Experiment with PCA on the MNIST dataset

```
# This part read idx files and store image data into train$x and
# test$x in matrix form, store corresponding labels in train$y
# and test$y in array form
load_image_file <- function(filename) {
   ret = list()
    f = file(filename,'rb')
    readBin(f,'integer',n=1,size=4,endian='big')
    ret$n = readBin(f,'integer',n=1,size=4,endian='big')
    nrow = readBin(f,'integer',n=1,size=4,endian='big')
    ncol = readBin(f,'integer',n=1,size=4,endian='big')
    x = readBin(f,'integer',n=ret$n*nrow*ncol,size=1,signed=F)
    ret$x = matrix(x, ncol=nrow*ncol, byrow=T)
    close(f)
    ret
}

load_label_file <- function(filename) {
    f = file(filename,'rb')
    readBin(f,'integer',n=1,size=4,endian='big')
    n = readBin(f,'integer',n=1,size=4,endian='big')
    y = readBin(f,'integer',n=n,size=1,signed=F)
    close(f)
    y
}

train <- load_image_file("data/train-images-idx3-ubyte")
test <- load_image_file("data/t10k-images-idx3-ubyte")

train$y <- load_label_file("data/train-labels-idx1-ubyte")
test$y <- load_label_file("data/t10k-labels-idx1-ubyte")
```

```
train$x <- train$x/255
test$x <- test$x/255

train_df <- data.frame(train$y, train$x) %>%
  rename(label = train.y)
test_df <- data.frame(test$y, test$x) %>%
  rename(label = test.y)
```

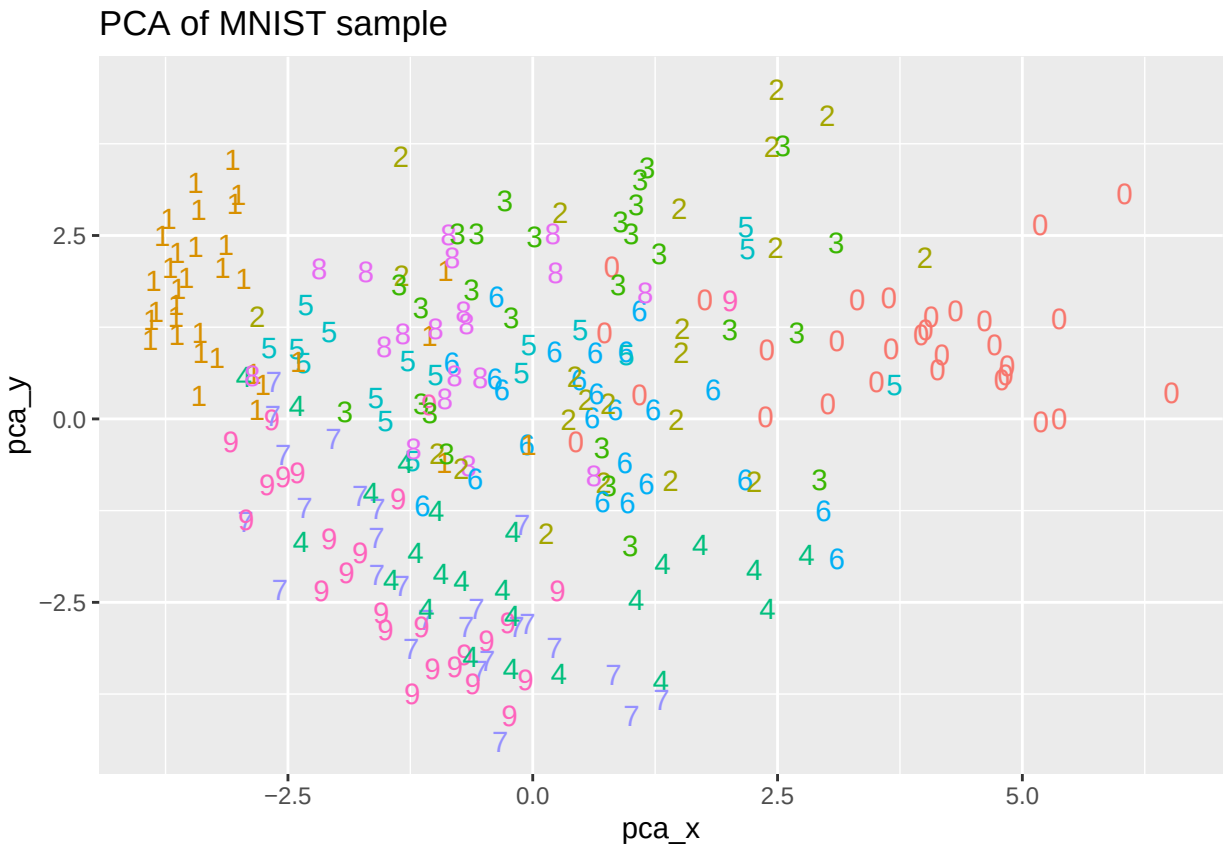
```
# Fit PCA on the training dataset
pca <- prcomp(train_df[, -1])
```

```
# Fit PCA on the test dataset
test_pca <- predict(pca, newdata = test_df)
```

