Data Mining and Statistical Learning

Homework 1

Introduction

The **zipcode** data set is a well-known data set in the machine learning and data mining literature.

The original scanned digits are binary and of different sizes and orientations; the images have been **deslanted** (i.e. A pre-processing step for handwritten text recognition that involves removing the writing style) and size normalized, resulting in **16 x 16** gray-scale images (Le Cun et al., 1990). Normalized handwritten digits were automatically scanned from envelopes by the U.S. Postal Service.

The data was in two gzipped files, and each line consists of the digit id **(0-9)** followed by the **256** gray-scale values. There are **7291** training observations and **2007** test observations. One can also obtain the training data as separate files per digit (and hence without the digit identifier in each row). The test set is notoriously **"difficult"**, and a **2.5%** error rate is excellent. These data were kindly made available by the neural network group at AT&T research labs (thanks to Yann Le Cunn).

Problem Description and Data Set

For this assignment, we will be working with "zip.train.csv" as the training data set and "zip.test.csv" as the testing data. In the **zipcode** data, the first column stands for the response (Y) variable and the other columns represent the independent variables (X_i's). For more detailed description of the dataset please visit the "Element of statistical learning website" at https://hastie.su.domains/ElemStatLearn/datasets/zip.info.txt

The objective of our analysis is classify two digits: **2's and 7's**. We will be using two widely used machine learning approaches in **Linear regression** and **K-nearest Neighbors** and we will be comparing their performances.

Since our data is relatively small, Cross-Validation will be utilized to assess the robustness of each method. For that, **Monte Carlo Cross Validation** will be employed to test each model's ability to predict new data (i.e. test data)

Exploratory Data Analysis

For the purpose of this assignment, we will be working with a subset of the original data (2's and 7's). Hence, we will construct *"ziptrain27"* as the new training data and we will keep *"zip.test.csv"* as the testing data.

The new data set consists of **1376** training observations. The training data set consists of **731** observations that are labeled **"2"** and **645** observations labeled **"7"**. Each row in our data represent either a 2 or a 7.

The following two images represent the letter image of the 5-th and 99-th rows. The 5th row in the matrix represents a *7*, while the 99th row represents a *2*.

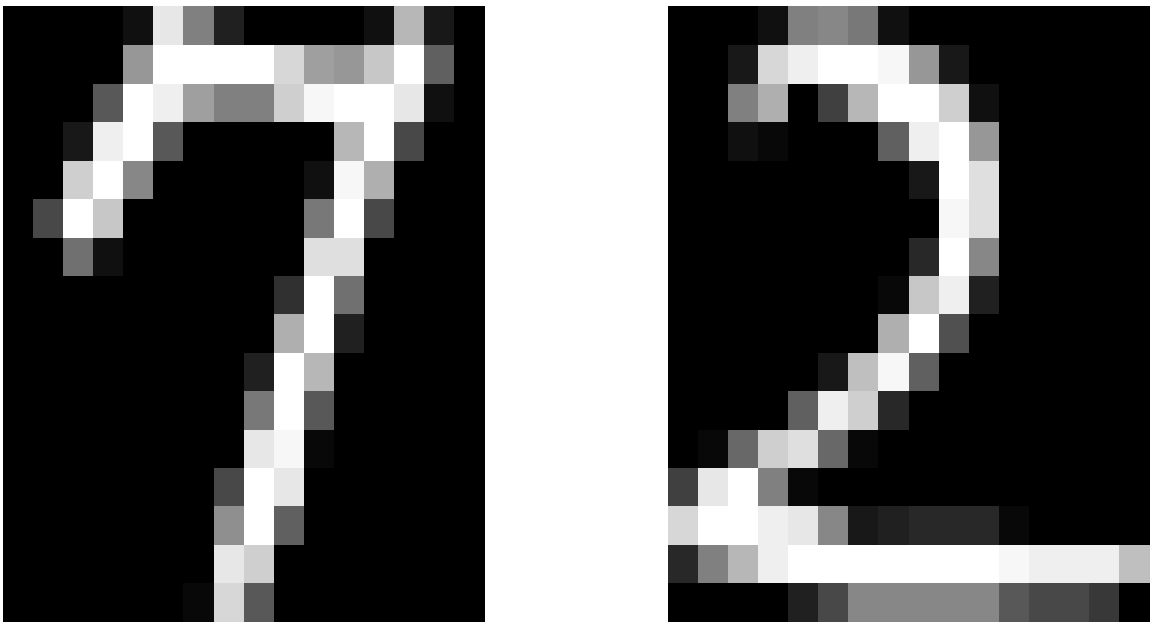


Figure 1: Images of 7 and 2

Since the data has **257** columns, it's very hard to display a summary statistics of the full training data. Therefore, a subset of the data was picked and a summary stats of **ziptrain27[5:17, 30:34]** can be seen below:

As clearly shown in the table above, each pixel value in the data set varies from $[-1, 1]$

Table 1: Summary statistics of ziptrain27[5:17, 30:34]

	V30	V31	V32	V33	V34
X	Min. :-1.0000	Min. :-1.0000	Min. :-1.0000	Min. :-1.0000	Min. :-1
X.1	1st Qu.:-0.4050	1st Qu.:-1.0000	1st Qu.:-1.0000	1st Qu.:-1.0000	1st Qu.:-1
X.2	Median : 0.5310	Median :-1.0000	Median :-1.0000	Median :-1.0000	Median :-1
X.3	Mean : 0.2335	Mean :-0.3805	Mean :-0.7785	Mean :-0.9448	Mean :-1
X.4	3rd Qu.: 0.9120	3rd Qu.: 0.4860	3rd Qu.:-0.9580	3rd Qu.:-1.0000	3rd Qu.:-1
X.5	Max. : 1.0000	Max. : 1.0000	Max. : 1.0000	Max. :-0.2830	Max. :-1

To be able gain a better understanding of the data, a correlation analysis will help empower us with valuable insights. Since our data consists of pixel intensities for each hand written digit, a correlation plot would show us if the pixels are related or not.

