HW4pt2

David Li

March 20, 2018

Appendix: R Code

Note that Some Labels and scales were tuned for the final draft's output

Credit: Based off very loosely of Roger Fan's online code BUT not identical!

```
## Data Formation & Cleaning ##

# Import the datasets first
spamtrain = read.csv("C:/Users/David/Desktop/School/Stats503/Datasets/spam-data/spam-train.txt", header=
spamtest = read.csv("C:/Users/David/Desktop/School/Stats503/Datasets/spam-data/spam-test.txt", header=F
spamnames = c("word_freq_make", "word_freq_address","word_freq_all","word_freq_3d",
"word_freq_our","word_freq_over", "word_freq_remove",
"word_freq_internet", "word_freq_order","word_freq_mail",
"word_freq_receive","word_freq_will", "word_freq_people",
"word_freq_report", "word_freq_addresses","word_freq_free",
"word_freq_business", "word_freq_email","word_freq_you",
"word_freq_credit", "word_freq_your", "word_freq_font",
"word_freq_000","word_freq_money","word_freq_hp", "word_freq_hpl",
"word_freq_george", "word_freq_650","word_freq_lab",
"word_freq_labs", "word_freq_telnet", "word_freq_857","word_freq_data",
"word_freq_415","word_freq_85", "word_freq_technology",
"word_freq_1999", "word_freq_parts","word_freq_pm",
"word_freq_direct", "word_freq_cs", "word_freq_meeting",
"word_freq_original","word_freq_project","word_freq_re",
"word_freq_edu","word_freq_table","word_freq_conference",
"char_freq_semicolon","char_freq_parenthesis","char_freq_bracket",
"char_freq_exclamation","char_freq_dollar", "char_freq_pound",
"capital_run_length_average", "capital_run_length_longest", "capital_run_length_total", "class")
colnames(spamtrain) = spamnames
colnames(spamtest) = spamnames

# Standardized form of datasets
standspamtrain = spamtrain
standspamtrain[,1:57] = scale(standspamtrain[,1:57])

standspamtest = spamtest
standspamtest[,1:57] = scale(standspamtest[,1:57])

# Packages for Use
library("dplyr")
library("gridBase")
library("gridExtra")
library("knitr")
library("ggplot2")
library("nnet")
```

```
library("class")
library("sparsediscrim")
library("reshape2")
library("e1071")
library("reshape2")
library("rpart")
library("rpart.plot")

### SVM CODE SECTION ###

# Original Data, linear kernel
folds = cv_partition(spamtrain$class, num_folds = 5)
costs = c(exp(-5), exp(-4), exp(-3), exp(-1), exp(1), exp(3), exp(5), exp(7))

train_cv_error_svm = function(costC) {
  #Train
  spam.svm = svm(class~., data=spamtrain,
                 kernel="linear", cost=costC)
  train_error = sum(spam.svm$fitted != spamtrain$class) / nrow(spamtrain)
  #Test
  test_error = sum(predict(spam.svm, spamtest) != spamtest$class) / nrow(spamtest)
  #CV error
  spame.cverr = sapply(folds, function(fold) {
    svmcv = svm(class~.,data = spamtrain, kernel="linear", cost=costC, subset = fold$training)
    svmpred = predict(svmcv, spamtrain[fold$test,])
    return(sum(svmpred != spamtrain$class[fold$test]) / length(fold$test))
  })
  cv_error = mean(spame.cverr)
  return(c(train_error, cv_error, test_error))
}

spam_cost_errors = sapply(costs, function(cost) train_cv_error_svm(cost))
df_errs = data.frame(t(spam_cost_errors), costs)
colnames(df_errs) = c('Train', 'CV', 'Test', 'Logcost')

dataL <- melt(df_errs, id="Logcost")
ggplot(dataL, aes_string(x="Logcost", y="value", colour="variable",
group="variable", linetype="variable", shape="variable")) +
geom_line(size=1) + labs(x = "Cost",
y = "Classification error",
colour="",group="",
linetype="",shape="") + scale_x_log10()

# Original Data, poly2 kernel

train_cv_error_svm = function(costC) {
  #Train
  spam.svm = svm(class~., data=spamtrain,
                 kernel="polynomial", degree = 2, cost=costC)
  train_error = sum(spam.svm$fitted != spamtrain$class) / nrow(spamtrain)
  #Test
  test_error = sum(predict(spam.svm, spamtest) != spamtest$class) / nrow(spamtest)
  #CV error
```

```
  spame.cverr = sapply(folds, function(fold) {
    svmcv = svm(class~.,data = spamtrain, kernel="polynomial", degree = 2, cost=costC, subset = fold$tra
    svmpred = predict(svmcv, spamtrain[fold$test,])
    return(sum(svmpred != spamtrain$class[fold$test]) / length(fold$test))
  })
  cv_error = mean(spame.cverr)
  return(c(train_error, cv_error, test_error))
}

spam_cost_errors = sapply(costs, function(cost) train_cv_error_svm(cost))
df_errs = data.frame(t(spam_cost_errors), costs)
colnames(df_errs) = c('Train', 'CV', 'Test', 'Logcost')

dataL <- melt(df_errs, id="Logcost")
ggplot(dataL, aes_string(x="Logcost", y="value", colour="variable",
group="variable", linetype="variable", shape="variable")) +
geom_line(size=1) + labs(x = "Cost",
y = "Classification error",
colour="",group="",
linetype="",shape="") + scale_x_log10()

# Original Data, poly3 kernel
train_cv_error_svm = function(costC) {
  #Train
  spam.svm = svm(class~., data=spamtrain,
                 kernel="polynomial", degree = 3, cost=costC)
  train_error = sum(spam.svm$fitted != spamtrain$class) / nrow(spamtrain)
  #Test
  test_error = sum(predict(spam.svm, spamtest) != spamtest$class) / nrow(spamtest)
  #CV error
  spame.cverr = sapply(folds, function(fold) {
    svmcv = svm(class~.,data = spamtrain, kernel="polynomial", degree = 3, cost=costC, subset = fold$tra
    svmpred = predict(svmcv, spamtrain[fold$test,])
    return(sum(svmpred != spamtrain$class[fold$test]) / length(fold$test))
  })
  cv_error = mean(spame.cverr)
  return(c(train_error, cv_error, test_error))
}

spam_cost_errors = sapply(costs, function(cost) train_cv_error_svm(cost))
df_errs = data.frame(t(spam_cost_errors), costs)
colnames(df_errs) = c('Train', 'CV', 'Test', 'Logcost')

dataL <- melt(df_errs, id="Logcost")
ggplot(dataL, aes_string(x="Logcost", y="value", colour="variable",
group="variable", linetype="variable", shape="variable")) +
geom_line(size=1) + labs(x = "Cost",
y = "Classification error",
colour="",group="",
linetype="",shape="") + scale_x_log10()

# Original Data, Gaussian kernel
train_cv_error_svm = function(costC) {
```

```
  #Train
  spam.svm = svm(class~., data=spamtrain,
                 kernel="radial", cost=costC)
  train_error = sum(spam.svm$fitted != spamtrain$class) / nrow(spamtrain)
  #Test
  test_error = sum(predict(spam.svm, spamtest) != spamtest$class) / nrow(spamtest)
  #CV error