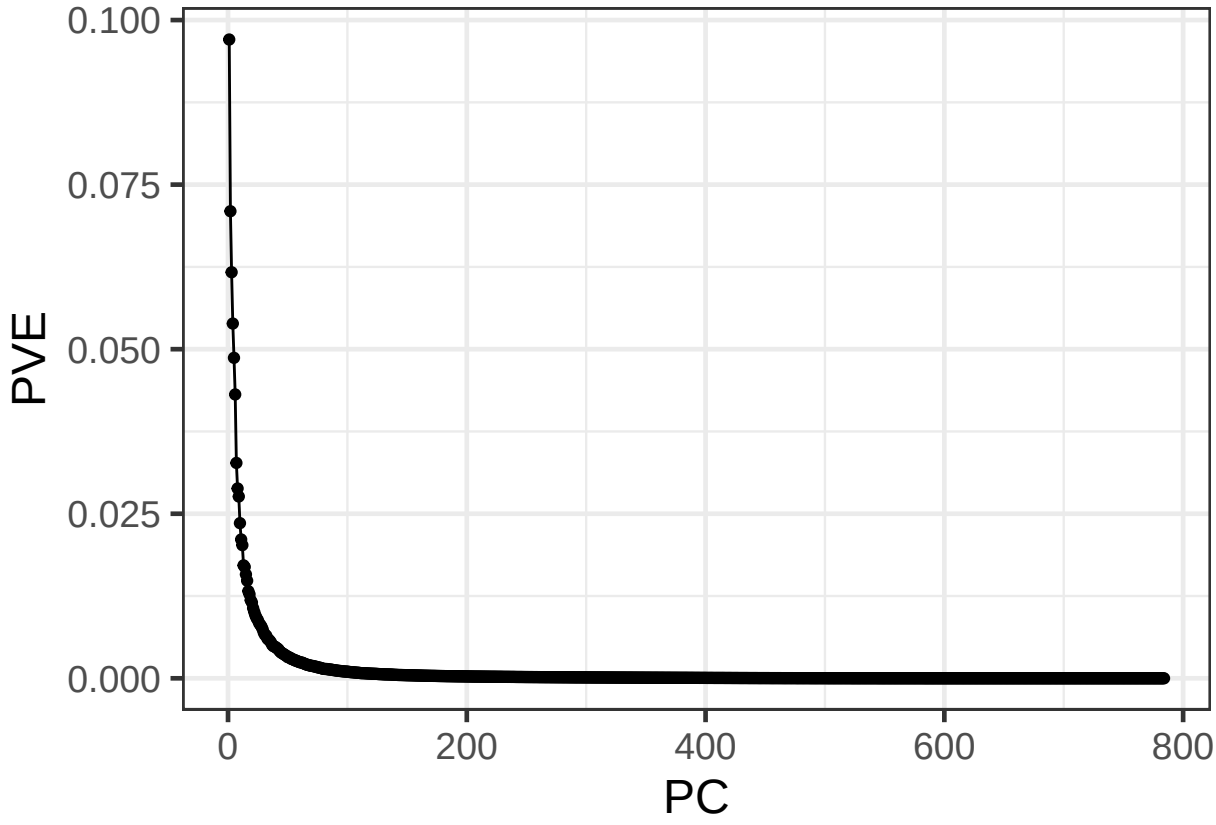
```
# Store the first two coordinates and the label in a data frame
pca_plot <- data.frame(pca_x = pca$x[, "PC1"], pca_y = pca$x[, "PC2"],
                       label = as.factor(train_df$label))
```

```
# Plot the first two principal components using the true labels as color
ggplot(pca_plot[1:250,], aes(x = pca_x, y = pca_y, color = label)) +
    ggtitle("PCA of MNIST sample") +
```