Below, we can see a plot of the correlation matrix of the training data. We can see luminance of the pixel that are in the interval of $[-1, 1]$ as different shades of blue. All of -1's are represented with darker shade of blue, while the brighter pixels in the plot represent 1's. As you can see, it is very hard to see pixel values for each digits (all values are on top of each other on both axis).

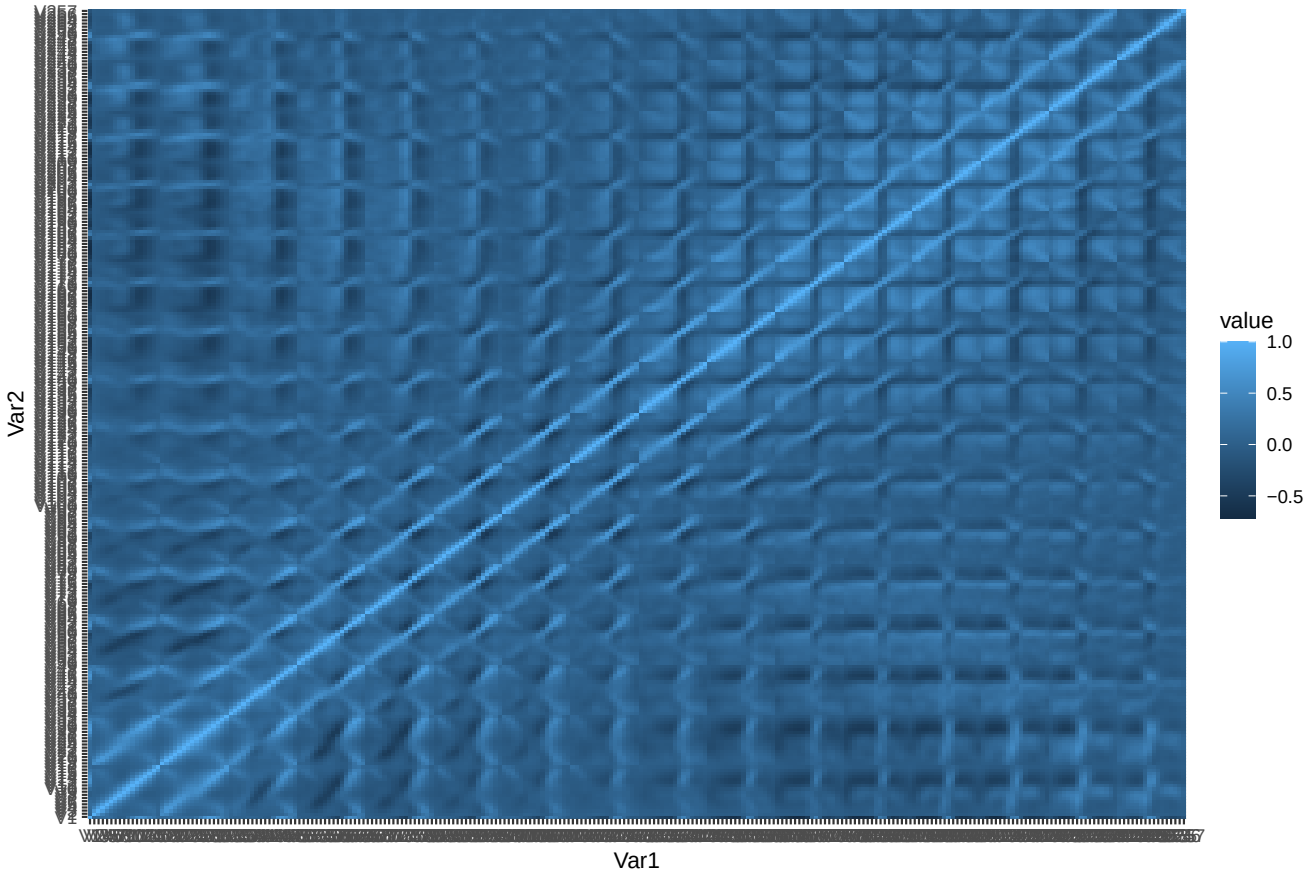


Figure 2: The correlation matrix of the training data set based on pixel intensity

For this reason, **Ranked Cross-Correlation** was mentioned on piazza as an alternative method. A Ranked Cross-Correlations not only explains relationships of a specific target feature with the rest of features but also shows the relationship of all values in the data in a nice tabular format.

The chart below shows the top 40th most relevant pairs of variables (i,e, pixels)filtered by p-values (0.05).

