 method = "svmPoly",   
 trControl = train\_control,  
 tuneGrid = grid\_search,  
)

end\_time <- Sys.time()  
end\_time - start\_time

## Time difference of 14.37419 mins

svm\_poly$results %>% top\_n(3, wt = Accuracy) %>%arrange(desc(Accuracy))

## degree C scale Accuracy Kappa Mean\_F1 Mean\_Sensitivity  
## 1 2 0.010 0.001 0.936872 0.9298273 0.9361794 0.9360717  
## 2 2 0.505 0.001 0.936872 0.9298273 0.9361794 0.9360717  
## 3 2 1.000 0.001 0.936872 0.9298273 0.9361794 0.9360717  
## 4 2 0.010 0.010 0.936872 0.9298273 0.9361794 0.9360717  
## 5 2 0.505 0.010 0.936872 0.9298273 0.9361794 0.9360717  
## 6 2 1.000 0.010 0.936872 0.9298273 0.9361794 0.9360717  
## 7 2 0.010 0.100 0.936872 0.9298273 0.9361794 0.9360717  
## 8 2 0.505 0.100 0.936872 0.9298273 0.9361794 0.9360717  
## 9 2 1.000 0.100 0.936872 0.9298273 0.9361794 0.9360717  
## Mean\_Specificity Mean\_Pos\_Pred\_Value Mean\_Neg\_Pred\_Value Mean\_Precision  
## 1 0.9929903 0.9368374 0.9929988 0.9368374  
## 2 0.9929903 0.9368374 0.9929988 0.9368374  
## 3 0.9929903 0.9368374 0.9929988 0.9368374  
## 4 0.9929903 0.9368374 0.9929988 0.9368374  
## 5 0.9929903 0.9368374 0.9929988 0.9368374  
## 6 0.9929903 0.9368374 0.9929988 0.9368374  
## 7 0.9929903 0.9368374 0.9929988 0.9368374  
## 8 0.9929903 0.9368374 0.9929988 0.9368374  
## 9 0.9929903 0.9368374 0.9929988 0.9368374  
## Mean\_Recall Mean\_Detection\_Rate Mean\_Balanced\_Accuracy AccuracySD  
## 1 0.9360717 0.0936872 0.964531 0.003664646  
## 2 0.9360717 0.0936872 0.964531 0.003664646  
## 3 0.9360717 0.0936872 0.964531 0.003664646  
## 4 0.9360717 0.0936872 0.964531 0.003664646  
## 5 0.9360717 0.0936872 0.964531 0.003664646  
## 6 0.9360717 0.0936872 0.964531 0.003664646  
## 7 0.9360717 0.0936872 0.964531 0.003664646  
## 8 0.9360717 0.0936872 0.964531 0.003664646  
## 9 0.9360717 0.0936872 0.964531 0.003664646  
## KappaSD Mean\_F1SD Mean\_SensitivitySD Mean\_SpecificitySD  
## 1 0.004071669 0.00389956 0.003666004 0.0004036429  
## 2 0.004071669 0.00389956 0.003666004 0.0004036429  
## 3 0.004071669 0.00389956 0.003666004 0.0004036429  
## 4 0.004071669 0.00389956 0.003666004 0.0004036429  
## 5 0.004071669 0.00389956 0.003666004 0.0004036429  
## 6 0.004071669 0.00389956 0.003666004 0.0004036429  
## 7 0.004071669 0.00389956 0.003666004 0.0004036429  
## 8 0.004071669 0.00389956 0.003666004 0.0004036429  
## 9 0.004071669 0.00389956 0.003666004 0.0004036429  
## Mean\_Pos\_Pred\_ValueSD Mean\_Neg\_Pred\_ValueSD Mean\_PrecisionSD Mean\_RecallSD  
## 1 0.004213273 0.0004060666 0.004213273 0.003666004  
## 2 0.004213273 0.0004060666 0.004213273 0.003666004  
## 3 0.004213273 0.0004060666 0.004213273 0.003666004  
## 4 0.004213273 0.0004060666 0.004213273 0.003666004  
## 5 0.004213273 0.0004060666 0.004213273 0.003666004  
## 6 0.004213273 0.0004060666 0.004213273 0.003666004  
## 7 0.004213273 0.0004060666 0.004213273 0.003666004  
## 8 0.004213273 0.0004060666 0.004213273 0.003666004  
## 9 0.004213273 0.0004060666 0.004213273 0.003666004  
## Mean\_Detection\_RateSD Mean\_Balanced\_AccuracySD  
## 1 0.0003664646 0.002034652  
## 2 0.0003664646 0.002034652  
## 3 0.0003664646 0.002034652  
## 4 0.0003664646 0.002034652  
## 5 0.0003664646 0.002034652  
## 6 0.0003664646 0.002034652  
## 7 0.0003664646 0.002034652  
## 8 0.0003664646 0.002034652  
## 9 0.0003664646 0.002034652