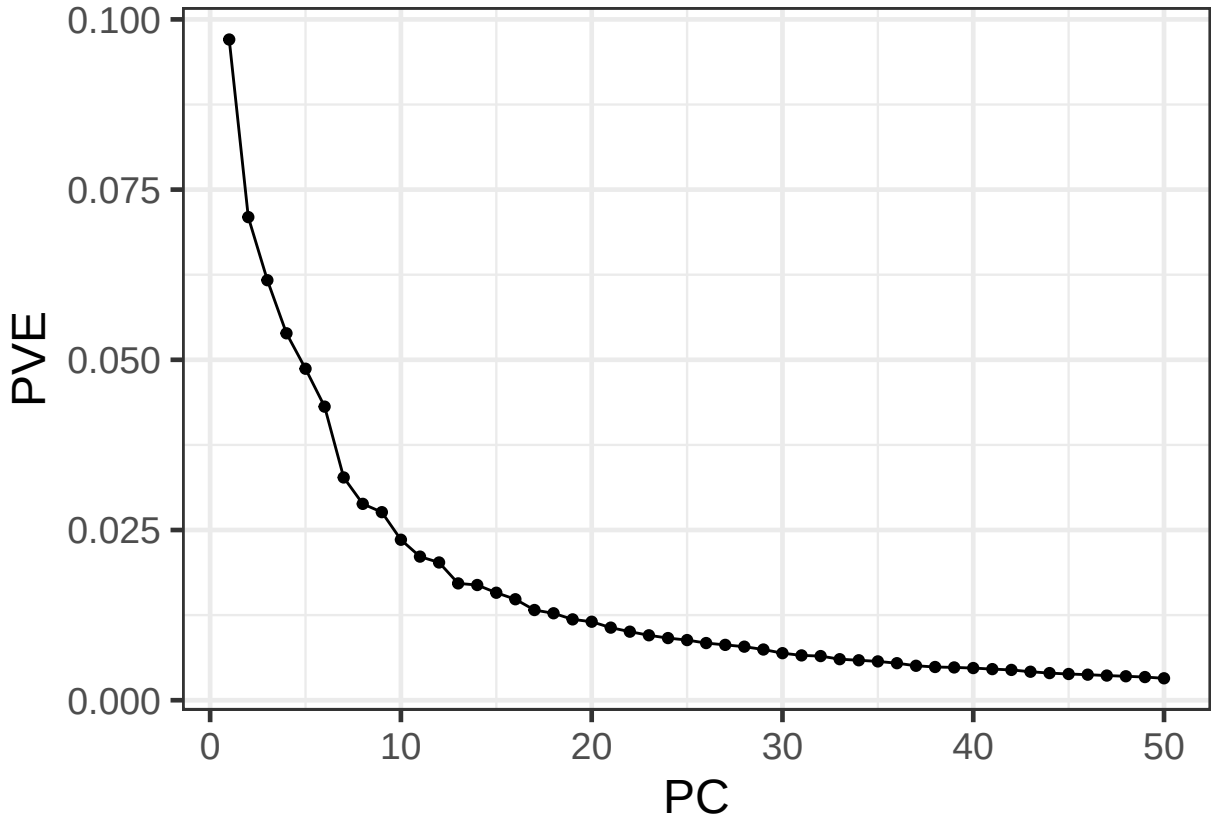
```
    geom_text(aes(label = label)) +
    theme(legend.position = "none")
```




```
d <- data.frame(PC = 1:784,
                PVE = pca$sdev^2 / sum(pca$sdev^2))
ggplot(d, aes(x = PC, y = PVE)) +
  geom_line() +
  geom_point() +
  theme_bw(base_size = 18)
```




```
ggplot(d[1:50,], aes(x = PC, y = PVE)) +
  geom_line() +
  geom_point() +
  theme_bw(base_size = 18)
```