  spame.cverr = sapply(folds, function(fold) {
    svmcv = svm(class~.,data = spamtrain, kernel="radial", cost=costC, subset = fold$training)
    svmpred = predict(svmcv, spamtrain[fold$test,])
    return(sum(svmpred != spamtrain$class[fold$test]) / length(fold$test))
  })
  cv_error = mean(spame.cverr)
  return(c(train_error, cv_error, test_error))
}

spam_cost_errors = sapply(costs, function(cost) train_cv_error_svm(cost))
df_errs = data.frame(t(spam_cost_errors), costs)
colnames(df_errs) = c('Train', 'CV', 'Test', 'Logcost')

dataL <- melt(df_errs, id="Logcost")
ggplot(dataL, aes_string(x="Logcost", y="value", colour="variable",
group="variable", linetype="variable", shape="variable")) +
geom_line(size=1) + labs(x = "Cost",
y = "Classification error",
colour="",group="",
linetype="",shape="") + scale_x_log10()

# Original Standardized Data, linear kernel
folds = cv_partition(standspamtrain$class, num_folds = 5)
costs = c(exp(-5), exp(-4), exp(-3), exp(-1), exp(1), exp(3), exp(5), exp(7))

train_cv_error_svm = function(costC) {
  #Train
  spam.svm = svm(class~., data=standspamtrain,
                 kernel="linear", cost=costC)
  train_error = sum(spam.svm$fitted != standspamtrain$class) / nrow(standspamtrain)
  #Test
  test_error = sum(predict(spam.svm, standspamtest) != standspamtest$class) / nrow(standspamtest)
  #CV error
  spame.cverr = sapply(folds, function(fold) {
    svmcv = svm(class~.,data = standspamtrain, kernel="linear", cost=costC, subset = fold$training)
    svmpred = predict(svmcv, standspamtrain[fold$test,])
    return(sum(svmpred != standspamtrain$class[fold$test]) / length(fold$test))
  })
  cv_error = mean(spame.cverr)
  return(c(train_error, cv_error, test_error))
}

spam_cost_errors = sapply(costs, function(cost) train_cv_error_svm(cost))
df_errs = data.frame(t(spam_cost_errors), costs)
colnames(df_errs) = c('Train', 'CV', 'Test', 'Logcost')
```

```r
dataL <- melt(df_errs, id="Logcost")
ggplot(dataL, aes_string(x="Logcost", y="value", colour="variable",
group="variable", linetype="variable", shape="variable")) +
geom_line(size=1) + labs(x = "Cost",
y = "Classification error",
colour="",group="",
linetype="",shape="") + scale_x_log10()

# Original Standardized Data, poly2 kernel

train_cv_error_svm = function(costC) {
  #Train
  spam.svm = svm(class~., data=standspamtrain,
                 kernel="polynomial", degree = 2, cost=costC)
  train_error = sum(spam.svm$fitted != standspamtrain$class) / nrow(standspamtrain)
  #Test
  test_error = sum(predict(spam.svm, standspamtest) != standspamtest$class) / nrow(standspamtest)
  #CV error
  spame.cverr = sapply(folds, function(fold) {
    svmcv = svm(class~.,data = standspamtrain, kernel="polynomial", degree = 2, cost=costC, subset = fo
    svmpred = predict(svmcv, standspamtrain[fold$test,])
    return(sum(svmpred != standspamtrain$class[fold$test]) / length(fold$test))
  })
  cv_error = mean(spame.cverr)
  return(c(train_error, cv_error, test_error))
}

spam_cost_errors = sapply(costs, function(cost) train_cv_error_svm(cost))
df_errs = data.frame(t(spam_cost_errors), costs)
colnames(df_errs) = c('Train', 'CV', 'Test', 'Logcost')

dataL <- melt(df_errs, id="Logcost")
ggplot(dataL, aes_string(x="Logcost", y="value", colour="variable",
group="variable", linetype="variable", shape="variable")) +
geom_line(size=1) + labs(x = "Cost",
y = "Classification error",
colour="",group="",
linetype="",shape="") + scale_x_log10()

# Original Standardized Data, poly3 kernel

train_cv_error_svm = function(costC) {
  #Train
  spam.svm = svm(class~., data=standspamtrain,
                 kernel="polynomial", degree = 3, cost=costC)
  train_error = sum(spam.svm$fitted != standspamtrain$class) / nrow(standspamtrain)
  #Test
  test_error = sum(predict(spam.svm, standspamtest) != standspamtest$class) / nrow(standspamtest)
  #CV error
  spame.cverr = sapply(folds, function(fold) {
    svmcv = svm(class~.,data = standspamtrain, kernel="polynomial", degree = 3, cost=costC, subset = fo
    svmpred = predict(svmcv, standspamtrain[fold$test,])
    return(sum(svmpred != standspamtrain$class[fold$test]) / length(fold$test))
```

```
  })
  cv_error = mean(spame.cverr)
  return(c(train_error, cv_error, test_error))
}

spam_cost_errors = sapply(costs, function(cost) train_cv_error_svm(cost))
df_errs = data.frame(t(spam_cost_errors), costs)
colnames(df_errs) = c('Train', 'CV', 'Test', 'Logcost')

dataL <- melt(df_errs, id="Logcost")
ggplot(dataL, aes_string(x="Logcost", y="value", colour="variable",
group="variable", linetype="variable", shape="variable")) +
geom_line(size=1) + labs(x = "Cost",
y = "Classification error",
colour="",group="",
linetype="",shape="") + scale_x_log10()

# Original Standardized Data, Gaussian kernel
train_cv_error_svm = function(costC) {
  #Train
  spam.svm = svm(class~., data=standspamtrain,
                 kernel="radial", cost=costC)
  train_error = sum(spam.svm$fitted != standspamtrain$class) / nrow(standspamtrain)
  #Test
  test_error = sum(predict(spam.svm, standspamtest) != standspamtest$class) / nrow(standspamtest)
  #CV error
  spame.cverr = sapply(folds, function(fold) {
    svmcv = svm(class~.,data = standspamtrain, kernel="radial", cost=costC, subset = fold$training)
    svmpred = predict(svmcv, standspamtrain[fold$test,])
    return(sum(svmpred != standspamtrain$class[fold$test]) / length(fold$test))
  })
  cv_error = mean(spame.cverr)
  return(c(train_error, cv_error, test_error))
}

spam_cost_errors = sapply(costs, function(cost) train_cv_error_svm(cost))
df_errs = data.frame(t(spam_cost_errors), costs)
colnames(df_errs) = c('Train', 'CV', 'Test', 'Logcost')

dataL <- melt(df_errs, id="Logcost")
ggplot(dataL, aes_string(x="Logcost", y="value", colour="variable",
group="variable", linetype="variable", shape="variable")) +
geom_line(size=1) + labs(x = "Cost",
y = "Classification error",
colour="",group="",
linetype="",shape="") + scale_x_log10()
```

```
## Data Formation & Cleaning, Ratio = 3:7 spam:non-spam ##

# Indicate which rows are spam and which aren't
rowsofspam = which(spamtrain$class == 1)
rowsofnotspam = which(spamtrain$class == 0)

# Training subset (size 2000) out of the original train set needs these numbers to satisfy the ratio
```

```
