IoA Project - MNIST_R

Aravind

March 13, 2018

DIGIT RECOGNIZER - IOA PROJECT

Read data : https://www.kaggle.com/c/digit-recognizer

```
train <- read.csv("C:/Users/Aravind/Documents/Digit_Recognizer/train.csv")
test <- read.csv("C:/Users/Aravind/Documents/Digit_Recognizer/test.csv")
```

Dimensions of train and test

```
dim(train)
```

```
## [1] 42000   785
```

```
dim(test)
```

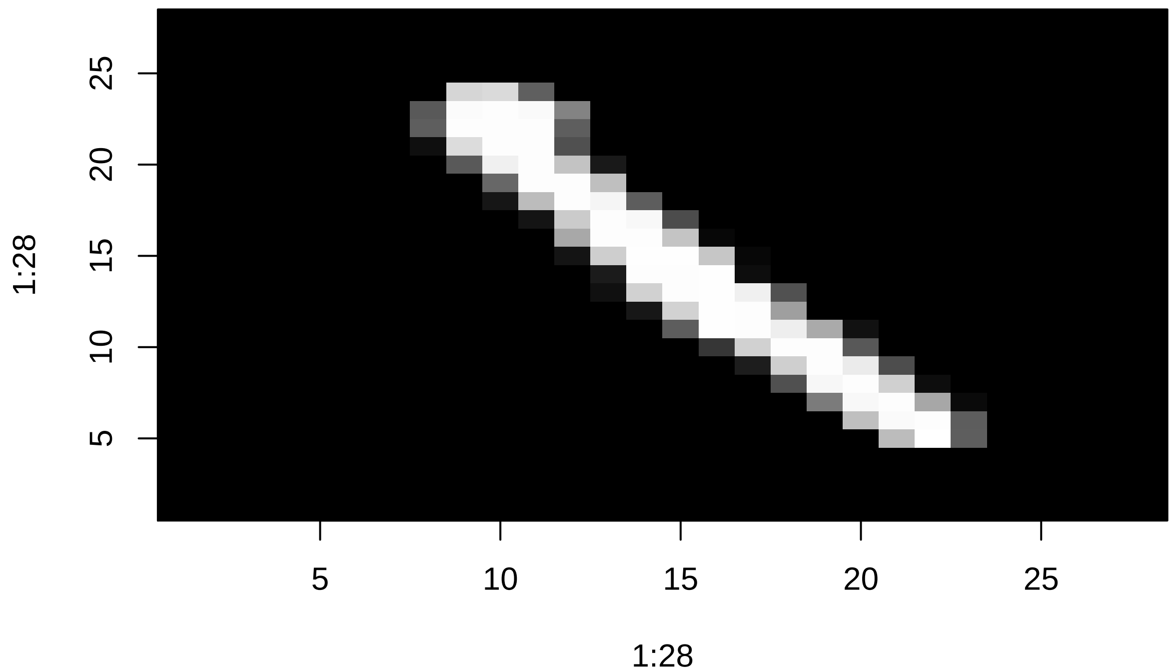
```
## [1] 28000   784
```

```
train$label <- as.factor(train$label)
```

Visualization

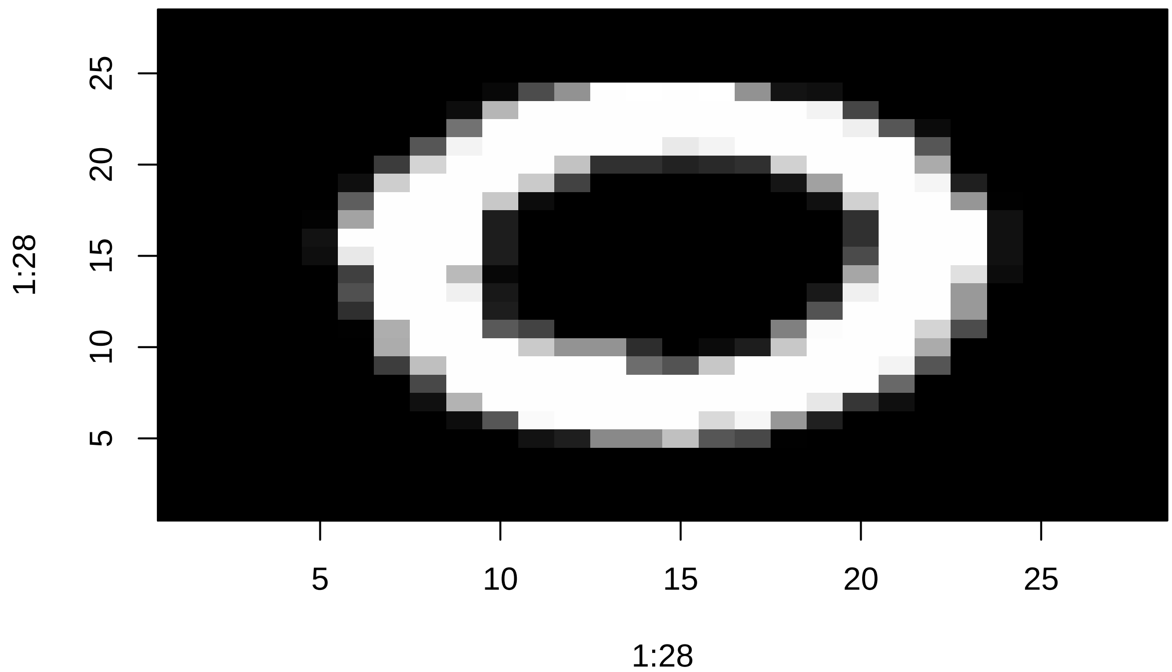
1st Image

```
img<-matrix((train[1,2:ncol(train)]), nrow=28, ncol=28) #For the 1st Image
img_numbers <- apply(img, 2, as.numeric)
image(1:28, 1:28, img_numbers, col=gray((0:255)/255))
```