start\_time <- Sys.time()  
prediction=predict(svm\_poly,test[,-1])  
end\_time <- Sys.time()  
end\_time - start\_time

## Time difference of 12.59481 secs

confusionMatrix(table(test$label,prediction))

## Confusion Matrix and Statistics  
##   
## prediction  
## Eight Five Four Nine One Seven Six Three Two Zero  
## Eight 365 8 1 2 5 0 2 7 3 0  
## Five 5 367 0 1 3 0 5 5 1 1  
## Four 0 0 390 10 1 0 1 0 2 0  
## Nine 2 2 9 353 2 11 0 4 0 3  
## One 2 1 0 0 472 0 0 1 2 0  
## Seven 0 0 10 6 4 444 0 0 1 0  
## Six 2 4 1 0 0 0 392 0 1 4  
## Three 10 8 0 1 7 3 1 408 5 3  
## Two 3 1 3 2 4 2 3 4 397 1  
## Zero 3 1 1 1 0 1 3 0 0 404  
##   
## Overall Statistics  
##   
## Accuracy : 0.9509   
## 95% CI : (0.944, 0.9573)  
## No Information Rate : 0.1186   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.9454   
##   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: Eight Class: Five Class: Four Class: Nine  
## Sensitivity 0.93112 0.93622 0.93976 0.93883  
## Specificity 0.99264 0.99448 0.99630 0.99137  
## Pos Pred Value 0.92875 0.94588 0.96535 0.91451  
## Neg Pred Value 0.99290 0.99344 0.99341 0.99397  
## Prevalence 0.09338 0.09338 0.09886 0.08957  
## Detection Rate 0.08695 0.08742 0.09290 0.08409  
## Detection Prevalence 0.09362 0.09242 0.09624 0.09195  
## Balanced Accuracy 0.96188 0.96535 0.96803 0.96510  
## Class: One Class: Seven Class: Six Class: Three Class: Two  
## Sensitivity 0.9478 0.9631 0.96314 0.95105 0.96359  
## Specificity 0.9984 0.9944 0.99683 0.98992 0.99392  
## Pos Pred Value 0.9874 0.9548 0.97030 0.91480 0.94524  
## Neg Pred Value 0.9930 0.9954 0.99605 0.99440 0.99603  
## Prevalence 0.1186 0.1098 0.09695 0.10219 0.09814  
## Detection Rate 0.1124 0.1058 0.09338 0.09719 0.09457  
## Detection Prevalence 0.1139 0.1108 0.09624 0.10624 0.10005  
## Balanced Accuracy 0.9731 0.9788 0.97999 0.97048 0.97876  
## Class: Zero  
## Sensitivity 0.97115  
## Specificity 0.99736  
## Pos Pred Value 0.97585  
## Neg Pred Value 0.99683  
## Prevalence 0.09909  
## Detection Rate 0.09624  
## Detection Prevalence 0.09862  
## Balanced Accuracy 0.98425