trainspamtargets = sample(rowsofspam, size = 600, replace = FALSE)
trainnonspamtargets = sample(rowsofnotspam, size = 1400, replace = FALSE)

# Test subset (size 1065) out of the original train set needs these numbers to satisfy the ratio
testspamtargets = sample(rowsofspam, size = 320, replace = FALSE)
testnonspamtargets = sample(rowsofnotspam, size = 745, replace = FALSE)

# Combine targets to point what rows in original train set to be splitting by
traincombinetargets = c(trainspamtargets, trainnonspamtargets)
testcombinetargets = c(testspamtargets, testnonspamtargets)

# New train and test datasets, with the spam:non-spam ratio needed
imbspamtrain37 = spamtrain[traincombinetargets,]
imbspamtest37 = spamtrain[testcombinetargets,]

# Standardized form of datasets
imbstandspamtrain37 = imbspamtrain37
imbstandspamtrain37[,1:57] = scale(imbstandspamtrain37[,1:57])

imbstandspamtest37 = imbspamtest37
imbstandspamtest37[,1:57] = scale(imbstandspamtest37[,1:57])

## Data Formation & Cleaning, Ratio = 2:8 spam:non-spam ##

# Training subset (size 2000) out of the original train set needs these numbers to satisfy the ratio
trainspamtargets = sample(rowsofspam, size = 400, replace = FALSE)
trainnonspamtargets = sample(rowsofnotspam, size = 1600, replace = FALSE)

# Test subset (size 1065) out of the original train set needs these numbers to satisfy the ratio
testspamtargets = sample(rowsofspam, size = 213, replace = FALSE)
testnonspamtargets = sample(rowsofnotspam, size = 852, replace = FALSE)

# Combine targets to point what rows in original train set to be splitting by
traincombinetargets = c(trainspamtargets, trainnonspamtargets)
testcombinetargets = c(testspamtargets, testnonspamtargets)

# New train and test datasets, with the spam:non-spam ratio needed
imbspamtrain28 = spamtrain[traincombinetargets,]
imbspamtest28 = spamtrain[testcombinetargets,]

# Standardized form of datasets
imbstandspamtrain28 = imbspamtrain28
imbstandspamtrain28[,1:57] = scale(imbstandspamtrain28[,1:57])

imbstandspamtest28 = imbspamtest28
imbstandspamtest28[,1:57] = scale(imbstandspamtest28[,1:57])

## Data Formation & Cleaning, Ratio = 1:9 spam:non-spam ##

# Training subset (size 2000) out of the original train set needs these numbers to satisfy the ratio
trainspamtargets = sample(rowsofspam, size = 200, replace = FALSE)
trainnonspamtargets = sample(rowsofnotspam, size = 1800, replace = FALSE)
```

```r
# Test subset (size 1065) out of the original train set needs these numbers to satisfy the ratio
testspamtargets = sample(rowsofspam, size = 107, replace = FALSE)
testnonspamtargets = sample(rowsofnotspam, size = 958, replace = FALSE)

# Combine targets to point what rows in original train set to be splitting by
traincombinetargets = c(trainspamtargets, trainnonspamtargets)
testcombinetargets = c(testspamtargets, testnonspamtargets)

# New train and test datasets, with the spam:non-spam ratio needed
imbspamtrain19 = spamtrain[traincombinetargets,]
imbspamtest19 = spamtrain[testcombinetargets,]

# Standardized form of datasets
imbstandspamtrain19 = imbspamtrain19
imbstandspamtrain19[,1:57] = scale(imbstandspamtrain19[,1:57])

imbstandspamtest19 = imbspamtest19
imbstandspamtest19[,1:57] = scale(imbstandspamtest19[,1:57])

# 3:7 Data, linear kernel
folds = cv_partition(imbspamtrain37$class, num_folds = 5)
costs = c(exp(-5), exp(-4), exp(-3), exp(-1), exp(1), exp(3), exp(5), exp(7))

train_cv_error_svm = function(costC) {
  #Train
  spam.svm = svm(class~., data=imbspamtrain37,
                 kernel="linear", cost=costC)
  train_error = sum(spam.svm$fitted != imbspamtrain37$class) / nrow(imbspamtrain37)
  #Test
  test_error = sum(predict(spam.svm, imbspamtest37) != imbspamtest37$class) / nrow(imbspamtest37)
  #CV error
  spame.cverr = sapply(folds, function(fold) {
    svmcv = svm(class~.,data = imbspamtrain37, kernel="linear", cost=costC, subset = fold$training)
    svmpred = predict(svmcv, imbspamtrain37[fold$test,])
    return(sum(svmpred != imbspamtrain37$class[fold$test]) / length(fold$test))
  })
  cv_error = mean(spame.cverr)
  return(c(train_error, cv_error, test_error))
}

spam_cost_errors = sapply(costs, function(cost) train_cv_error_svm(cost))
df_errs = data.frame(t(spam_cost_errors), costs)
colnames(df_errs) = c('Train', 'CV', 'Test', 'Logcost')

dataL <- melt(df_errs, id="Logcost")
ggplot(dataL, aes_string(x="Logcost", y="value", colour="variable",
group="variable", linetype="variable", shape="variable")) +
geom_line(size=1) + labs(x = "Cost",
y = "Classification error",
colour="",group="",
linetype="",shape="") + scale_x_log10()

# 3:7 Data, poly2 kernel
```

```
train_cv_error_svm = function(costC) {
  #Train
  spam.svm = svm(class~., data=imbspamtrain37,
                 kernel="polynomial", degree = 2, cost=costC)
  train_error = sum(spam.svm$fitted != imbspamtrain37$class) / nrow(imbspamtrain37)
  #Test
  test_error = sum(predict(spam.svm, imbspamtest37) != imbspamtest37$class) / nrow(imbspamtest37)
  #CV error
  spame.cverr = sapply(folds, function(fold) {
    svmcv = svm(class~.,data = imbspamtrain37, kernel="polynomial", degree = 2, cost=costC, subset = fo
    svmpred = predict(svmcv, imbspamtrain37[fold$test,])
    return(sum(svmpred != imbspamtrain37$class[fold$test]) / length(fold$test))
  })
  cv_error = mean(spame.cverr)
  return(c(train_error, cv_error, test_error))
}

spam_cost_errors = sapply(costs, function(cost) train_cv_error_svm(cost))
df_errs = data.frame(t(spam_cost_errors), costs)
colnames(df_errs) = c('Train', 'CV', 'Test', 'Logcost')

dataL <- melt(df_errs, id="Logcost")
ggplot(dataL, aes_string(x="Logcost", y="value", colour="variable",
group="variable", linetype="variable", shape="variable")) +
geom_line(size=1) + labs(x = "Cost",
y = "Classification error",
colour="",group="",
linetype="",shape="") + scale_x_log10()

# 3:7 Data, poly3 kernel
train_cv_error_svm = function(costC) {
  #Train
  spam.svm = svm(class~., data=imbspamtrain37,
                 kernel="polynomial", degree = 3, cost=costC)
  train_error = sum(spam.svm$fitted != imbspamtrain37$class) / nrow(imbspamtrain37)
  #Test
  test_error = sum(predict(spam.svm, imbspamtest37) != imbspamtest37$class) / nrow(imbspamtest37)
  #CV error
  spame.cverr = sapply(folds, function(fold) {