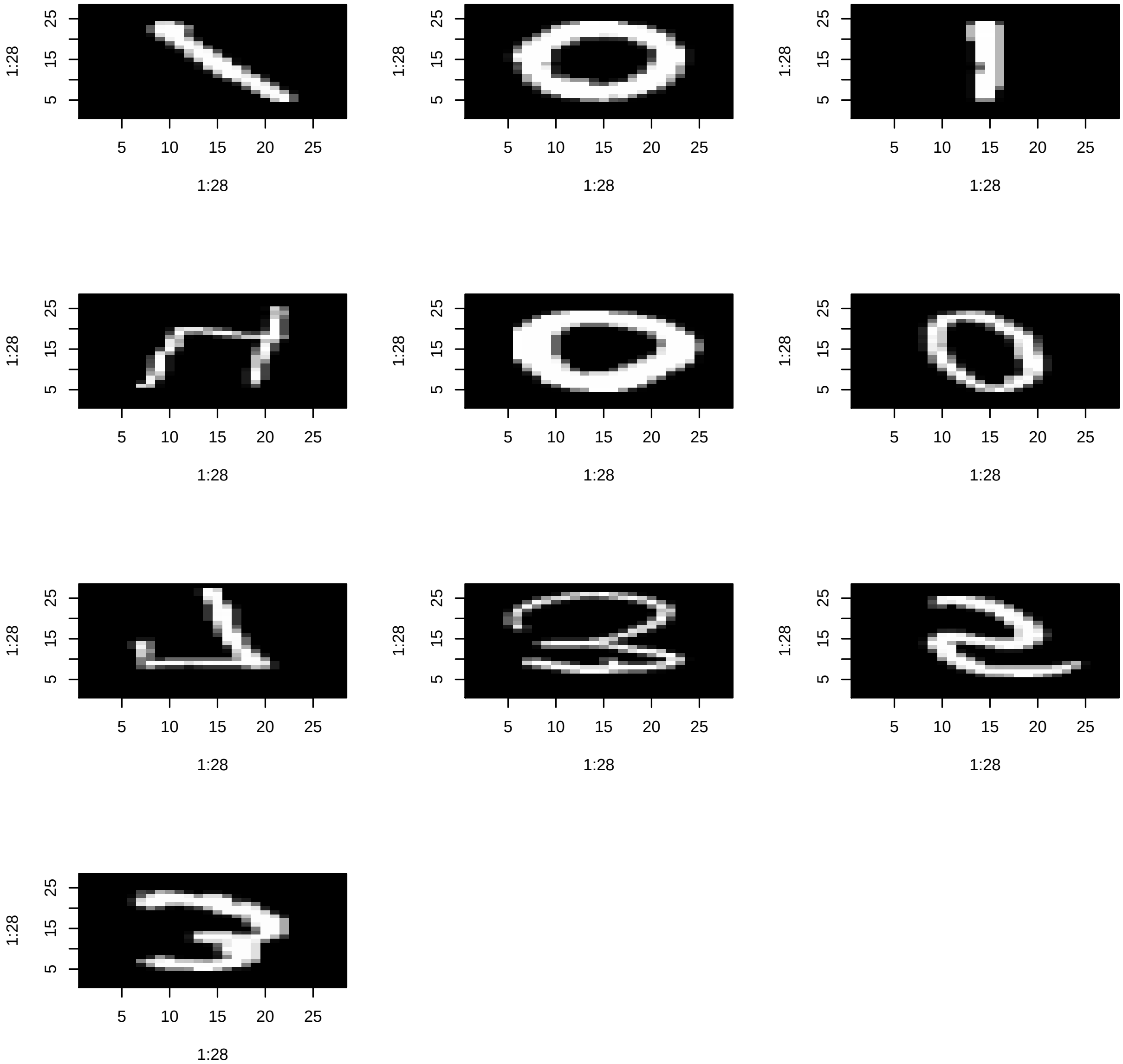
2nd Image

```
img<-matrix((train[2,2:ncol(train)]), nrow=28, ncol=28) #For the 2nd Image
img_numbers <- apply(img, 2, as.numeric)
image(1:28, 1:28, img_numbers, col=gray((0:255)/255))
```