## SVM – Radial

Here we will try the SVM model with kernel = ‘Radial’   
We can see it does not perform well on our dataset.   
The training accuracy is 0.1121  
The time it took to complete the entire training is 12.63 mins.  
The testing accuracy is 0.1139

modelLookup('svmRadial')

## model parameter label forReg forClass probModel  
## 1 svmRadial sigma Sigma TRUE TRUE TRUE  
## 2 svmRadial C Cost TRUE TRUE TRUE

grid\_search=expand.grid(sigma=c(seq(0.1,1,3)),  
 C=c(seq(0.01,1,length=3))  
 )  
  
train\_control <- trainControl(method = "cv",   
 number = 3,  
 summaryFunction=multiClassSummary  
)  
  
start\_time <- Sys.time()  
svm\_r <- train(label~.,data=data,   
 method = "svmRadial",   
 trControl = train\_control,  
 tuneGrid = grid\_search,  
)

end\_time <- Sys.time()  
end\_time - start\_time

## Time difference of 12.63593 mins

svm\_r$results %>% top\_n(3, wt = Accuracy) %>%arrange(desc(Accuracy))

## sigma C Accuracy Kappa Mean\_F1 Mean\_Sensitivity Mean\_Specificity  
## 1 0.1 0.010 0.1121964 0 NaN 0.1 0.9  
## 2 0.1 0.505 0.1121964 0 NaN 0.1 0.9  
## 3 0.1 1.000 0.1121964 0 NaN 0.1 0.9  
## Mean\_Pos\_Pred\_Value Mean\_Neg\_Pred\_Value Mean\_Precision Mean\_Recall  
## 1 NaN NaN NaN 0.1  
## 2 NaN NaN NaN 0.1  
## 3 NaN NaN NaN 0.1  
## Mean\_Detection\_Rate Mean\_Balanced\_Accuracy AccuracySD KappaSD Mean\_F1SD  
## 1 0.01121964 0.5 0.000122433 0 NA  
## 2 0.01121964 0.5 0.000122433 0 NA  
## 3 0.01121964 0.5 0.000122433 0 NA  
## Mean\_SensitivitySD Mean\_SpecificitySD Mean\_Pos\_Pred\_ValueSD  
## 1 0 0 NA  
## 2 0 0 NA  
## 3 0 0 NA  
## Mean\_Neg\_Pred\_ValueSD Mean\_PrecisionSD Mean\_RecallSD Mean\_Detection\_RateSD  
## 1 NA NA 0 1.22433e-05  
## 2 NA NA 0 1.22433e-05  
## 3 NA NA 0 1.22433e-05  
## Mean\_Balanced\_AccuracySD  
## 1 0  
## 2 0  
## 3 0

start\_time <- Sys.time()  
prediction=predict(svm\_r,test[,-1])  
end\_time <- Sys.time()  
end\_time - start\_time

## Time difference of 1.617556 mins

confusionMatrix(table(test$label,prediction))