    svmcv = svm(class~.,data = imbspamtrain37, kernel="polynomial", degree = 3, cost=costC, subset = fo
    svmpred = predict(svmcv, imbspamtrain37[fold$test,])
    return(sum(svmpred != imbspamtrain37$class[fold$test]) / length(fold$test))
  })
  cv_error = mean(spame.cverr)
  return(c(train_error, cv_error, test_error))
}

spam_cost_errors = sapply(costs, function(cost) train_cv_error_svm(cost))
df_errs = data.frame(t(spam_cost_errors), costs)
colnames(df_errs) = c('Train', 'CV', 'Test', 'Logcost')

dataL <- melt(df_errs, id="Logcost")
ggplot(dataL, aes_string(x="Logcost", y="value", colour="variable",
```

```
group="variable", linetype="variable", shape="variable")) +
geom_line(size=1) + labs(x = "Cost",
y = "Classification error",
colour="",group="",
linetype="",shape="") + scale_x_log10()

# 3:7 Data, Gaussian kernel
train_cv_error_svm = function(costC) {
  #Train
  spam.svm = svm(class~., data=imbspamtrain37,
                 kernel="radial", cost=costC)
  train_error = sum(spam.svm$fitted != imbspamtrain37$class) / nrow(imbspamtrain37)
  #Test
  test_error = sum(predict(spam.svm, imbspamtest37) != imbspamtest37$class) / nrow(imbspamtest37)
  #CV error
  spame.cverr = sapply(folds, function(fold) {
    svmcv = svm(class~.,data = imbspamtrain37, kernel="radial", cost=costC, subset = fold$training)
    svmpred = predict(svmcv, imbspamtrain37[fold$test,])
    return(sum(svmpred != imbspamtrain37$class[fold$test]) / length(fold$test))
  })
  cv_error = mean(spame.cverr)
  return(c(train_error, cv_error, test_error))
}

spam_cost_errors = sapply(costs, function(cost) train_cv_error_svm(cost))
df_errs = data.frame(t(spam_cost_errors), costs)
colnames(df_errs) = c('Train', 'CV', 'Test', 'Logcost')

dataL <- melt(df_errs, id="Logcost")
ggplot(dataL, aes_string(x="Logcost", y="value", colour="variable",
group="variable", linetype="variable", shape="variable")) +
geom_line(size=1) + labs(x = "Cost",
y = "Classification error",
colour="",group="",
linetype="",shape="") + scale_x_log10()

# 3:7 Standardized Data, linear kernel
folds = cv_partition(imbstandspamtrain37$class, num_folds = 5)
costs = c(exp(-5), exp(-4), exp(-3), exp(-1), exp(1), exp(3), exp(5), exp(7))

train_cv_error_svm = function(costC) {
  #Train
  spam.svm = svm(class~., data=imbstandspamtrain37,
                 kernel="linear", cost=costC)
  train_error = sum(spam.svm$fitted != imbstandspamtrain37$class) / nrow(imbstandspamtrain37)
  #Test
  test_error = sum(predict(spam.svm, imbstandspamtest37) != imbstandspamtest37$class) / nrow(imbstandspa
  #CV error
  spame.cverr = sapply(folds, function(fold) {
    svmcv = svm(class~.,data = imbstandspamtrain37, kernel="linear", cost=costC, subset = fold$training
    svmpred = predict(svmcv, imbstandspamtrain37[fold$test,])
    return(sum(svmpred != imbstandspamtrain37$class[fold$test]) / length(fold$test))
  })
```

```
  cv_error = mean(spame.cverr)
  return(c(train_error, cv_error, test_error))
}

spam_cost_errors = sapply(costs, function(cost) train_cv_error_svm(cost))
df_errs = data.frame(t(spam_cost_errors), costs)
colnames(df_errs) = c('Train', 'CV', 'Test', 'Logcost')

dataL <- melt(df_errs, id="Logcost")
ggplot(dataL, aes_string(x="Logcost", y="value", colour="variable",
group="variable", linetype="variable", shape="variable")) +
geom_line(size=1) + labs(x = "Cost",
y = "Classification error",
colour="",group="",
linetype="",shape="") + scale_x_log10()

# 3:7 Standardized Data, poly2 kernel

train_cv_error_svm = function(costC) {
  #Train
  spam.svm = svm(class~., data=imbstandspamtrain37,
                 kernel="polynomial", degree = 2, cost=costC)
  train_error = sum(spam.svm$fitted != imbstandspamtrain37$class) / nrow(imbstandspamtrain37)
  #Test
  test_error = sum(predict(spam.svm, imbstandspamtest37) != imbstandspamtest37$class) / nrow(imbstandspa
  #CV error
  spame.cverr = sapply(folds, function(fold) {
    svmcv = svm(class~.,data = imbstandspamtrain37, kernel="polynomial", degree = 2, cost=costC, subset
    svmpred = predict(svmcv, imbstandspamtrain37[fold$test,])
    return(sum(svmpred != imbstandspamtrain37$class[fold$test]) / length(fold$test))
  })
  cv_error = mean(spame.cverr)
  return(c(train_error, cv_error, test_error))
}

spam_cost_errors = sapply(costs, function(cost) train_cv_error_svm(cost))
df_errs = data.frame(t(spam_cost_errors), costs)
colnames(df_errs) = c('Train', 'CV', 'Test', 'Logcost')

dataL <- melt(df_errs, id="Logcost")
ggplot(dataL, aes_string(x="Logcost", y="value", colour="variable",
group="variable", linetype="variable", shape="variable")) +
geom_line(size=1) + labs(x = "Cost",
y = "Classification error",
colour="",group="",
linetype="",shape="") + scale_x_log10()

# 3:7 Standardized Data, poly3 kernel
train_cv_error_svm = function(costC) {
  #Train
  spam.svm = svm(class~., data=imbstandspamtrain37,
                 kernel="polynomial", degree = 3, cost=costC)
  train_error = sum(spam.svm$fitted != imbstandspamtrain37$class) / nrow(imbstandspamtrain37)
```

```
  #Test
  test_error = sum(predict(spam.svm, imbstandspamtest37) != imbstandspamtest37$class) / nrow(imbstandspa
  #CV error
  spame.cverr = sapply(folds, function(fold) {
    svmcv = svm(class~.,data = imbstandspamtrain37, kernel="polynomial", degree = 3, cost=costC, subset
    svmpred = predict(svmcv, imbstandspamtrain37[fold$test,])
    return(sum(svmpred != imbstandspamtrain37$class[fold$test]) / length(fold$test))
  })
  cv_error = mean(spame.cverr)
  return(c(train_error, cv_error, test_error))
}

spam_cost_errors = sapply(costs, function(cost) train_cv_error_svm(cost))
df_errs = data.frame(t(spam_cost_errors), costs)
colnames(df_errs) = c('Train', 'CV', 'Test', 'Logcost')

dataL <- melt(df_errs, id="Logcost")
ggplot(dataL, aes_string(x="Logcost", y="value", colour="variable",
group="variable", linetype="variable", shape="variable")) +
geom_line(size=1) + labs(x = "Cost",
y = "Classification error",
colour="",group="",
linetype="",shape="") + scale_x_log10()

# 3:7 Standardized Data, Gaussian kernel
train_cv_error_svm = function(costC) {
  #Train
  spam.svm = svm(class~., data=imbstandspamtrain37,
                 kernel="radial", cost=costC)