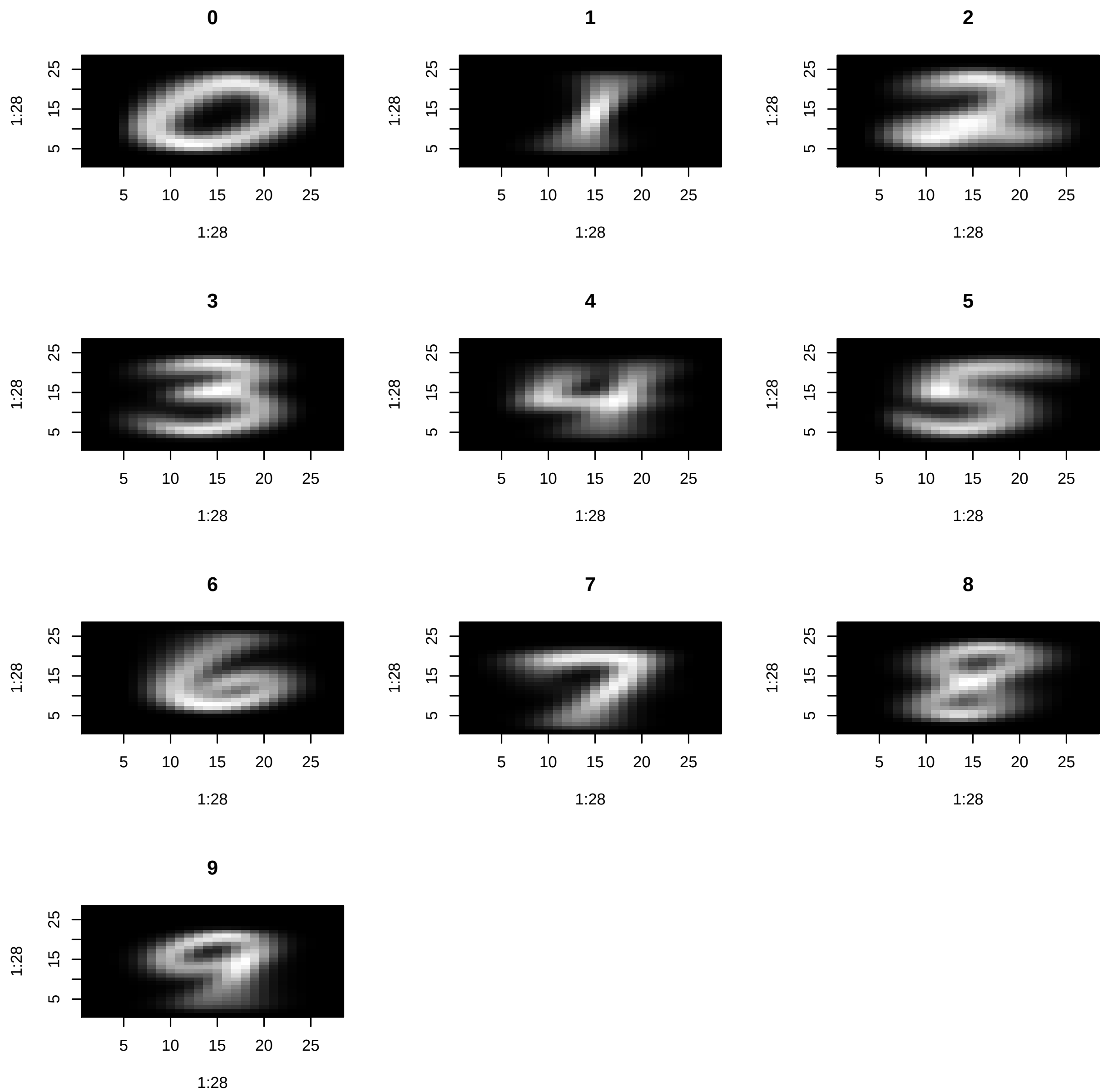
For the first ten rows

```
par(mfrow=c(4, 3))
for (i in 1:10){
    img<-matrix((train[i,2:ncol(train)]), nrow=28, ncol=28)
    img_numbers <- apply(img, 2, as.numeric)
    image(1:28, 1:28, img_numbers, col=gray((0:255)/255))
}
```