We only need 20 PCs to capture 90% of the variance in our dataset.

```
set.seed(1)
pca <- prcomp(train_df[, -1], rank. = 20)

pca.tr <- data.frame(label = train_df[, 1], pca$x)
pca.tr$label <- as.factor(pca.tr$label)
```

```
pca.tst <- test_pca[, 1:20]  # select the first 20 PCs
pca.tst <- as.data.frame(pca.tst)

pca.tst <- pca.tst %>% mutate(label = test_df$label)
```

```
set.seed(1)

rf <- randomForest(pca.tr[, -1], pca.tr$label, ntree=500)
rf
```

```
##
## Call:
##  randomForest(x = pca.tr[, -1], y = pca.tr$label, ntree = 500)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 4
##
##         OOB estimate of  error rate: 5.04%
## Confusion matrix:
##      0    1    2    3    4    5    6    7    8    9 class.error
## 0 5805    0   18    7    8    8   47    4   17    9  0.01992234
```

```
## 1    0 6612   47   15    6   13   11   11   19    8  0.01928211
## 2   34   11 5658   54   33   11   28   49   72    8  0.05035247
## 3    7    4   75 5683    2  101   20   58  132   49  0.07307128
## 4    8   21   23    5 5530    2   40   22   22  169  0.05340637
## 5   22    6   21   89   24 5130   46    9   40   34  0.05368013
## 6   30    5   13    3   14   52 5784    0   15    2  0.02264278
## 7    2   24   67    9   31   11    0 5995   18  108  0.04309657
## 8   12   38   51  180   26  115   28   14 5329   58  0.08921552
## 9   22   13   18   93  152   28    7  112   56 5448  0.08421583
```

```
pred.rf <- predict(rf, pca.tst, type = "class")
(conf.rf <- table(pred.rf, pca.tst$label))
```

```
##
## pred.rf    0    1    2    3    4    5    6    7    8    9
##       0  963    0    9    1    0    3    8    1    6    5
##       1    0 1122    1    0    1    1    4    4    0    7
##       2    3    2  976    8    4    4    1   17    6    2
##       3    0    4   12  953    0   17    0    2   20   10
##       4    0    0    5    0  921    4    4    7    8   27
##       5    4    0    2   14    3  844    5    1   20    7
##       6    8    4    3    1   10    6  934    0    5    1
##       7    1    0    9    9    3    2    0  973    5   10
##       8    1    3   13   21    4    7    2    3  896   11
##       9    0    0    2    3   36    4    0   20    8  929
```

```
(sum(conf.rf) - sum(diag(conf.rf))) /
  sum(conf.rf)
```

```
## [1] 0.0489
```

The misclassification rate is 4.89%. The pair that is most difficult to predict are 4 and 9.

Classification Tree

```
t <- tree(label ~., data = pca.tr, split = "deviance")
summary(t)
```

```
##
## Classification tree:
## tree(formula = label ~ ., data = pca.tr, split = "deviance")
## Variables actually used in tree construction:
## [1] "PC2" "PC7" "PC4" "PC5" "PC1" "PC6" "PC8" "PC3"
## Number of terminal nodes:  13
## Residual mean deviance:  2.338 = 140200 / 59990
## Misclassification error rate: 0.3579 = 21474 / 60000
```

```
pred.tree <- predict(t, newdata = pca.tst, type = "class")
(conf.tree <- table(pred.tree, test_df$label))
```

```
##
## pred.tree    0    1    2    3    4    5    6    7    8    9
##         0  747    0   66   99    1  163   39    3   78   10
##         1    0 1055   11   11   26   13   10   53    8   30
##         2   26   24  705   26   10   42   82   20   85    4
##         3   10    4   27  601    2   52    1    0   49    5
```

```
##          4   17    0   37    9  820  112   35  101   37  572
##          5   77   41   42  164   15  388   83   18  169   18
##          6   53   11   57   37   32   33  706   13   21   30
##          7   21    0    4    5    7   18    0  592    4   38
##          8   14    0   75   47    6   67    1   30  452   11
##          9   15    0    8   11   63    4    1  198   71  291
```