  train_error = sum(spam.svm$fitted != imbstandspamtrain37$class) / nrow(imbstandspamtrain37)
  #Test
  test_error = sum(predict(spam.svm, imbstandspamtest37) != imbstandspamtest37$class) / nrow(imbstandspa
  #CV error
  spame.cverr = sapply(folds, function(fold) {
    svmcv = svm(class~.,data = imbstandspamtrain37, kernel="radial", cost=costC, subset = fold$training
    svmpred = predict(svmcv, imbstandspamtrain37[fold$test,])
    return(sum(svmpred != imbstandspamtrain37$class[fold$test]) / length(fold$test))
  })
  cv_error = mean(spame.cverr)
  return(c(train_error, cv_error, test_error))
}

spam_cost_errors = sapply(costs, function(cost) train_cv_error_svm(cost))
df_errs = data.frame(t(spam_cost_errors), costs)
colnames(df_errs) = c('Train', 'CV', 'Test', 'Logcost')

dataL <- melt(df_errs, id="Logcost")
ggplot(dataL, aes_string(x="Logcost", y="value", colour="variable",
group="variable", linetype="variable", shape="variable")) +
geom_line(size=1) + labs(x = "Cost",
y = "Classification error",
colour="",group="",
linetype="",shape="") + scale_x_log10()
```

```r
# 2:8 Data, linear kernel
folds = cv_partition(imbspamtrain28$class, num_folds = 5)
costs = c(exp(-5), exp(-4), exp(-3), exp(-1), exp(1), exp(3), exp(5), exp(7))

train_cv_error_svm = function(costC) {
  #Train
  spam.svm = svm(class~., data=imbspamtrain28,
                 kernel="linear", cost=costC)
  train_error = sum(spam.svm$fitted != imbspamtrain28$class) / nrow(imbspamtrain28)
  #Test
  test_error = sum(predict(spam.svm, imbspamtest28) != imbspamtest28$class) / nrow(imbspamtest28)
  #CV error
  spame.cverr = sapply(folds, function(fold) {
    svmcv = svm(class~.,data = imbspamtrain28, kernel="linear", cost=costC, subset = fold$training)
    svmpred = predict(svmcv, imbspamtrain28[fold$test,])
    return(sum(svmpred != imbspamtrain28$class[fold$test]) / length(fold$test))
  })
  cv_error = mean(spame.cverr)
  return(c(train_error, cv_error, test_error))
}

spam_cost_errors = sapply(costs, function(cost) train_cv_error_svm(cost))
df_errs = data.frame(t(spam_cost_errors), costs)
colnames(df_errs) = c('Train', 'CV', 'Test', 'Logcost')

dataL <- melt(df_errs, id="Logcost")
ggplot(dataL, aes_string(x="Logcost", y="value", colour="variable",
group="variable", linetype="variable", shape="variable")) +
geom_line(size=1) + labs(x = "Cost",
y = "Classification error",
colour="",group="",
linetype="",shape="") + scale_x_log10()

# 2:8 Data, poly2 kernel

train_cv_error_svm = function(costC) {
  #Train
  spam.svm = svm(class~., data=imbspamtrain28,
                 kernel="polynomial", degree = 2, cost=costC)
  train_error = sum(spam.svm$fitted != imbspamtrain28$class) / nrow(imbspamtrain28)
  #Test
  test_error = sum(predict(spam.svm, imbspamtest28) != imbspamtest28$class) / nrow(imbspamtest28)
  #CV error
  spame.cverr = sapply(folds, function(fold) {
    svmcv = svm(class~.,data = imbspamtrain28, kernel="polynomial", degree = 2, cost=costC, subset = fo
    svmpred = predict(svmcv, imbspamtrain28[fold$test,])
    return(sum(svmpred != imbspamtrain28$class[fold$test]) / length(fold$test))
  })
  cv_error = mean(spame.cverr)
  return(c(train_error, cv_error, test_error))
}

spam_cost_errors = sapply(costs, function(cost) train_cv_error_svm(cost))
```

```
df_errs = data.frame(t(spam_cost_errors), costs)
colnames(df_errs) = c('Train', 'CV', 'Test', 'Logcost')

dataL <- melt(df_errs, id="Logcost")
ggplot(dataL, aes_string(x="Logcost", y="value", colour="variable",
group="variable", linetype="variable", shape="variable")) +
geom_line(size=1) + labs(x = "Cost",
y = "Classification error",
colour="",group="",
linetype="",shape="") + scale_x_log10()

# 2:8 Data, poly3 kernel
train_cv_error_svm = function(costC) {
  #Train
  spam.svm = svm(class~., data=imbspamtrain28,
                 kernel="polynomial", degree = 3, cost=costC)
  train_error = sum(spam.svm$fitted != imbspamtrain28$class) / nrow(imbspamtrain28)
  #Test
  test_error = sum(predict(spam.svm, imbspamtest28) != imbspamtest28$class) / nrow(imbspamtest28)
  #CV error
  spame.cverr = sapply(folds, function(fold) {
    svmcv = svm(class~.,data = imbspamtrain28, kernel="polynomial", degree = 3, cost=costC, subset = fo
    svmpred = predict(svmcv, imbspamtrain28[fold$test,])
    return(sum(svmpred != imbspamtrain28$class[fold$test]) / length(fold$test))
  })
  cv_error = mean(spame.cverr)
  return(c(train_error, cv_error, test_error))
}

spam_cost_errors = sapply(costs, function(cost) train_cv_error_svm(cost))
df_errs = data.frame(t(spam_cost_errors), costs)
colnames(df_errs) = c('Train', 'CV', 'Test', 'Logcost')

dataL <- melt(df_errs, id="Logcost")
ggplot(dataL, aes_string(x="Logcost", y="value", colour="variable",
group="variable", linetype="variable", shape="variable")) +
geom_line(size=1) + labs(x = "Cost",
y = "Classification error",
colour="",group="",
linetype="",shape="") + scale_x_log10()

# 2:8 Data, Gaussian kernel
train_cv_error_svm = function(costC) {
  #Train
  spam.svm = svm(class~., data=imbspamtrain28,
                 kernel="radial", cost=costC)
  train_error = sum(spam.svm$fitted != imbspamtrain28$class) / nrow(imbspamtrain28)
  #Test
  test_error = sum(predict(spam.svm, imbspamtest28) != imbspamtest28$class) / nrow(imbspamtest28)
  #CV error
  spame.cverr = sapply(folds, function(fold) {
    svmcv = svm(class~.,data = imbspamtrain28, kernel="radial", cost=costC, subset = fold$training)
    svmpred = predict(svmcv, imbspamtrain28[fold$test,])
```

```
    return(sum(svmpred != imbspamtrain28$class[fold$test]) / length(fold$test))
  })
  cv_error = mean(spame.cverr)
  return(c(train_error, cv_error, test_error))
}

spam_cost_errors = sapply(costs, function(cost) train_cv_error_svm(cost))
df_errs = data.frame(t(spam_cost_errors), costs)
colnames(df_errs) = c('Train', 'CV', 'Test', 'Logcost')

dataL <- melt(df_errs, id="Logcost")
ggplot(dataL, aes_string(x="Logcost", y="value", colour="variable",
group="variable", linetype="variable", shape="variable")) +
geom_line(size=1) + labs(x = "Cost",
y = "Classification error",
colour="",group="",
linetype="",shape="") + scale_x_log10()

# 2:8 Standardized Data, linear kernel
folds = cv_partition(imbstandspamtrain28$class, num_folds = 5)
costs = c(exp(-5), exp(-4), exp(-3), exp(-1), exp(1), exp(3), exp(5), exp(7))