Average image of each digit

```
par(mfrow=c(4,3))
img<-array(dim=c(10,28*28))
for(i in 0:9){
  img[i+1,]<-apply(train[train[,1]==i,-1],2,sum)
  img[i+1,]<-img[i+1,]/255*255
  im<-array(img[i+1,],dim=c(28,28))
  im<-im[,28:1] #right side up
  image(1:28,1:28,im,col = grey(0:255/255),main=i)
}
```



Split train data

```r
require(caret)
```

```
## Loading required package: caret
## Loading required package: lattice
## Loading required package: ggplot2
```

```r
set.seed(123)
index <- createDataPartition(train$label, p=0.80, list = F)
train_set <- train[index,]
```

```r
test_set <- train[-index,]
```

Scaling and Centering

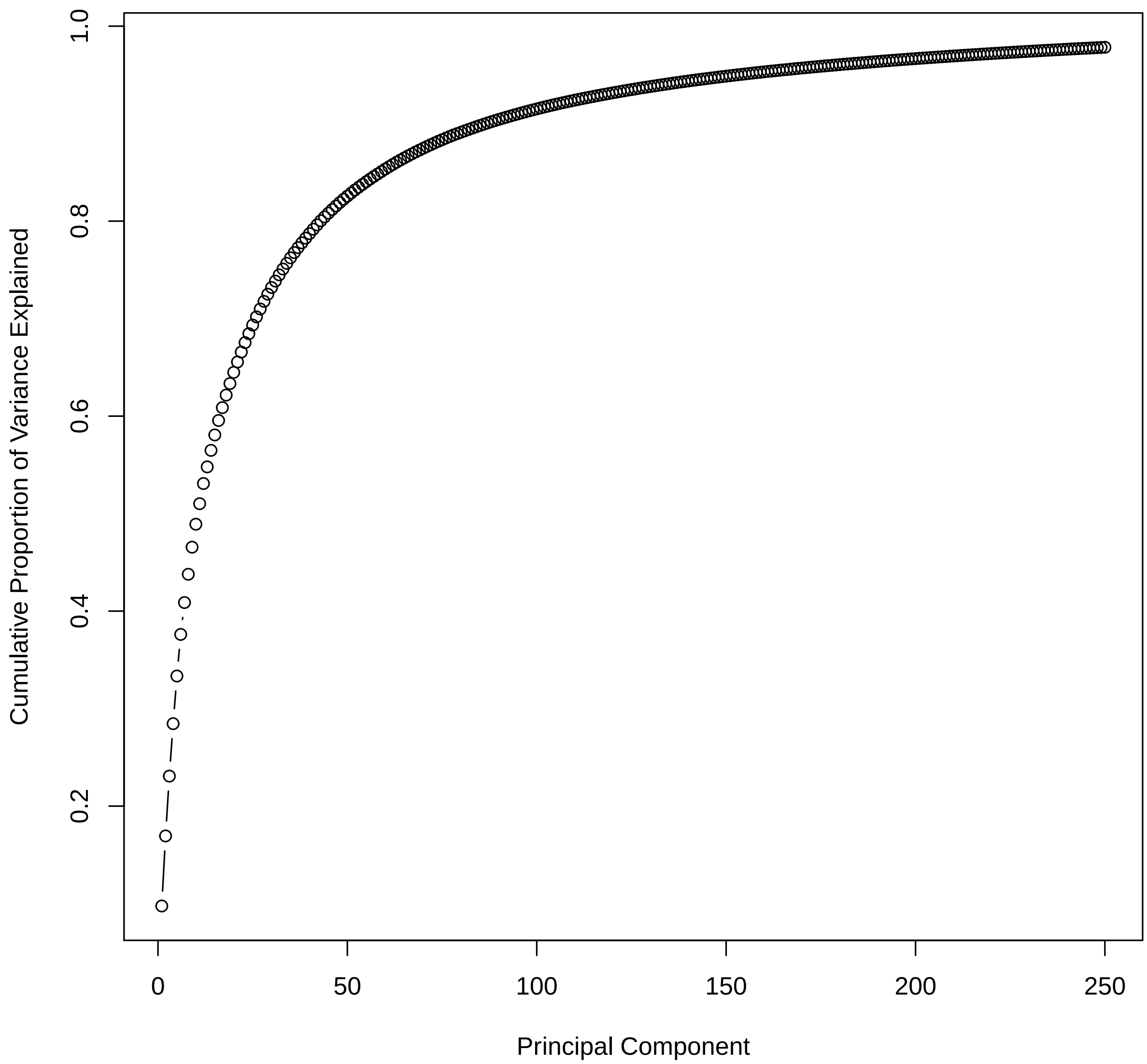
```r
X <- train_set[,-1]
X_scale <- X/255
X_center <- scale(X_scale, center = T, scale = F)
Y <- test_set[,-1]
Y_scale <- Y/255
Y_center <- scale(Y_scale, center = T, scale = F)
```