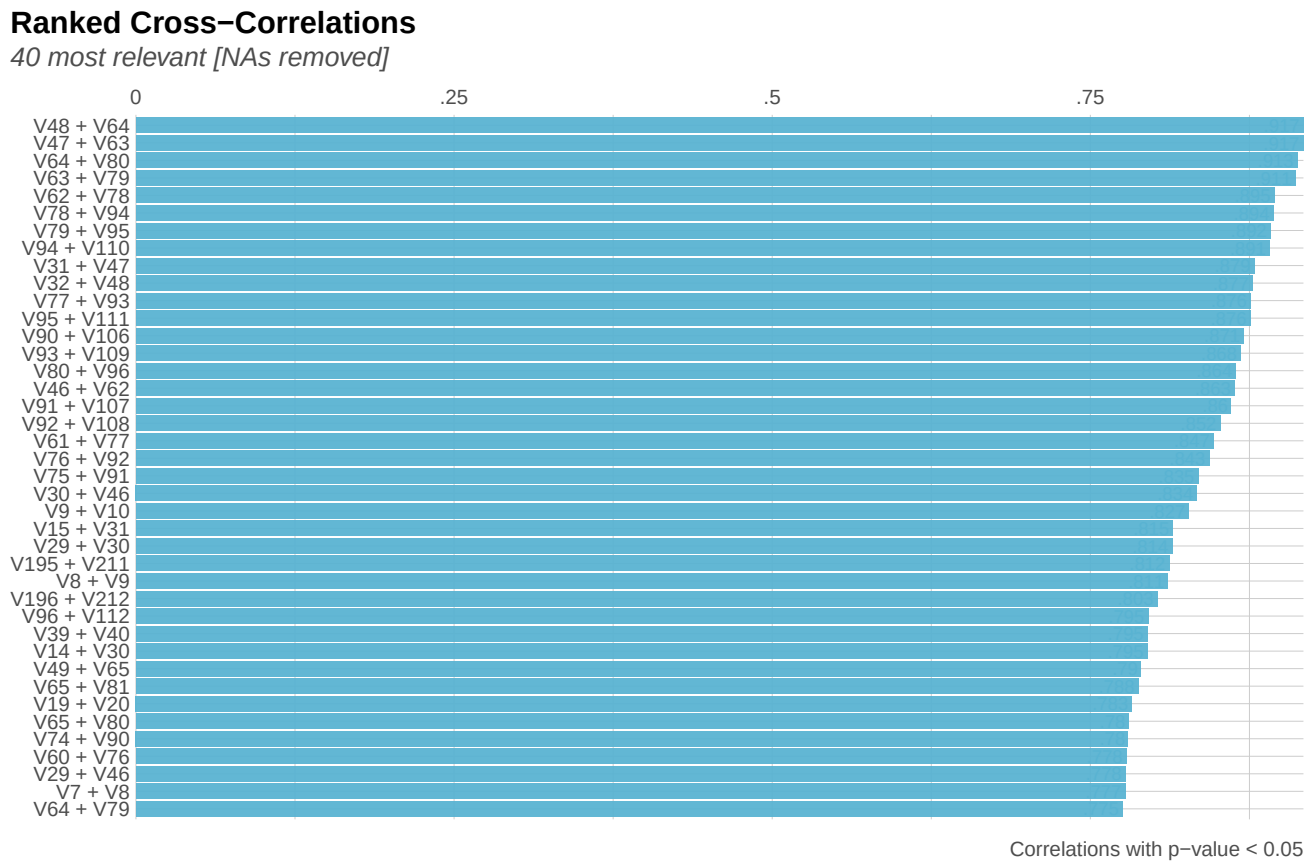


Figure 3: The Ranked cross-correlation matrix of the most relevant results filtered by p-value

The ranked cross correlation clearly shows the pairs of pixels that are closely related. This makes perfect sense since we are handling hand-written digits. We can expect that each pixel will most likely be highly correlated and closely related to the pixel adjacent to it from all directions (top, bottom, left, or right)

Methods/Methodology

As previously discussed, we will be applying and assessing the performance of two machine learning methods: **Linear regression** and **K-nearest Neighbors** with $K \in [1, 3, 5, 7, 9, 11, 13, 15]$ using a training and testing data sets as well as using Monte Carlo cross validation.

Monte Carlo Cross Validation algorithm splits the observed data points into training and testing subsets, and repeats the above computation $B = 700$ times.

In the context of this homework, we are going to combine $n_1 = 1376$ training data and $n_2 = 345$ testing data together into a larger data set. Then for each loop $b = 1, \ldots, B$ we will randomly select $n_1 = 1376$ as a new training subset and use the remaining $n_2 = 345$ data as the new testing subset. Within each loop, we first build different models from the training data of that specific loop and then evaluate their performances on the corresponding testing data.

Therefore, for each model, we will obtain B values of the testing errors on B different subsets of testing data, denote by TE_b for $b = 1, \ldots, B$. Then **the average** performances of each model will be summarized by the sample mean and sample variances of these B values

Our modeling approach is split in two parts:

1. Build a Linear Regression model and a KNN model for each $K \in [1, 3, 5, 7, 9, 11, 13, 15]$ then evaluate the training and testing errors for each model.
2. Build new models and test them on the testing data set while performing Monte Carlo Cross Validation to assess the robustness of each model.

Results

1. Using separate training and testing sets

1.1 Linear Regression After building a linear regression model and testing it on both training and testing data, we will summarize the training and testing error results in the table below.

Table 2: Linear Regression Error rate

Training Error	Testing Error
0.000727	0.017391

As expected, we see that linear regression Training Error <<<<<<< Testing Error.

1.1.1 Linear Regression model performance To assess the robustness of our linear regression model, we can calculate the model's R-squared and Adjusted r-squared (table below).

Table 3: LR model performance

Rsquared	Adjusted.Rsquared
0.942471	0.929309

R^2 here indicates that it is useful to use the independent X_i's variables to predict the Y. Note that the R^2 does not take into account how many observations are in the data or how many parameters are in the model. In practice, R^2_{adj} is widely used as it is generally better than the R^2 as a model selection criteria. In our model, both R^2 and R^2_{adj} indicate that our model is good enough to be used as a classification model for our use case.

1.2 K-nearest Neighbors After building several KNN models for $K \in [1, 3, 5, 7, 9, 11, 13, 15]$ and testing them on both training and testing data, we will summarize the training and testing error results in the table below.