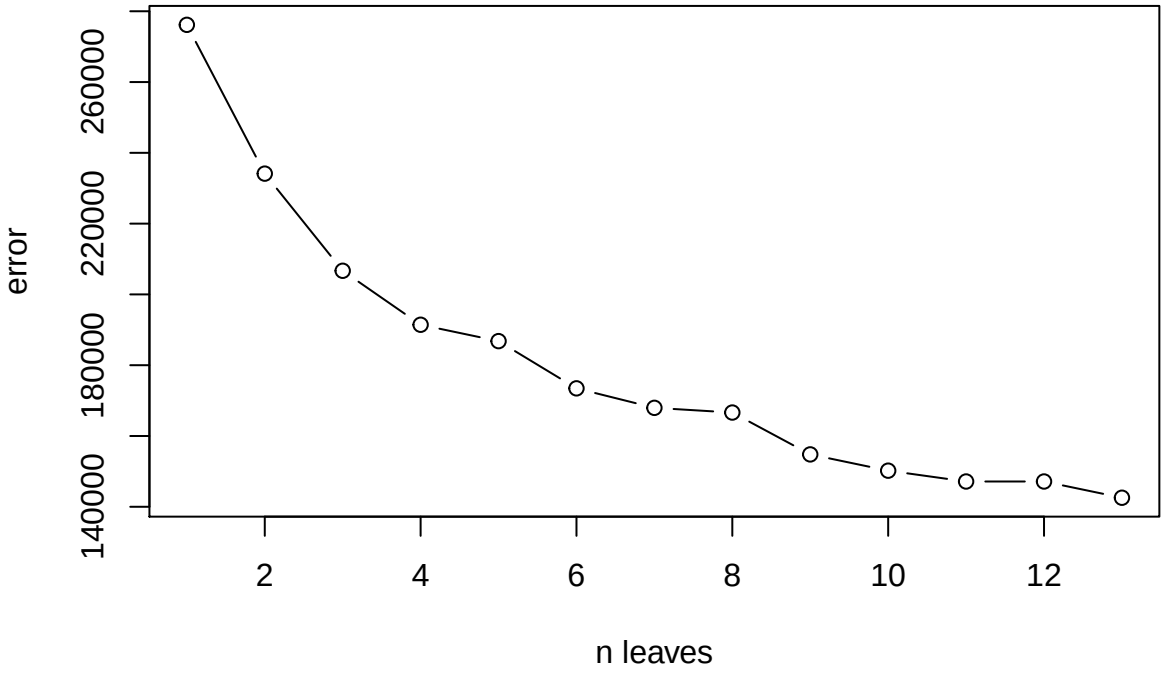
```
(sum(conf.tree) - sum(diag(conf.tree))) /
  sum(conf.tree)
```

```
## [1] 0.3643
```

4-9 is still the most difficult pair to predict, followed closely by 5-0, 7-9, 5-3, 5-8.

Pruning tree

```
t.cv <- cv.tree(t)
plot(t.cv$size, t.cv$dev, type = "b", xlab = "n leaves", ylab = "error")
```



Not a good case for pruning (best n = 13 was already chosen).