Principal Component Analysis

```r
pca<-princomp(X_center)
std_dev <- pca[1:250]$sdev
pr_var <- std_dev^2
prop_varex <- pr_var/sum(pr_var)
```

Plot

```r
plot(cumsum(prop_varex[1:250]), xlab = "Principal Component",
     ylab = "Cumulative Proportion of Variance Explained",
     type = "b")
```



Using first 250 components we can explain ~100% of variation

Spliting PCA components into train and test

```
train_set_pca <- data.frame(predict(pca, newdata = train_set[,-1]))[1:250]
train_set_pca$label <- train_set$label
test_set_pca <- data.frame(predict(pca, newdata = test_set[,-1]))[1:250]
test_set_pca$label <- test_set$label
```

Model Building

DECISION TREE

```r
require(rpart)
```

```
## Loading required package: rpart
```

```r
model_rpart <- rpart(label ~., data = train_set_pca)
```

Predict test_set

```r
pred_rpart <- predict(model_rpart, newdata = test_set_pca[-251],type = 'class')
```

Confusion matrix

```r
cm = table('Actual Digit' = test_set_pca[, 251], 'Predicted Digit' =  pred_rpart)
cm
```

```
##             Predicted Digit
## Actual Digit   0   1   2   3   4   5   6   7   8   9
##            0 475   0  95 136   1  75  12  12   7  13
##            1   0 778  43  30   0  38  23   0  24   0
##            2  32   5 597  75   7  10  36   3  43  27
##            3  20   4  61 614   1  35  38   0  87  10
##            4   0  18  35  19 468  13  11  12  31 207
##            5  27   2 102 237  21 208  22  10  94  36
##            6  43   2 108  61   4   9 443  26  27 104
##            7   0  49  32  24  15  31   1 487  35 206
##            8  25   6  83  81   7  67  14   1 483  45
##            9   4  47  15  22  98  33   2  54  32 530
```

Accuracy

```r
print(sum(diag(cm))/sum(cm))
```

```
## [1] 0.6054073
```

Decision tree gives an accurary of 60%

Random Forest Model

```r
require(randomForest)
```

```
## Loading required package: randomForest
## randomForest 4.6-12
## Type rfNews() to see new features/changes/bug fixes.
##
## Attaching package: 'randomForest'
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
model <- randomForest(label ~., data = train_set_pca, ntree = 100)
```

Predict test_set

```
pred <- predict(model, newdata = test_set_pca[-251])
```

Confusion Matrix

```
cm = table('Actual Digit' = test_set_pca[, 251], 'Predicted Digit' =  pred)
cm
```

```
##             Predicted Digit
## Actual Digit   0   1   2   3   4   5   6   7   8   9
##            0 807   0   2   1   0   4   7   0   5   0
##            1   0 914   5   2   1   4   5   2   2   1
##            2   6   3 777  12   8   3   2   3  18   3
##            3   3   0  16 792   2  23   5   5  16   8
##            4   3   3   1   0 772   3   3   2   5  22
##            5   3   1   3  28   6 697   8   2   5   6
##            6   4   0   5   0   3   3 810   0   2   0
##            7   1   2   3   0  10   2   1 830   8  23
##            8   1   8  10  21  10  16   9   1 728   8
##            9   6   2   2  11  29   7   3  16   4 757
```