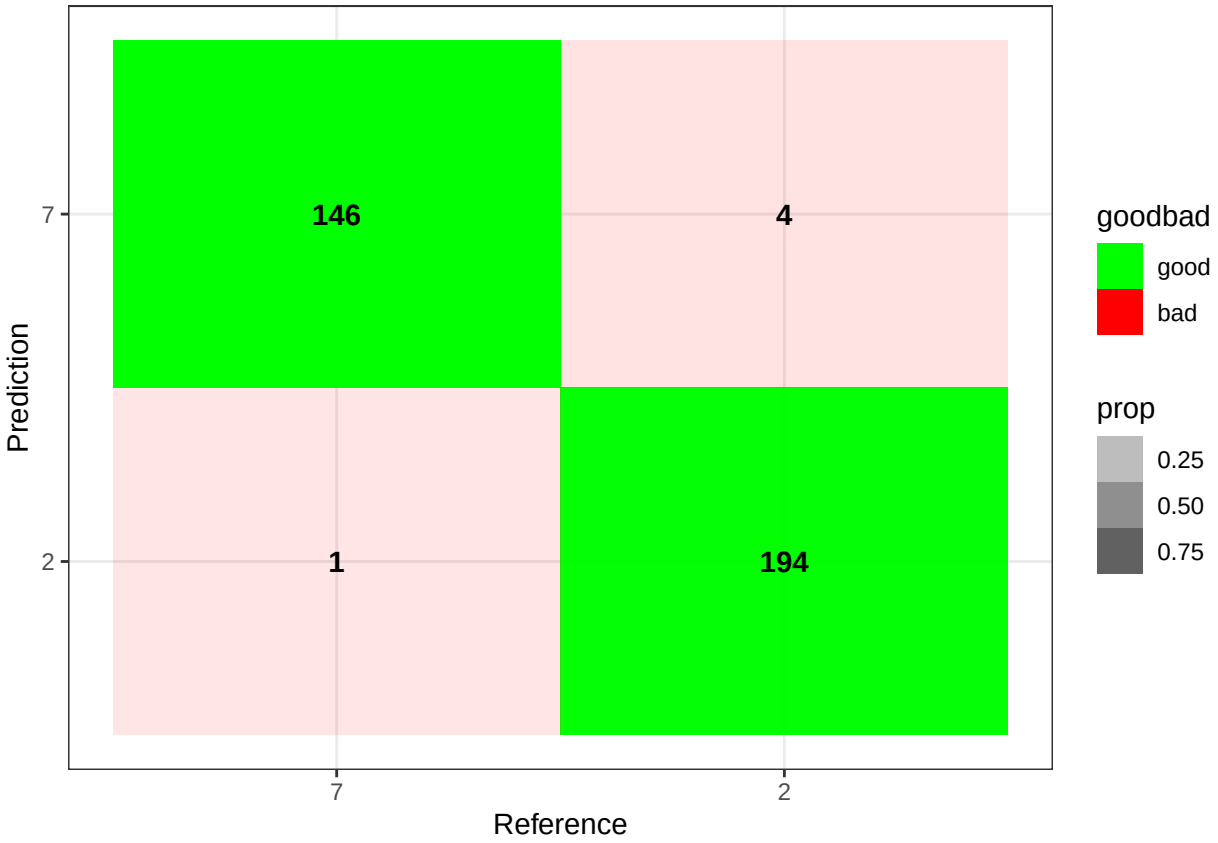
Table 4: KNN Error rate

K	Training.Error	Testing.Error
1	0.000000	0.017391
3	0.010174	0.014493
5	0.012355	0.014493
7	0.014535	0.017391
9	0.015988	0.017391
11	0.015988	0.017391
13	0.017442	0.020290
15	0.017442	0.020290
17	0.018895	0.023188
19	0.020349	0.026087

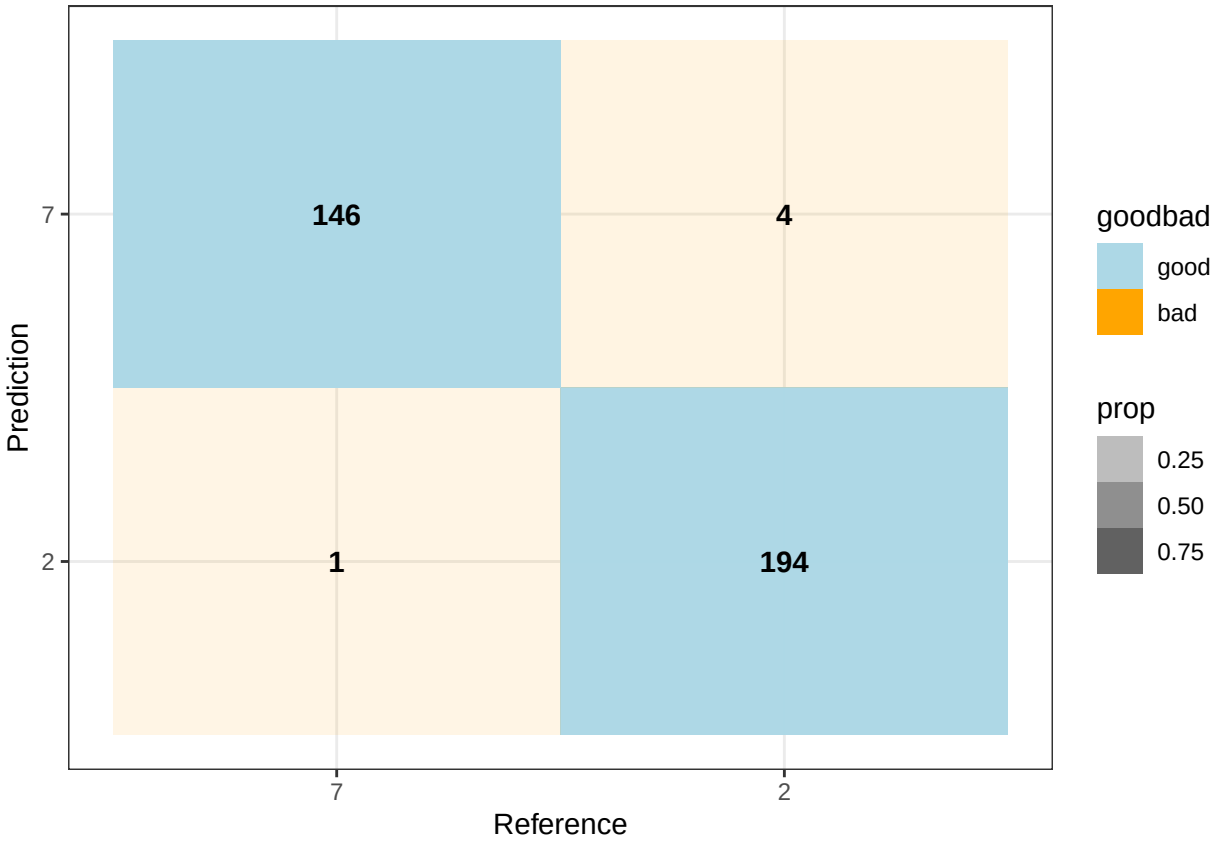
Ignoring the case when $K = 1$ (trivial case), we see that our KNN model performed best on Testing data when $K = 3$ and $K = 5$. The plot below further shows how the training/testing errors vary with respect to K.