train_cv_error_svm = function(costC) {
  #Train
  spam.svm = svm(class~., data=imbstandspamtrain28,
                 kernel="linear", cost=costC)
  train_error = sum(spam.svm$fitted != imbstandspamtrain28$class) / nrow(imbstandspamtrain28)
  #Test
  test_error = sum(predict(spam.svm, imbstandspamtest28) != imbstandspamtest28$class) / nrow(imbstandsp
  #CV error
  spame.cverr = sapply(folds, function(fold) {
    svmcv = svm(class~.,data = imbstandspamtrain28, kernel="linear", cost=costC, subset = fold$training
    svmpred = predict(svmcv, imbstandspamtrain28[fold$test,])
    return(sum(svmpred != imbstandspamtrain28$class[fold$test]) / length(fold$test))
  })
  cv_error = mean(spame.cverr)
  return(c(train_error, cv_error, test_error))
}

spam_cost_errors = sapply(costs, function(cost) train_cv_error_svm(cost))
df_errs = data.frame(t(spam_cost_errors), costs)
colnames(df_errs) = c('Train', 'CV', 'Test', 'Logcost')

dataL <- melt(df_errs, id="Logcost")
ggplot(dataL, aes_string(x="Logcost", y="value", colour="variable",
group="variable", linetype="variable", shape="variable")) +
geom_line(size=1) + labs(x = "Cost",
y = "Classification error",
colour="",group="",
linetype="",shape="") + scale_x_log10()

# 2:8 Standardized Data, poly2 kernel
```

```
train_cv_error_svm = function(costC) {
  #Train
  spam.svm = svm(class~., data=imbstandspamtrain28,
                 kernel="polynomial", degree = 2, cost=costC)
  train_error = sum(spam.svm$fitted != imbstandspamtrain28$class) / nrow(imbstandspamtrain28)
  #Test
  test_error = sum(predict(spam.svm, imbstandspamtest28) != imbstandspamtest28$class) / nrow(imbstandspa
  #CV error
  spame.cverr = sapply(folds, function(fold) {
    svmcv = svm(class~.,data = imbstandspamtrain28, kernel="polynomial", degree = 2, cost=costC, subset
    svmpred = predict(svmcv, imbstandspamtrain28[fold$test,])
    return(sum(svmpred != imbstandspamtrain28$class[fold$test]) / length(fold$test))
  })
  cv_error = mean(spame.cverr)
  return(c(train_error, cv_error, test_error))
}

spam_cost_errors = sapply(costs, function(cost) train_cv_error_svm(cost))
df_errs = data.frame(t(spam_cost_errors), costs)
colnames(df_errs) = c('Train', 'CV', 'Test', 'Logcost')

dataL <- melt(df_errs, id="Logcost")
ggplot(dataL, aes_string(x="Logcost", y="value", colour="variable",
group="variable", linetype="variable", shape="variable")) +
geom_line(size=1) + labs(x = "Cost",
y = "Classification error",
colour="",group="",
linetype="",shape="") + scale_x_log10()

# 2:8 Standardized Data, poly3 kernel
train_cv_error_svm = function(costC) {
  #Train
  spam.svm = svm(class~., data=imbstandspamtrain28,
                 kernel="polynomial", degree = 3, cost=costC)
  train_error = sum(spam.svm$fitted != imbstandspamtrain28$class) / nrow(imbstandspamtrain28)
  #Test
  test_error = sum(predict(spam.svm, imbstandspamtest28) != imbstandspamtest28$class) / nrow(imbstandspa
  #CV error
  spame.cverr = sapply(folds, function(fold) {
    svmcv = svm(class~.,data = imbstandspamtrain28, kernel="polynomial", degree = 3, cost=costC, subset
    svmpred = predict(svmcv, imbstandspamtrain28[fold$test,])
    return(sum(svmpred != imbstandspamtrain28$class[fold$test]) / length(fold$test))
  })
  cv_error = mean(spame.cverr)
  return(c(train_error, cv_error, test_error))
}

spam_cost_errors = sapply(costs, function(cost) train_cv_error_svm(cost))
df_errs = data.frame(t(spam_cost_errors), costs)
colnames(df_errs) = c('Train', 'CV', 'Test', 'Logcost')

dataL <- melt(df_errs, id="Logcost")
ggplot(dataL, aes_string(x="Logcost", y="value", colour="variable",
```

```
group="variable", linetype="variable", shape="variable")) +
geom_line(size=1) + labs(x = "Cost",
y = "Classification error",
colour="",group="",
linetype="",shape="") + scale_x_log10()

# 2:8 Standardized Data, Gaussian kernel
train_cv_error_svm = function(costC) {
  #Train
  spam.svm = svm(class~., data=imbstandspamtrain28,
                 kernel="radial", cost=costC)
  train_error = sum(spam.svm$fitted != imbstandspamtrain28$class) / nrow(imbstandspamtrain28)
  #Test
  test_error = sum(predict(spam.svm, imbstandspamtest28) != imbstandspamtest28$class) / nrow(imbstandspa
  #CV error
  spame.cverr = sapply(folds, function(fold) {
    svmcv = svm(class~.,data = imbstandspamtrain28, kernel="radial", cost=costC, subset = fold$training
    svmpred = predict(svmcv, imbstandspamtrain28[fold$test,])
    return(sum(svmpred != imbstandspamtrain28$class[fold$test]) / length(fold$test))
  })
  cv_error = mean(spame.cverr)
  return(c(train_error, cv_error, test_error))
}

spam_cost_errors = sapply(costs, function(cost) train_cv_error_svm(cost))
df_errs = data.frame(t(spam_cost_errors), costs)
colnames(df_errs) = c('Train', 'CV', 'Test', 'Logcost')

dataL <- melt(df_errs, id="Logcost")
ggplot(dataL, aes_string(x="Logcost", y="value", colour="variable",
group="variable", linetype="variable", shape="variable")) +
geom_line(size=1) + labs(x = "Cost",
y = "Classification error",
colour="",group="",
linetype="",shape="") + scale_x_log10()

# 1:9 Data, linear kernel
folds = cv_partition(imbspamtrain19$class, num_folds = 5)
costs = c(exp(-5), exp(-4), exp(-3), exp(-1), exp(1), exp(3), exp(5), exp(7))

train_cv_error_svm = function(costC) {
  #Train
  spam.svm = svm(class~., data=imbspamtrain19,
                 kernel="linear", cost=costC)
  train_error = sum(spam.svm$fitted != imbspamtrain19$class) / nrow(imbspamtrain19)
  #Test
  test_error = sum(predict(spam.svm, imbspamtest19) != imbspamtest19$class) / nrow(imbspamtest19)
  #CV error
  spame.cverr = sapply(folds, function(fold) {
    svmcv = svm(class~.,data = imbspamtrain19, kernel="linear", cost=costC, subset = fold$training)
    svmpred = predict(svmcv, imbspamtrain19[fold$test,])
    return(sum(svmpred != imbspamtrain19$class[fold$test]) / length(fold$test))
  })
```

```
  cv_error = mean(spame.cverr)
  return(c(train_error, cv_error, test_error))
}

spam_cost_errors = sapply(costs, function(cost) train_cv_error_svm(cost))
df_errs = data.frame(t(spam_cost_errors), costs)
colnames(df_errs) = c('Train', 'CV', 'Test', 'Logcost')

dataL <- melt(df_errs, id="Logcost")
ggplot(dataL, aes_string(x="Logcost", y="value", colour="variable",