Accuracy

```
print(sum(diag(cm))/sum(cm))
```

```
## [1] 0.9390186
```

Random Forest gives an accuracy of 93.9%

K-NN

```r
require(class)
```

```
## Loading required package: class
```

```r
model_knn_pred <- knn(train = train_set_pca[,-251],
                  test = test_set_pca[,-251],
                  cl = train_set_pca[,251],
                  k=3)
```

Confusion Matrix

```r
cm_knn <- table('Actual Digit' = test_set_pca[, 251], 'Predicted Digit' =  model_knn_pred)
cm_knn
```

```
##              Predicted Digit
## Actual Digit   0   1   2   3   4   5   6   7   8   9
##            0 819   1   0   0   0   2   3   0   1   0
##            1   0 929   2   0   0   0   1   3   0   1
##            2   6  10 796   1   1   0   1  17   2   1
##            3   1   3   7 838   0   7   1   2   8   3
##            4   0   7   0   0 783   0   1   0   0  23
##            5   0   0   1  16   1 726   9   0   3   3
##            6   4   0   0   0   1   1 821   0   0   0
##            7   0   7   1   0   1   0   0 858   0  13
##            8   3   9   6   6   5  10   3   1 754  15
##            9   1   0   2   4  14   1   1   9   1 804
```

Accuracy

```r
print(sum(diag(cm_knn))/sum(cm_knn))
```

```
## [1] 0.96808
```

K-NN gives an accuracy of 96.8%

SVM

```r
require(e1071)
```

```
## Loading required package: e1071
```

```r
model_svm = svm(formula = label ~ .,
                data = train_set_pca,
                type = 'C-classification',
                kernel = 'linear')
```

```r
pred_svm <- predict(model_svm, newdata = test_set_pca[-251])
```

Confusion Matrix

```r
cm_svm = table('Actual Digit' = test_set_pca[, 251], 'Predicted Digit' =  pred_svm)
cm_svm
```

```
##             Predicted Digit
## Actual Digit   0   1   2   3   4   5   6   7   8   9
##            0 797   0   4   0   3   6   9   0   7   0
##            1   0 920   4   3   0   2   1   2   4   0
##            2  14  13 722  17  18   7  10  11  21   2
##            3   5   7  14 783   3  26   0  13  15   4
##            4   0   3   5   0 771   1   7   5   4  18
##            5   9   4   6  29   7 675   9   3  13   4
##            6   7   3   7   4   4  13 788   1   0   0
##            7   2   4   6   6   9   1   0 825   3  24
##            8   7  19   5  21   7  24   1   1 723   4
##            9   4   2   3   4  39   6   0  31   9 739
```

Accuracy

```r
print(sum(diag(cm_svm))/sum(cm_svm))
```

```
## [1] 0.9222249
```

SVM gives an accuracy of 92%

XG BOOST

```r
require(xgboost)
```

```
## Loading required package: xgboost
```

```r
model_xgb <- xgboost(data = as.matrix(train_set_pca[-251]), label = train_set_pca$label, nrounds = 500)
```

By using nround = 500, we are able to reduce rmse value from 4.2 to 0.16. The lesser it is the better your model performs.

Prediction

```r
pred_xgb <- predict(model_xgb, newdata = as.matrix(test_set_pca[-251]))
pred_xgb <- (pred_xgb >= 0.5)
```

Confusion Matrix

```r
cm_xgb <- table('Actual Digit' = test_set_pca[, 251], 'Predicted Digit' =  pred_xgb)
cm_xgb
```

```
##             Predicted Digit
## Actual Digit FALSE TRUE
##            0   126  700
##            1     1  935
##            2     2  833
##            3     0  870
##            4     0  814
##            5     0  759
##            6     1  826
##            7     0  880
##            8     0  812
##            9     0  837
```

Accuracy

```r
print(sum(cm_xgb[,2])/sum(cm_xgb))
```

```
## [1] 0.9845164
```

XGBOOST gives an accuracy of 98.45%

Plot accuracy of each digits

```r
total <- apply(cm_xgb,1, sum)
total <- array(total)
total
```

```
##  [1] 826 936 835 870 814 759 827 880 812 837
```

```r
crt <- c()
for (i in 1:10){
 crt[i] <- cm_xgb[i,2]
}
crt
```

```
##  [1] 700 935 833 870 814 759 826 880 812 837
```

```r
result <- crt/total
result
```

```
##  [1] 0.8474576 0.9989316 0.9976048 1.0000000 1.0000000 1.0000000 0.9987908
##  [8] 1.0000000 1.0000000 1.0000000
```