## Confusion Matrix and Statistics  
##   
## prediction  
## Eight Five Four Nine One Seven Six Three Two Zero  
## Eight 0 0 0 0 393 0 0 0 0 0  
## Five 0 0 0 0 388 0 0 0 0 0  
## Four 0 0 0 0 404 0 0 0 0 0  
## Nine 0 0 0 0 386 0 0 0 0 0  
## One 0 0 0 0 478 0 0 0 0 0  
## Seven 0 0 0 0 465 0 0 0 0 0  
## Six 0 0 0 0 404 0 0 0 0 0  
## Three 0 0 0 0 446 0 0 0 0 0  
## Two 0 0 0 0 420 0 0 0 0 0  
## Zero 0 0 0 0 414 0 0 0 0 0  
##   
## Overall Statistics  
##   
## Accuracy : 0.1139   
## 95% CI : (0.1044, 0.1239)  
## No Information Rate : 1   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : 0   
##   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: Eight Class: Five Class: Four Class: Nine  
## Sensitivity NA NA NA NA  
## Specificity 0.90638 0.90758 0.90376 0.90805  
## Pos Pred Value NA NA NA NA  
## Neg Pred Value NA NA NA NA  
## Prevalence 0.00000 0.00000 0.00000 0.00000  
## Detection Rate 0.00000 0.00000 0.00000 0.00000  
## Detection Prevalence 0.09362 0.09242 0.09624 0.09195  
## Balanced Accuracy NA NA NA NA  
## Class: One Class: Seven Class: Six Class: Three Class: Two  
## Sensitivity 0.1139 NA NA NA NA  
## Specificity NA 0.8892 0.90376 0.8938 0.9  
## Pos Pred Value NA NA NA NA NA  
## Neg Pred Value NA NA NA NA NA  
## Prevalence 1.0000 0.0000 0.00000 0.0000 0.0  
## Detection Rate 0.1139 0.0000 0.00000 0.0000 0.0  
## Detection Prevalence 0.1139 0.1108 0.09624 0.1062 0.1  
## Balanced Accuracy NA NA NA NA NA  
## Class: Zero  
## Sensitivity NA  
## Specificity 0.90138  
## Pos Pred Value NA  
## Neg Pred Value NA  
## Prevalence 0.00000  
## Detection Rate 0.00000  
## Detection Prevalence 0.09862  
## Balanced Accuracy NA

## Random Forest

The last model we will try is Random Forest. It is an ensemble method. It creates many Decision Trees and then ensembles the output of all the trees and gives us a result.  
The hyperparameter of the model is ‘mtry’. We try a few values of ‘mtry’  
The time to complete the training process is 12.37 mins.  
The training accuracy is 0.9182  
The testing accuracy is 0.9259

modelLookup('rf')

## model parameter label forReg forClass probModel  
## 1 rf mtry #Randomly Selected Predictors TRUE TRUE TRUE

grid\_search=expand.grid(mtry=c(1:4))  
  
  
train\_control <- trainControl(method = "cv",   
 number = 3,  
 summaryFunction=multiClassSummary  
)  
  
start\_time <- Sys.time()  
random\_forest <- train(label~.,data=data,   
 method = "rf",   
 trControl = train\_control,  
 tuneGrid = grid\_search,  
)

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

end\_time <- Sys.time()  
end\_time - start\_time

## Time difference of 12.37853 mins

random\_forest$results %>% top\_n(3, wt = Accuracy) %>%arrange(desc(Accuracy))

## mtry Accuracy Kappa Mean\_F1 Mean\_Sensitivity Mean\_Specificity  
## 1 4 0.9182924 0.9091534 0.9172713 0.9165366 0.9909110  
## 2 3 0.8935211 0.8815840 0.8923946 0.8911356 0.9881397  
## 3 2 0.8089532 0.7872525 0.8045851 0.8018385 0.9786468  
## Mean\_Pos\_Pred\_Value Mean\_Neg\_Pred\_Value Mean\_Precision Mean\_Recall  
## 1 0.9196984 0.9909637 0.9196984 0.9165366  
## 2 0.8981047 0.9882550 0.8981047 0.8911356  
## 3 0.8449564 0.9792798 0.8449564 0.8018385  
## Mean\_Detection\_Rate Mean\_Balanced\_Accuracy AccuracySD KappaSD  
## 1 0.09182924 0.9537238 0.004246819 0.004720915  
## 2 0.08935211 0.9396376 0.009339476 0.010389341  
## 3 0.08089532 0.8902426 0.004797437 0.005351052  
## Mean\_F1SD Mean\_SensitivitySD Mean\_SpecificitySD Mean\_Pos\_Pred\_ValueSD  
## 1 0.004596305 0.004350061 0.0004691197 0.005338675  
## 2 0.010093487 0.009789769 0.0010350990 0.011080680  
## 3 0.004575617 0.004603449 0.0005400232 0.003757622  
## Mean\_Neg\_Pred\_ValueSD Mean\_PrecisionSD Mean\_RecallSD Mean\_Detection\_RateSD  
## 1 0.0004717081 0.005338675 0.004350061 0.0004246819  
## 2 0.0010317101 0.011080680 0.009789769 0.0009339476  
## 3 0.0005177934 0.003757622 0.004603449 0.0004797437  
## Mean\_Balanced\_AccuracySD  
## 1 0.002409505  
## 2 0.005412427  
## 3 0.002569777

start\_time <- Sys.time()  
prediction=predict(random\_forest,test[,-1])  
end\_time <- Sys.time()  
end\_time - start\_time