To assess how many labels (2's and 7's) were correctly classified vs how many were misclassified for $K = 3$ and $K = 5$, we can use a confusion matrix. The confusion matrix is widely utilized for the performance evaluations of classification methods.

1.2.1 K-nearest neighbors best model performance ($K = 3$)



1.2.2 K-nearest neighbors best model performance ($K = 5$)



From the two confusion matrices, we can see that both KNN models misclassified four 2's as 7 and one 7 as a 2. Both models performed well given that some of the hand-written digits are very hard to decipher. I personally had a very hard time figuring out if a row image in my exploratory analysis was a 2 or a 7.

2. Using Monte Carlo Cross Validation

In this section, we will summarize the results of Monte Carlo Cross Validation using $B = 700$

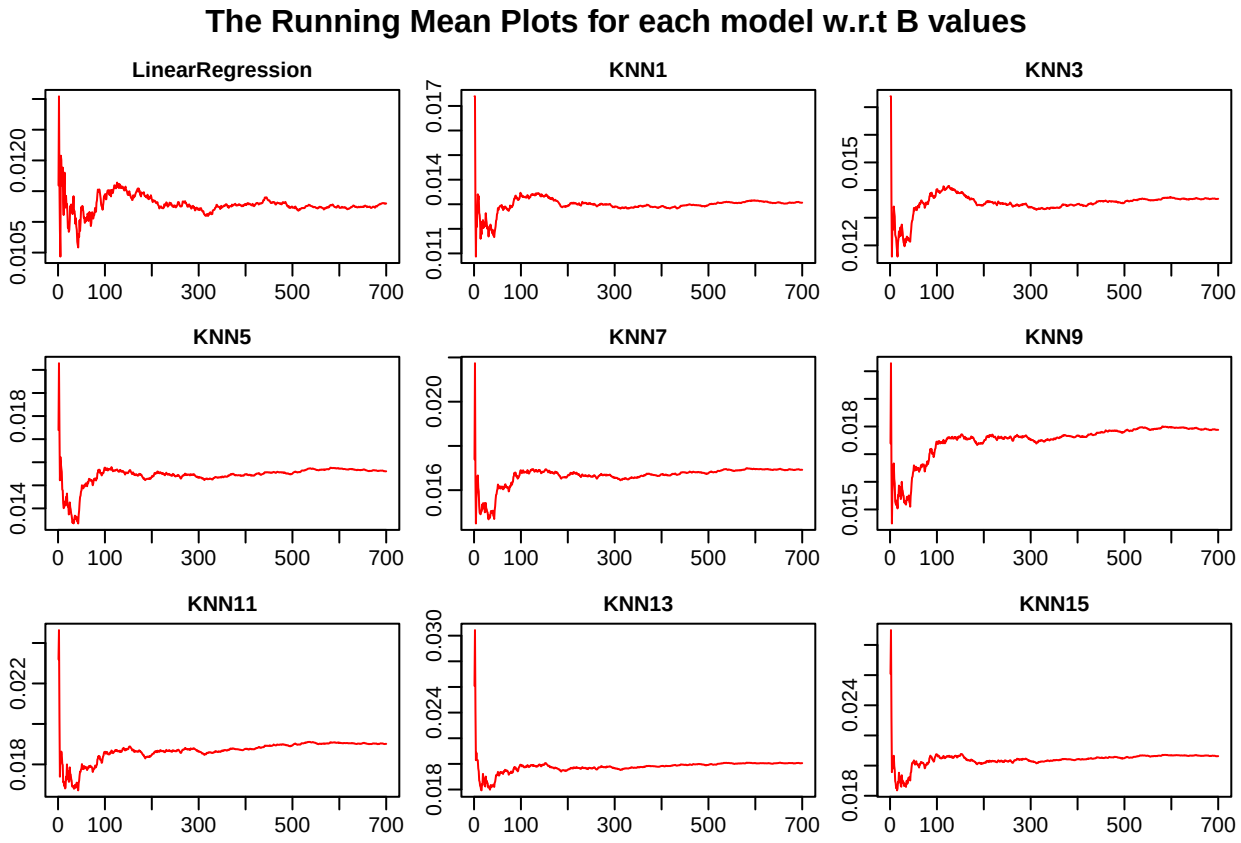
The average performances of each model can be summarized by the sample mean and sample variances of these B values.