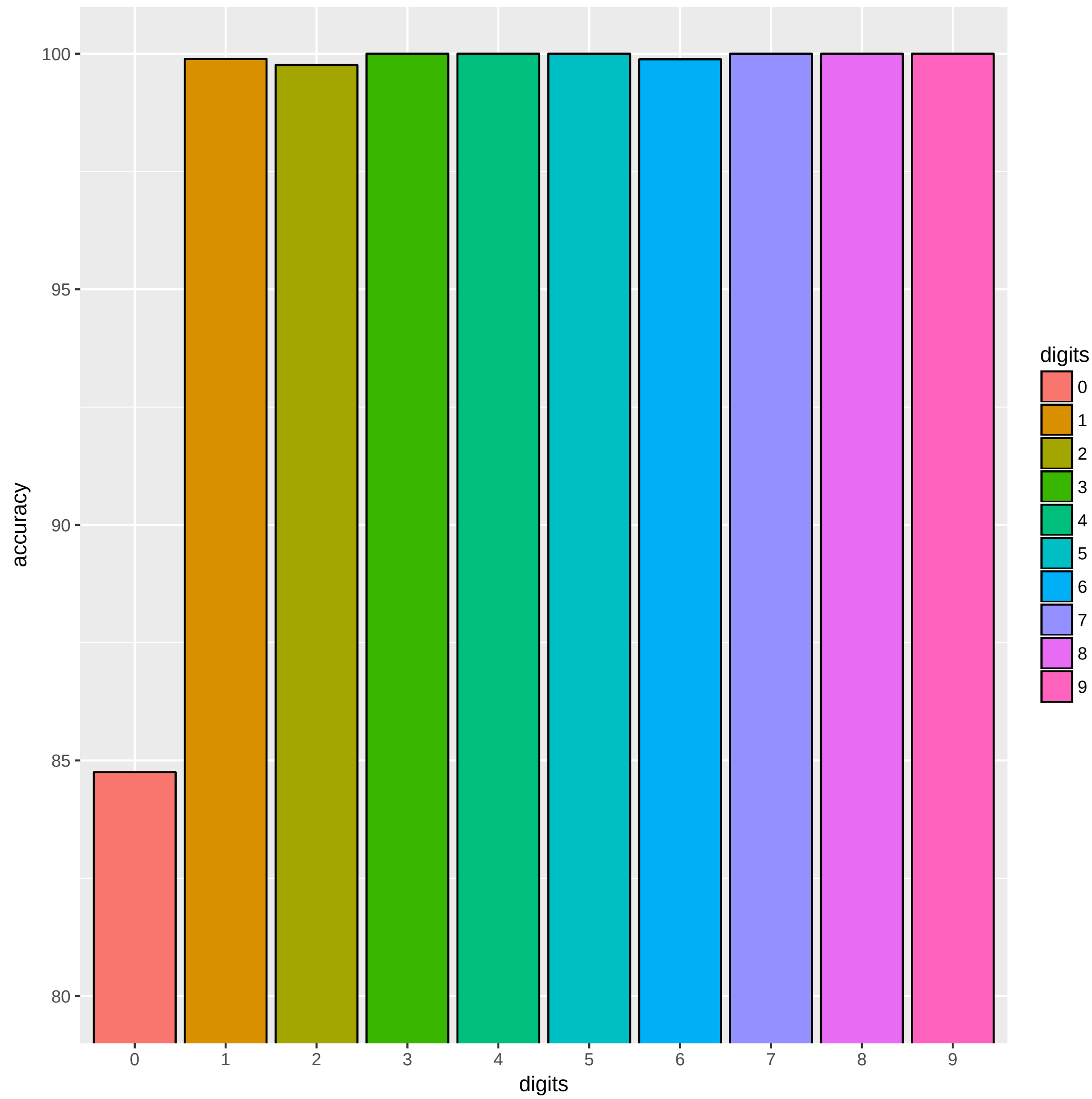
```r
digits <- c(0:9)
digits
```

```
##  [1] 0 1 2 3 4 5 6 7 8 9
```

```r
result_df <- data.frame(digits = digits, accuracy = result)
result_df$digits <- as.factor(result_df$digits)
result_df$accuracy <- result_df$accuracy*100
result_df$accuracy <- round(result_df$accuracy, digits = 2)
result_df
```

```
##    digits accuracy
## 1       0    84.75
## 2       1    99.89
## 3       2    99.76
## 4       3   100.00
## 5       4   100.00
## 6       5   100.00
## 7       6    99.88
## 8       7   100.00
## 9       8   100.00
## 10      9   100.00
```