Bagging tree

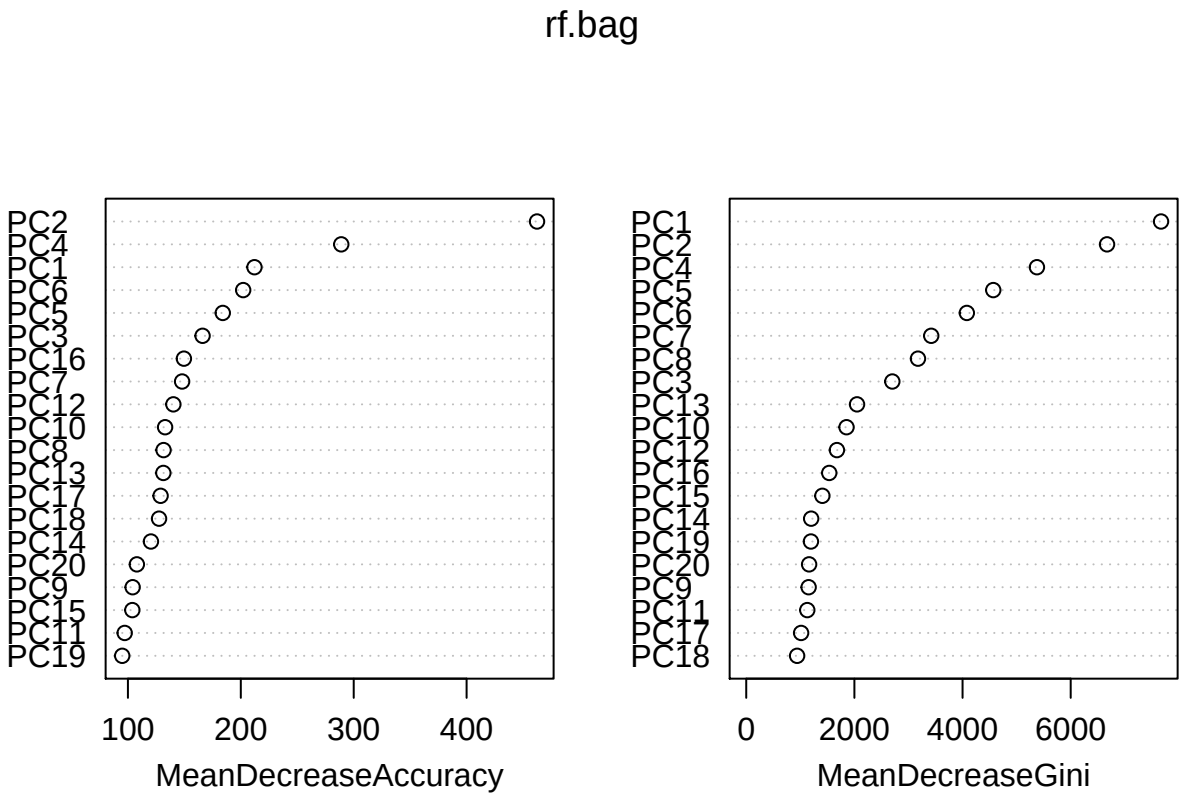
```
set.seed(1)

p <- ncol(pca.tr)-1
rf.bag <- randomForest(label ~., data = pca.tr,
                       mtry = p/3, importance = TRUE)
rf.bag
```

```
##
```

```
## Call:
##  randomForest(formula = label ~ ., data = pca.tr, mtry = p/3,      importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 7
##
##         OOB estimate of  error rate: 5.27%
## Confusion matrix:
##      0    1    2    3    4    5    6    7    8    9 class.error
## 0 5796    0   12    8   11   15   47    6   17   11  0.02144184
## 1    0 6611   38   18    7   14   14    9   21   10  0.01943044
## 2   39   10 5637   54   44    9   28   52   72   13  0.05387714
## 3   11    6   83 5656    4  109   19   60  136   47  0.07747513
## 4    7   21   32    6 5495    2   40   28   24  187  0.05939747
## 5   27    5   16   88   30 5107   45   12   48   43  0.05792289
## 6   27    6   17    2   16   54 5783    0   10    3  0.02281176
## 7    6   23   65   10   41   12    0 5970   19  119  0.04708699
## 8   13   41   52  164   28  110   26   20 5338   59  0.08767732
## 9   20   14   14   91  148   34   11  115   58 5444  0.08488822
```

```
varImpPlot(rf.bag)
```




```
pred.bag <- predict(rf.bag, newdata = pca.tst, type = "class")
(conf.bag <- table(pred.bag, pca.tst$label))
```

```
##
## pred.bag    0    1    2    3    4    5    6    7    8    9
##        0  960    0    9    1    0    3    9    1    7    4
##        1    0 1122    1    0    1    2    3    4    0    8
##        2    4    1  981    8    6    4    2   19   10    2
```

```
##         3    0    2   11  951    0   13    0    2   21    9
##         4    0    0    5    0  916    8    4    6    9   31
##         5    4    2    2   17    3  842    4    2   20    8
##         6    9    5    2    1    9    6  934    0    5    1
##         7    2    0    8    9    4    1    0  968    5   12
##         8    1    3   12   20    7    8    2    2  888   11
##         9    0    0    1    3   36    5    0   24    9  923
```