## Time difference of 0.8374429 secs

confusionMatrix(table(test$label,prediction))

## Confusion Matrix and Statistics  
##   
## prediction  
## Eight Five Four Nine One Seven Six Three Two Zero  
## Eight 349 6 2 4 9 1 2 12 1 7  
## Five 1 345 3 4 3 0 9 18 1 4  
## Four 2 0 370 25 2 2 2 0 1 0  
## Nine 4 0 11 343 3 13 1 8 0 3  
## One 1 1 0 0 471 0 0 1 4 0  
## Seven 1 0 10 13 8 431 0 1 0 1  
## Six 1 7 1 0 0 0 390 0 0 5  
## Three 12 2 0 7 10 8 2 389 11 5  
## Two 5 0 5 0 5 4 3 1 394 3  
## Zero 3 0 1 0 0 0 4 1 0 405  
##   
## Overall Statistics  
##   
## Accuracy : 0.9259   
## 95% CI : (0.9176, 0.9337)  
## No Information Rate : 0.1217   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.9176   
##   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: Eight Class: Five Class: Four Class: Nine  
## Sensitivity 0.92084 0.95568 0.91811 0.86616  
## Specificity 0.98848 0.98879 0.99104 0.98869  
## Pos Pred Value 0.88804 0.88918 0.91584 0.88860  
## Neg Pred Value 0.99212 0.99580 0.99130 0.98610  
## Prevalence 0.09028 0.08599 0.09600 0.09433  
## Detection Rate 0.08313 0.08218 0.08814 0.08171  
## Detection Prevalence 0.09362 0.09242 0.09624 0.09195  
## Balanced Accuracy 0.95466 0.97224 0.95458 0.92743  
## Class: One Class: Seven Class: Six Class: Three Class: Two  
## Sensitivity 0.9217 0.9390 0.94431 0.90255 0.95631  
## Specificity 0.9981 0.9909 0.99630 0.98487 0.99313  
## Pos Pred Value 0.9854 0.9269 0.96535 0.87220 0.93810  
## Neg Pred Value 0.9892 0.9925 0.99394 0.98881 0.99524  
## Prevalence 0.1217 0.1093 0.09838 0.10267 0.09814  
## Detection Rate 0.1122 0.1027 0.09290 0.09266 0.09385  
## Detection Prevalence 0.1139 0.1108 0.09624 0.10624 0.10005  
## Balanced Accuracy 0.9599 0.9650 0.97031 0.94371 0.97472  
## Class: Zero  
## Sensitivity 0.93533  
## Specificity 0.99761  
## Pos Pred Value 0.97826  
## Neg Pred Value 0.99260  
## Prevalence 0.10314  
## Detection Rate 0.09647  
## Detection Prevalence 0.09862  
## Balanced Accuracy 0.96647

## 

## Conclusion

Comparing all our models with the accuracy on the training set as well as the testing set:

|  |  |  |
| --- | --- | --- |
| Model | Cross Validation - Accuracy | Testing Accuracy |
| Naïve Bayes | 0.4571 | 0.4636 |
| Decision Tree | 0.7291 | 0.7306 |
| KNN | 0.9252 | 0.9366 |
| SVM-Linear | 0.9045 | 0.9123 |
| SVM-Poly | 0.9369 | 0.9509 |
| SVM-Radial | 0.1121 | 0.1139 |
| Random Forest | 0.9182 | 0.9259 |

As we can see the models, we tried this week are more complex than the simple ones we tired last week. We get a better accuracy for all the new models other than the SVM-Radial.