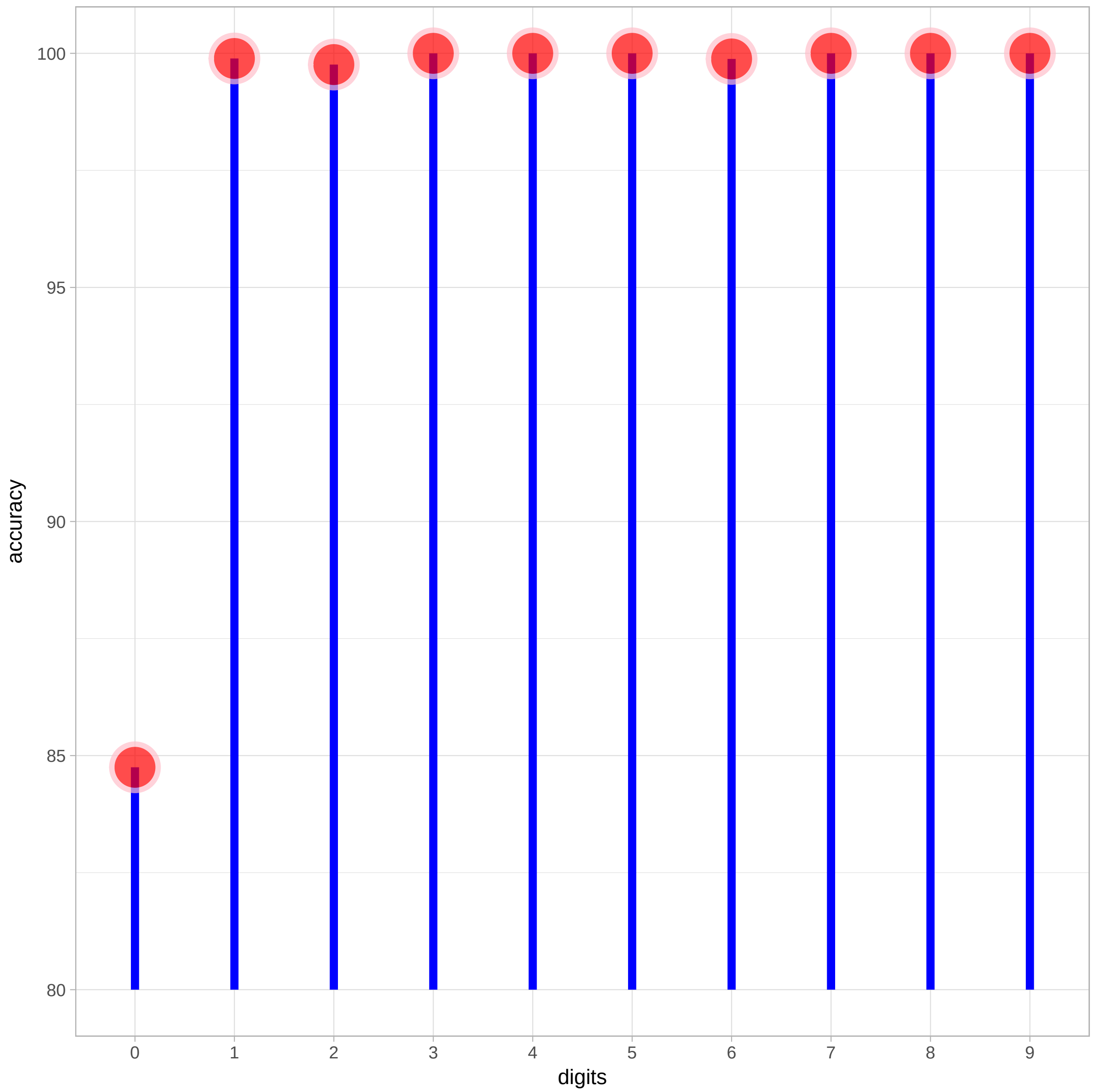
```r
require(ggplot2)
ggplot(result_df, aes(digits,accuracy, fill = digits)) + geom_bar(colour="black",stat = "identity") + co
```



The barplot shows the model struggles to identify the digit 0.

Lollipop Chart

```
require(ggplot2)
ggplot(result_df, aes(x=digits, y=accuracy, fill = digits)) +
  geom_segment( aes(x=digits, xend=digits, y=80, yend=accuracy), size=2, color="blue", linetype="solid"
  geom_point( size=10, color="pink", fill=alpha("red", 0.3), alpha=0.7, shape=21, stroke=2) +
  theme_light()
```



100
95
90
85
80
accuracy
0
1
2
3
4
5
6
7
8
9
digits