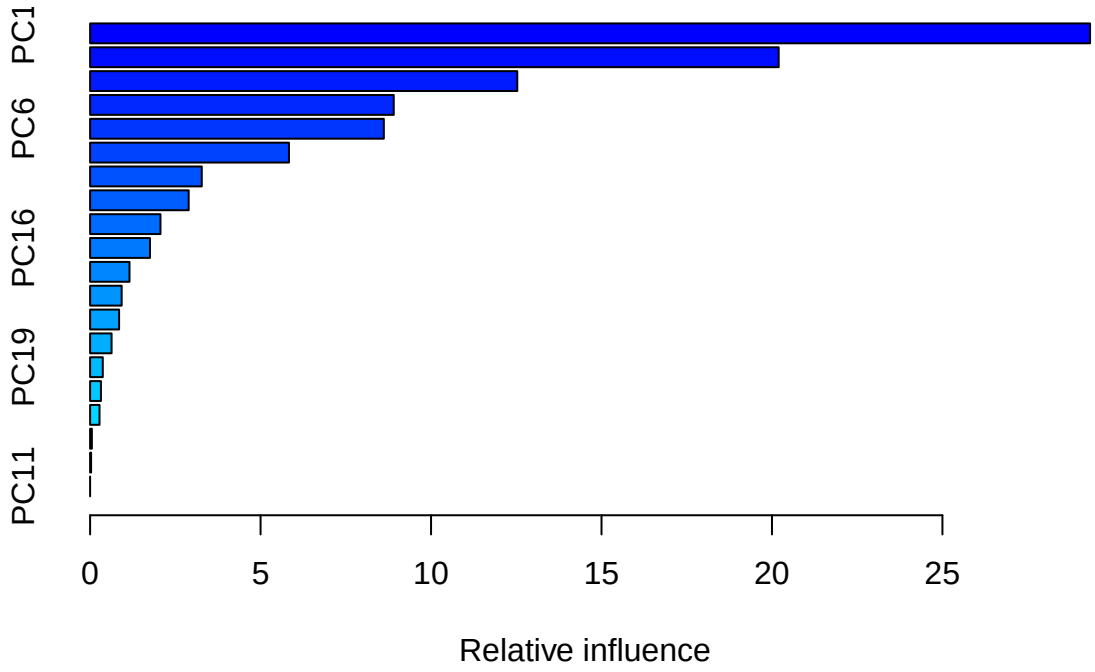
```
(sum(conf.bag) - sum(diag(conf.bag))) /
  sum(conf.bag)
```

```
## [1] 0.0515
```

The misclassification rate is 5.15%.