group="variable", linetype="variable", shape="variable")) +
geom_line(size=1) + labs(x = "Cost",
y = "Classification error",
colour="",group="",
linetype="",shape="") + scale_x_log10()

# 1:9 Data, poly2 kernel

train_cv_error_svm = function(costC) {
  #Train
  spam.svm = svm(class~., data=imbspamtrain19,
                 kernel="polynomial", degree = 2, cost=costC)
  train_error = sum(spam.svm$fitted != imbspamtrain19$class) / nrow(imbspamtrain19)
  #Test
  test_error = sum(predict(spam.svm, imbspamtest19) != imbspamtest19$class) / nrow(imbspamtest19)
  #CV error
  spame.cverr = sapply(folds, function(fold) {
    svmcv = svm(class~.,data = imbspamtrain19, kernel="polynomial", degree = 2, cost=costC, subset = fo
    svmpred = predict(svmcv, imbspamtrain19[fold$test,])
    return(sum(svmpred != imbspamtrain19$class[fold$test]) / length(fold$test))
  })
  cv_error = mean(spame.cverr)
  return(c(train_error, cv_error, test_error))
}

spam_cost_errors = sapply(costs, function(cost) train_cv_error_svm(cost))
df_errs = data.frame(t(spam_cost_errors), costs)
colnames(df_errs) = c('Train', 'CV', 'Test', 'Logcost')

dataL <- melt(df_errs, id="Logcost")
ggplot(dataL, aes_string(x="Logcost", y="value", colour="variable",
group="variable", linetype="variable", shape="variable")) +
geom_line(size=1) + labs(x = "Cost",
y = "Classification error",
colour="",group="",
linetype="",shape="") + scale_x_log10()

# 1:9 Data, poly3 kernel
train_cv_error_svm = function(costC) {
  #Train
  spam.svm = svm(class~., data=imbspamtrain19,
                 kernel="polynomial", degree = 3, cost=costC)
  train_error = sum(spam.svm$fitted != imbspamtrain19$class) / nrow(imbspamtrain19)
```

```
  #Test
  test_error = sum(predict(spam.svm, imbspamtest19) != imbspamtest19$class) / nrow(imbspamtest19)
  #CV error
  spame.cverr = sapply(folds, function(fold) {
    svmcv = svm(class~.,data = imbspamtrain19, kernel="polynomial", degree = 3, cost=costC, subset = fo
    svmpred = predict(svmcv, imbspamtrain19[fold$test,])
    return(sum(svmpred != imbspamtrain19$class[fold$test]) / length(fold$test))
  })
  cv_error = mean(spame.cverr)
  return(c(train_error, cv_error, test_error))
}

spam_cost_errors = sapply(costs, function(cost) train_cv_error_svm(cost))
df_errs = data.frame(t(spam_cost_errors), costs)
colnames(df_errs) = c('Train', 'CV', 'Test', 'Logcost')

dataL <- melt(df_errs, id="Logcost")
ggplot(dataL, aes_string(x="Logcost", y="value", colour="variable",
group="variable", linetype="variable", shape="variable")) +
geom_line(size=1) + labs(x = "Cost",
y = "Classification error",
colour="",group="",
linetype="",shape="") + scale_x_log10()

# 1:9 Data, Gaussian kernel
train_cv_error_svm = function(costC) {
  #Train
  spam.svm = svm(class~., data=imbspamtrain19,
                 kernel="radial", cost=costC)
  train_error = sum(spam.svm$fitted != imbspamtrain19$class) / nrow(imbspamtrain19)
  #Test
  test_error = sum(predict(spam.svm, imbspamtest19) != imbspamtest19$class) / nrow(imbspamtest19)
  #CV error
  spame.cverr = sapply(folds, function(fold) {
    svmcv = svm(class~.,data = imbspamtrain19, kernel="radial", cost=costC, subset = fold$training)
    svmpred = predict(svmcv, imbspamtrain19[fold$test,])
    return(sum(svmpred != imbspamtrain19$class[fold$test]) / length(fold$test))
  })
  cv_error = mean(spame.cverr)
  return(c(train_error, cv_error, test_error))
}

spam_cost_errors = sapply(costs, function(cost) train_cv_error_svm(cost))
df_errs = data.frame(t(spam_cost_errors), costs)
colnames(df_errs) = c('Train', 'CV', 'Test', 'Logcost')

dataL <- melt(df_errs, id="Logcost")
ggplot(dataL, aes_string(x="Logcost", y="value", colour="variable",
group="variable", linetype="variable", shape="variable")) +
geom_line(size=1) + labs(x = "Cost",
y = "Classification error",
colour="",group="",
linetype="",shape="") + scale_x_log10()
```

```
# 1:9 Standardized Data, linear kernel
folds = cv_partition(imbstandspamtrain19$class, num_folds = 5)
costs = c(exp(-5), exp(-4), exp(-3), exp(-1), exp(1), exp(3), exp(5), exp(7))

train_cv_error_svm = function(costC) {
  #Train
  spam.svm = svm(class~., data=imbstandspamtrain19,
                 kernel="linear", cost=costC)
  train_error = sum(spam.svm$fitted != imbstandspamtrain19$class) / nrow(imbstandspamtrain19)
  #Test
  test_error = sum(predict(spam.svm, imbstandspamtest19) != imbstandspamtest19$class) / nrow(imbstandspa
  #CV error
  spame.cverr = sapply(folds, function(fold) {
    svmcv = svm(class~.,data = imbstandspamtrain19, kernel="linear", cost=costC, subset = fold$training
    svmpred = predict(svmcv, imbstandspamtrain19[fold$test,])
    return(sum(svmpred != imbstandspamtrain19$class[fold$test]) / length(fold$test))
  })
  cv_error = mean(spame.cverr)
  return(c(train_error, cv_error, test_error))
}

spam_cost_errors = sapply(costs, function(cost) train_cv_error_svm(cost))
df_errs = data.frame(t(spam_cost_errors), costs)
colnames(df_errs) = c('Train', 'CV', 'Test', 'Logcost')

dataL <- melt(df_errs, id="Logcost")
ggplot(dataL, aes_string(x="Logcost", y="value", colour="variable",
group="variable", linetype="variable", shape="variable")) +
geom_line(size=1) + labs(x = "Cost",
y = "Classification error",
colour="",group="",
linetype="",shape="") + scale_x_log10()

# 1:9 Standardized Data, poly2 kernel

train_cv_error_svm = function(costC) {
  #Train
  spam.svm = svm(class~., data=imbstandspamtrain19,
                 kernel="polynomial", degree = 2, cost=costC)
  train_error = sum(spam.svm$fitted != imbstandspamtrain19$class) / nrow(imbstandspamtrain19)
  #Test
  test_error = sum(predict(spam.svm, imbstandspamtest19) != imbstandspamtest19$class) / nrow(imbstandspa
  #CV error
  spame.cverr = sapply(folds, function(fold) {
    svmcv = svm(class~.,data = imbstandspamtrain19, kernel="polynomial", degree = 2, cost=costC, subset
    svmpred = predict(svmcv, imbstandspamtrain19[fold$test,])
    return(sum(svmpred != imbstandspamtrain19$class[fold$test]) / length(fold$test))
  })
  cv_error = mean(spame.cverr)
  return(c(train_error, cv_error, test_error))
}

spam_cost_errors = sapply(costs, function(cost) train_cv_error_svm(cost))
```

```
df_errs = data.frame(t(spam_cost_errors), costs)
colnames(df_errs) = c('Train', 'CV', 'Test', 'Logcost')