Table 5: Sample mean/variance of MCCV Errors for B = 700

	LinearRegression	KNN1	KNN3	KNN5	KNN7	KNN9	KNN11	KNN13	KNN15
Mean	0.01130	0.01306	0.01369	0.01561	0.01692	0.01788	0.01901	0.02006	0.02063
Variance	0.00003	0.00003	0.00003	0.00004	0.00004	0.00004	0.00004	0.00005	0.00005

We can clearly see that both Linear regression model and KNN with $K = 3$ had the same sample variance of testing error, while linear regression had a somewhat smaller mean testing error. To be able to see how the testing error varies with B values, we can take a look at the running mean plots from this Monte Carlo simulation on a single plot for all the models.

From the running mean plots, it is evident that each model's testing error is converging to its expected prediction error

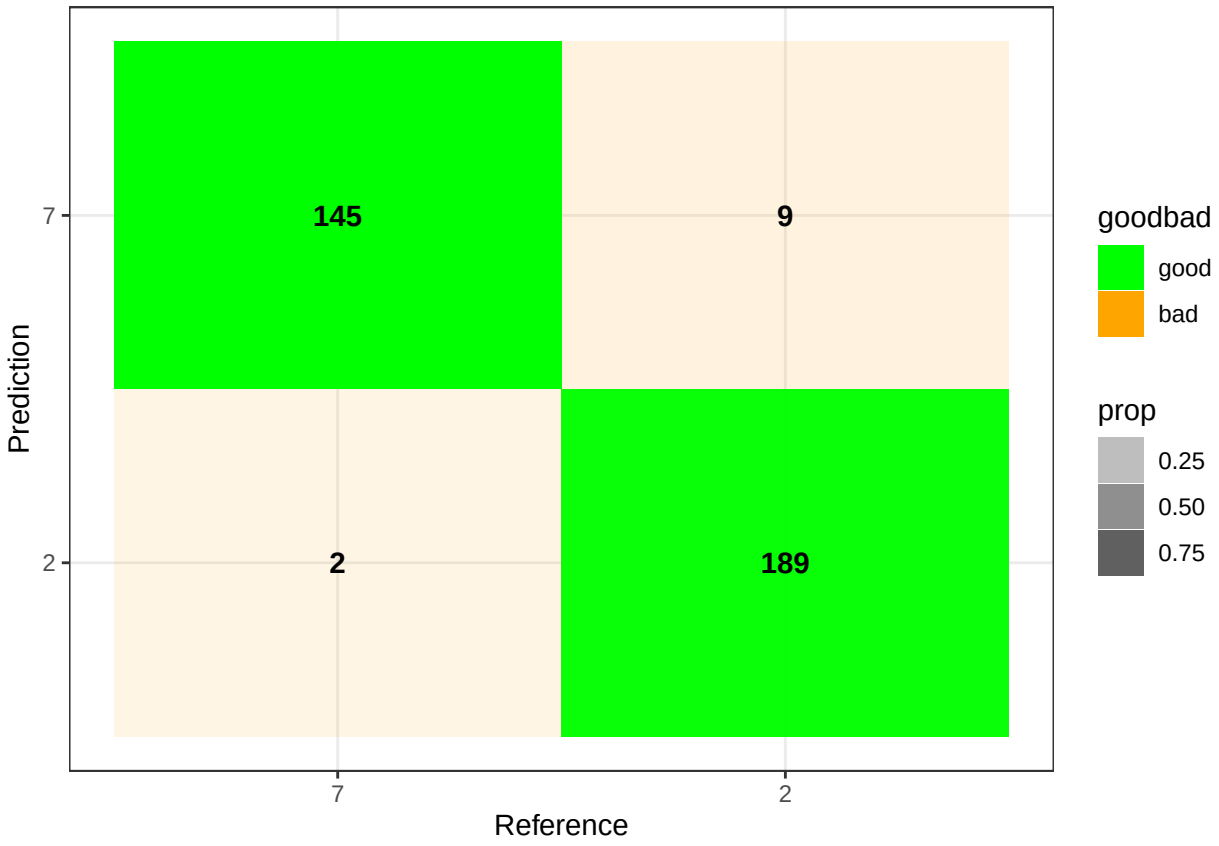


Conclusion

Using Monte Carlo Cross Validation was crucial to fully understand the robustness of each model used in this assignment. From looking at the sample mean and variance of the testing errors, we can say that linear regression and k-nearest neighbors with $K = 3$ are the "best" models. However, a smaller K value can have a higher influence on the result. Additionally, a $K = 3$ is not useful for our use case. In practice, the optimal value of K is $K = \sqrt{n}$ where n stands for the number of samples in the training dataset.

As an experiment, we can fit a knn model with $K = \sqrt{n} = \sqrt{1376} \approx 37$ where **1376** is the number of training observations in our data. We can visually inspect the model's performance in the confusion matrix below. A majority vote between the $K = 37$ most similar pixels to a given "unseen" observation seems like a robust approach

in our case.