Boosting tree

```
set.seed(1)
boost.mnist <- gbm(label~., data = pca.tr,
                   distribution = "multinomial",
                   n.trees = 50, interaction.depth = 1,
                   shrinkage = 0.1)
summary(boost.mnist)
```




```
##       var     rel.inf
## PC1   PC1 29.32983526
## PC2   PC2 20.19729124
## PC4   PC4 12.52937052
## PC5   PC5  8.90577175
## PC6   PC6  8.61493591
## PC7   PC7  5.83248543
## PC8   PC8  3.27468586
## PC3   PC3  2.89073995
```

```
## PC13 PC13  2.06341543
## PC16 PC16  1.75585198
## PC9   PC9  1.15496116
## PC20 PC20  0.92517800
## PC12 PC12  0.85059045
## PC15 PC15  0.62954685
## PC19 PC19  0.37121670
## PC14 PC14  0.31766673
## PC10 PC10  0.27587466
## PC17 PC17  0.05185516
## PC18 PC18  0.02872696
## PC11 PC11  0.00000000
```

```
pred.boost <- predict(boost.mnist, newdata = pca.tst, n.trees = 50)
pred.boost <- apply(pred.boost, 1, which.max)

(conf.boost <- table(pred.boost, pca.tst$label))
```

```
##
## pred.boost    0    1    2    3    4    5    6    7    8    9
##          1  862    0   27   14    2   37   21    7   32   12
##          2    0 1067    3    7   21    6    4   41    3   28
##          3   17    7  781   17   16   31   53   30   25   11
##          4   11    6   41  805    1  119    7    2   72   13
##          5    2    0   15    9  721   44   10   10   10  158
##          6   42    8   14   59   14  571   50   11   44   18
##          7   24   20   49   37   45   39  786    2    6   11
##          8   10    3   20   12   21   25    9  825   14   46
##          9    3   24   72   41   22   14   10   40  745   17
##          10   9    0   10    9  119    6    8   60   23  695
```

```
(sum(conf.boost) - sum(diag(conf.boost))) /
  sum(conf.boost)
```

```
## [1] 0.2142
```

Logistic Regression

```
# create dummy variables for the digits.

pca.log.tr <- pca.tr %>%
  mutate(iszero = as.numeric(label == 0),
         isone = as.numeric(label == 1),
         istwo = as.numeric(label == 2),
         isthree = as.numeric(label == 3),
         isfour = as.numeric(label == 4),
         isfive = as.numeric(label == 5),
         issix = as.numeric(label == 6),
         isseven = as.numeric(label == 7),
         iseight = as.numeric(label == 8),
         isnine  = as.numeric(label == 9))

pca.log.tst <- pca.tst %>%
  mutate(iszero = as.numeric(label == 0),
```

```
          isone = as.numeric(label == 1),
          istwo = as.numeric(label == 2),
          isthree = as.numeric(label == 3),
          isfour = as.numeric(label == 4),
          isfive = as.numeric(label == 5),
          issix = as.numeric(label == 6),
          isseven = as.numeric(label == 7),
          iseight = as.numeric(label == 8),
          isnine  = as.numeric(label == 9))
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
ProbabilityOfEachValue <- data.frame(predict(prob.zero, test.zero),
                                     predict(prob.one, test.one),
                                     predict(prob.two, test.two),
                                     predict(prob.three, test.three),
                                     predict(prob.four, test.four),
                                     predict(prob.five, test.five),
                                     predict(prob.six, test.six),
                                     predict(prob.seven, test.seven),
                                     predict(prob.eight, test.eight),
                                     predict(prob.nine, test.nine))
```

```
# Find the index with the highest probability predicted by the models for each class and store it in a

Label <- rep(NA, nrow(ProbabilityOfEachValue))
for (i in seq(nrow(ProbabilityOfEachValue)))
{
  Label[i] <- which.max(ProbabilityOfEachValue[i,])
}
(conf.log <- table(Label, pca.tst$label))
```

```
##
## Label    0    1    2    3    4    5    6    7    8    9
##     1  948    0   14    4    3   19   18    4   16    9
##     2    0 1100   16    1    3    4    4   10   17   10
##     3    4    3  844   22    9    7   12   39   16   16
##     4    3    2   29  872    1   68    2    4   52   16
##     5    1    0   13    1  868   21   15   15   10   76
##     6   10    3    5   43   13  679   26    3   39   29
##     7    8    4   30    4   14   25  878    1   16    0
##     8    1    1   19   19    2   15    1  914    6   38
##     9    5   22   45   30   12   33    2    5  781   14
##     10   0    0   17   14   57   21    0   33   21  801
```

```
(sum(conf.log) - sum(diag(conf.log))) / sum(conf.log)
```

```
## [1] 0.1315
```

The misclassification rate is 13.15%.