References

1. Monte Carlo cross validation:https://stats.stackexchange.com/questions/51416/k-fold-vs-monte-carlo-cross-validation
2. kable table documentation: https://bookdown.org/yihui/rmarkdown-cookbook/kable.html
3. rmeanplot tutorial: https://www.rpubs.com/Benjamin_Chan_Chun_Ho/Full_Model_12
4. Confusion Matrix: https://stackoverflow.com/questions/37897252/plot-confusion-matrix-in-r-using-ggplot
5. Confusion Matrix: https://www.sciencedirect.com/topics/engineering/confusion-matrix
6. Ranked Cross correlation: https://laresbernardo.github.io/lares/reference/corr_cross.html

Appendix

R Code

```
# Settings
knitr::opts_chunk$set(echo = FALSE, message = FALSE, warning = FALSE, fig.pos = 'H')


# Check if packages are installed. If not, install them.
install.packages <- function(pkg) {
    new.pkg <- pkg[!(pkg %in% installed.packages()[, "Package"])]
    if (length(new.pkg)) {
        install.packages(new.pkg)
    }
}


packages <- c("ggplot2", "mcmcplots", "knitr","xtable",
              "kableExtra", "reshape2", "devtools",
              "lares", "caret", "splines", "class", "dplyr")


install.packages(packages)
library(mcmcplots)
library(knitr)
library(kableExtra)
library(ggplot2)
library(reshape2)
library(devtools)
library(lares)
library(caret)
library(splines)
library(class)
library(extrafont)
library(xtable)
library(dplyr)
options(kableExtra.latex.load_packages = FALSE)
```

```r
loadfonts()


devtools::install_github("haozhu233/kableExtra")
# reading data and checking its dimension
ziptrain <- read.table(file="zip.train.csv", sep = ",")
ziptest <- read.table(file="zip.test.csv", sep = ",")
ziptrain27 <- subset(ziptrain, ziptrain[,1]==2 | ziptrain[,1]==7)
ziptest27 <- subset(ziptest, ziptest[,1]==2 | ziptest[,1]==7)
dim(ziptrain)# dimension of the original training data
dim(ziptest) # dimension of the test data
dim(ziptrain27) # dimension of the subset data
sum(ziptrain27[,1] == 2) # sum of all 2's
sum(ziptrain27[,1] == 7) # sum of all 7's


# display images
rowindex = 5; ## You can try other "rowindex" values to see other rows
Xval1 = t(matrix(data.matrix(ziptrain27[,-1])[rowindex,],byrow=TRUE,16,16)[16:1,]);
rowindex_ = 99; ## You can try other "rowindex" values to see other rows
Xval2 = t(matrix(data.matrix(ziptrain27[,-1])[rowindex_,],byrow=TRUE,16,16)[16:1,]);
par(mfrow=c(1,2))    # set the plotting area into a 1*2 array
image(Xval1,col=gray(0:32/32),axes=FALSE) ## Also try "col=gray(0:32/32)"
image(Xval2,col=gray(0:32/32),axes=FALSE)
subset_ziptrain27 = ziptrain27[5:17, 30:34] #


data_summary <- data.frame(unclass(summary(subset_ziptrain27)),  # Convert summary to data frame
                           check.names = FALSE)
data_summary %>%
  kable(caption = "Summary statistics of ziptrain27[5:17, 30:34]",
        digits = 3, align = "l") %>%kable_styling()
ziptrain27Cor <- round(cor(ziptrain27),2)%>% melt() # get correlation matrix
# plot the correlation matrix
ggplot(data = ziptrain27Cor, aes(x=Var1, y=Var2, fill=value))+ geom_tile()
# Show only most relevant results filtered by p-value
corr_cross(ziptrain27, rm.na = TRUE, max_pvalue = 0.05, top = 40)
```

```r
# regression on training data
mod1 <- lm( V1 ~ . , data= ziptrain27);
pred.train <- predict.lm(mod1, ziptrain27[,-1]);
y1pred.train <- 2 + 5*(pred.train >= 4.5);
trainngerror_lr = mean( y1pred.train != ziptrain27[,1])

# predict on testing set
pred.test <- predict.lm(mod1, ziptest27[,-1]);
ypred.test <- 2 + 5*(pred.test >= 4.5);
testingerror_lr = mean( ypred.test != ziptest27[,1])



table_lm <- data.frame(
  Dataset = c('Training Data'),
  Mean.Training.Error = trainngerror_lr
)
table2_lm <- data.frame(
  Dataset = c('Testing Data'),
  Mean.Testing.Error =  testingerror_lr
)

error_data_lm = cbind(trainngerror_lr,testingerror_lr)
colnames(error_data_lm) <- c( "Training Error", "Testing Error")
knitr::kable(error_data_lm,"pipe", digit=6,
             caption = "Linear Regression Error rate")

# Get the R^2
model_params <- data.frame(
  Rsquared =  summary(mod1)$r.squared,
  Adjusted.Rsquared = summary(mod1)$adj.r.squared
)

knitr::kable(model_params,"pipe", digit=6,caption = "LR model performance")
# %>%kable_styling(position = "center", full_width = F)
```

```r
# KNN Training Error
x_new <- ziptrain27[,-1]
z_train <- c()
k.vals = seq(1, 20, 2)
for (i in k.vals) {
  ypred.train <- knn(train=ziptrain27[,-1], test=x_new, cl = ziptrain27[,1], k = i)
  z_train <- c(z_train, mean(ypred.train != ziptrain27[,1]))
  }
train_error_tbl <- data.frame(
  K =  k.vals,
  Training.Error = z_train
)

# KNN Testing Error
x_new2 <- ziptest27[,-1]
z_test <- c()
for (i in k.vals) {
  ypred2.test <- knn(train=ziptrain27[,-1], test=x_new2, cl = ziptrain27[,1], k = i)
  z_test <- c(z_test, mean(ypred2.test != ziptest27[,1]))
  }
test_error_tbl <- data.frame(
  K = k.vals,
  Testing.Error = z_test
)

error_data = cbind(train_error_tbl,test_error_tbl)
tb_data = error_data[, c(1, 2, 4)]
knitr::kable(tb_data,"pipe", digit=6,caption = "KNN Error rate")
# %>%kable_styling(position = "center", full_width = F)
# %>%
#   row_spec(2, bold = T, color = "white", background = "green")
# plot
error_data = cbind(train_error_tbl,test_error_tbl)
```

```
d <- melt(error_data, id.vars="K")


# Everything on the same plot
par(cex.main=1)
ggplot(d, aes(K,value, col=variable)) +
  geom_line() +
  geom_point(size=3) +
  theme_classic() +
  ggtitle("Training vs. Testing Errors") +
  theme(plot.title = element_text(hjust = 0.5, size=12, face='bold'))+
  xlab('K') +
  ylab('Error')
# plot the confusion matrix
train_labels = ziptrain27[,1]
test_labels = ziptest27[,-1]
# dim(train_labels)
# dim(test_labels)
knn.mod.1 <- knn(train=ziptrain27[,-1], test=x_new2, cl =train_labels, k = 3)


# cm = confusionMatrix(as.factor(knn.mod) ,as.factor(ziptest27[,1]), dnn = c("Prediction", "Reference"))


table <- data.frame(confusionMatrix(as.factor(knn.mod.1) ,as.factor(ziptest27[,1]),
                                    dnn = c("Prediction", "Reference"))$table)


plotTable <- table %>%
  mutate(goodbad = ifelse(table$Prediction == table$Reference, "good", "bad")) %>%
  group_by(Reference) %>%
  mutate(prop = Freq/sum(Freq))


ggplot(data = plotTable, mapping = aes(x = Reference,
                                       y = Prediction, fill = goodbad, alpha = prop)) +
  geom_tile() +
  geom_text(aes(label = Freq), vjust = .5, fontface  = "bold", alpha = 1) +
  scale_fill_manual(values = c(good = "green", bad = "red")) +
```

```
  theme_bw() +
  xlim(rev(levels(table$Reference)))



knn.mod.2 <- knn(train=ziptrain27[,-1], test=x_new2, cl =train_labels, k = 5)

tbl <- data.frame(confusionMatrix(as.factor(knn.mod.2) ,as.factor(ziptest27[,1]),
                                  dnn = c("Prediction", "Reference"))$table)

plotTable <- tbl %>%
  mutate(goodbad = ifelse(tbl$Prediction == tbl$Reference, "good", "bad")) %>%
  group_by(Reference) %>%
  mutate(prop = Freq/sum(Freq))

ggplot(data = plotTable, mapping = aes(x = Reference,
                                       y = Prediction, fill = goodbad, alpha = prop)) +
  geom_tile() +
  geom_text(aes(label = Freq), vjust = .5, fontface  = "bold", alpha = 1) +
  scale_fill_manual(values = c(good = "lightblue", bad = "orange")) +
  theme_bw() +
  xlim(rev(levels(table$Reference)))

zip27full = rbind(ziptrain27, ziptest27) ### combine to a full data set
n1 = 1376; # training set sample size
n2= 345; # testing set sample size
n = dim(zip27full)[1]; ## the total sample size

### Initialize the TE values for all models in all $B=100$ loops
set.seed(7406); ### set the seed for randomization
B = 700; ### number of loops
TEALL = NULL; ### Final TE values

for (b in 1:B){
```

```
  ### randomly select n1 observations as a new training subset in each loop
  flag <- sort(sample(1:n, n1));
  zip27traintemp <- zip27full[flag,];
  zip27testtemp <- zip27full[-flag,];
  xnew2 <- zip27testtemp[,-1];


  # linear regression
  model <- lm( V1 ~ . , data= zip27traintemp)
  pred.test <- predict.lm(model, zip27testtemp[,-1]);
  ypred.test <- 2 + 5*(pred.test >= 4.5);
  TEAL = mean( ypred.test != zip27testtemp[,1])

  # KNN
  for (i in 0:7){
    k.i = 2*i + 1
    ypred <- knn(zip27traintemp[,-1], xnew2, zip27traintemp[,1], k=k.i);
    error_knn_test <- mean( ypred != zip27testtemp[,1]);
    TEAL <- cbind(TEAL, error_knn_test)
   }
  TEALL = rbind(TEALL,TEAL);
}
colnames(TEALL) <- c( "LinearRegression","KNN1", "KNN3", "KNN5",
                      "KNN7","KNN9", "KNN11", "KNN13", "KNN15")
Mean = apply(TEALL, 2, mean);
Variance  = apply(TEALL, 2, var);

data = rbind(Mean, Variance)


# displaying the tables
knitr::kable(data,"pipe", digit=5,
             caption = paste("Sample mean/variance of MCCV Errors for B = ", B))%>%
          kable_styling(position = "center", full_width = F)
```

```
      # %>%
      #     column_spec(3, bold = T, color = "white", background = "green")%>%
      #     column_spec(2, bold = T, color = "white", background = "green")
par(cex.main=1)
rmeanplot(TEALL, style="plain", col="red",
          plot.title = "The Running Mean Plots for each model w.r.t B values")
# fitting KNN with k=37
knn.mod.optimal <- knn(train=ziptrain27[,-1], test=x_new2, cl =train_labels, k = 37)


tbl <- data.frame(confusionMatrix(as.factor(knn.mod.optimal) ,as.factor(ziptest27[,1]),
                                  dnn = c("Prediction", "Reference"))$table)


plotTable <- tbl %>%
  mutate(goodbad = ifelse(tbl$Prediction == tbl$Reference, "good", "bad")) %>%
  group_by(Reference) %>%
  mutate(prop = Freq/sum(Freq))


ggplot(data = plotTable, mapping = aes(x = Reference,
                                       y = Prediction, fill = goodbad, alpha = prop)) +
  geom_tile() +
  geom_text(aes(label = Freq), vjust = .5, fontface  = "bold", alpha = 1) +
  scale_fill_manual(values = c(good = "green", bad = "orange")) +
  theme_bw() +
  xlim(rev(levels(table$Reference)))
```