dataL <- melt(df_errs, id="Logcost")
ggplot(dataL, aes_string(x="Logcost", y="value", colour="variable",
group="variable", linetype="variable", shape="variable")) +
geom_line(size=1) + labs(x = "Cost",
y = "Classification error",
colour="",group="",
linetype="",shape="") + scale_x_log10()

# 1:9 Standardized Data, poly3 kernel
train_cv_error_svm = function(costC) {
  #Train
  spam.svm = svm(class~., data=imbstandspamtrain19,
                 kernel="polynomial", degree = 3, cost=costC)
  train_error = sum(spam.svm$fitted != imbstandspamtrain19$class) / nrow(imbstandspamtrain19)
  #Test
  test_error = sum(predict(spam.svm, imbstandspamtest19) != imbstandspamtest19$class) / nrow(imbstandspa
  #CV error
  spame.cverr = sapply(folds, function(fold) {
    svmcv = svm(class~.,data = imbstandspamtrain19, kernel="polynomial", degree = 3, cost=costC, subset
    svmpred = predict(svmcv, imbstandspamtrain19[fold$test,])
    return(sum(svmpred != imbstandspamtrain19$class[fold$test]) / length(fold$test))
  })
  cv_error = mean(spame.cverr)
  return(c(train_error, cv_error, test_error))
}

spam_cost_errors = sapply(costs, function(cost) train_cv_error_svm(cost))
df_errs = data.frame(t(spam_cost_errors), costs)
colnames(df_errs) = c('Train', 'CV', 'Test', 'Logcost')

dataL <- melt(df_errs, id="Logcost")
ggplot(dataL, aes_string(x="Logcost", y="value", colour="variable",
group="variable", linetype="variable", shape="variable")) +
geom_line(size=1) + labs(x = "Cost",
y = "Classification error",
colour="",group="",
linetype="",shape="") + scale_x_log10()

# 1:9 Standardized Data, Gaussian kernel
train_cv_error_svm = function(costC) {
  #Train
  spam.svm = svm(class~., data=imbstandspamtrain19,
                 kernel="radial", cost=costC)
  train_error = sum(spam.svm$fitted != imbstandspamtrain19$class) / nrow(imbstandspamtrain19)
  #Test
  test_error = sum(predict(spam.svm, imbstandspamtest19) != imbstandspamtest19$class) / nrow(imbstandspa
  #CV error
  spame.cverr = sapply(folds, function(fold) {
    svmcv = svm(class~.,data = imbstandspamtrain19, kernel="radial", cost=costC, subset = fold$training
    svmpred = predict(svmcv, imbstandspamtrain19[fold$test,])
```

```
    return(sum(svmpred != imbstandspamtrain19$class[fold$test]) / length(fold$test))
  })
  cv_error = mean(spame.cverr)
  return(c(train_error, cv_error, test_error))
}

spam_cost_errors = sapply(costs, function(cost) train_cv_error_svm(cost))
df_errs = data.frame(t(spam_cost_errors), costs)
colnames(df_errs) = c('Train', 'CV', 'Test', 'Logcost')

dataL <- melt(df_errs, id="Logcost")
ggplot(dataL, aes_string(x="Logcost", y="value", colour="variable",
group="variable", linetype="variable", shape="variable")) +
geom_line(size=1) + labs(x = "Cost",
y = "Classification error",
colour="",group="",
linetype="",shape="") + scale_x_log10()

# Bootstrapping Sample Code

funcboot = function(data= spamtrain, class = "class", samp_index_loc){
  newdata = data.frame(cat=data[samp_index_loc,which(colnames(data)==class)],samp_index_loc)
  bootstrapn = length(which(newdata$class == 0)) - length(which(newdata$class == 1)
  bootstrapsamples(newdata$class == 1 = sample(x=which(newdata$class == 1, size=bootstrapn, replace = TR
  out_index = c(which(newdata$class == 0), which(newdata$class == 1, bootstrapsamples(newdata$class == 1
  bts_index = newdata[out_index,]$samp_index_loc
return(bts_index)
}

index1to9 = funcboot(spamtrain, samp_index_loc = which(df_svm2$ratio == "90/10"))
index2to8 = funcboot(spamtrain, samp_index_loc = which(df_svm2$ratio == "80/20"))
index3to7 = funcboot(spamtrain, samp_index_loc = which(df_svm2$ratio == "70/30"))
```

```
### NEURAL NET CODE SECTION ###

y_names = colnames(standspamtrain)[ncol(standspamtrain)]
x_names = colnames(standspamtrain)[-ncol(standspamtrain)]
formula = paste(paste(y_names, collapse='+'),'~',paste(x_names, collapse='+'))
folds = cv_partition(standspamtrain$class, num_folds = 5)

neural_net_layer1 = function() {

  spam = neuralnet(formula,
                   data=standspamtrain,
                   hidden = c(3),
                   linear.output=FALSE,
                   lifesign='full',
                   lifesign.step = 5000,
                   stepmax = 1e+05)

  pred = function(nn, dat) {
      result = compute(nn, dat[,-ncol(dat)])$net.result[,1]
      result = round(result, 0)
      return(mean(result != dat$class))
```

```
  }

  train_error = pred(spam, standspamtrain)

  test_error = pred(spam, standspamtest)

  autoe.cverr = sapply(folds, function(fold){

    cv = neuralnet(formula,
                   data=standspamtrain,
                   hidden = c(3),
                   linear.output=FALSE,
                   lifesign='full',
                   lifesign.step = 5000,
                   stepmax = 1e+05)

    pred_acc = pred(cv, standspamtrain[fold$test,])

    return(pred_acc)
  })

  cv_error = mean(autoe.cverr)

  return(c(train_error, cv_error, test_error))

}

neural_net_layer2 = function() {

  spam = neuralnet(formula,
                   data=standspamtrain,
                   hidden = c(4),
                   linear.output=FALSE,
                   lifesign='full',
                   lifesign.step = 5000,
                   stepmax = 1e+05)

  pred = function(nn, dat) {
      result = compute(nn, dat[,-ncol(dat)])$net.result[,1]
      result = round(result, 0)
      return(mean(result != dat$class))
  }

  train_error = pred(spam, standspamtrain)

  test_error = pred(spam, standspamtest)

  autoe.cverr = sapply(folds, function(fold){

    cv = neuralnet(formula,
                   data=standspamtrain,
                   hidden = c(4),
                   linear.output=FALSE,
```

```
                    lifesign='full',
                    lifesign.step = 5000,
                    stepmax = 1e+05)

    pred_acc = pred(cv, standspamtrain[fold$test,])

    return(pred_acc)
  })

  cv_error = mean(autoe.cverr)

  return(c(train_error, cv_error, test_error))

}

neural_net_layer3 = function() {

  spam = neuralnet(formula,
                    data=standspamtrain,
                    hidden = c(2, 2),
                    linear.output=FALSE,
                    lifesign='full',
                    lifesign.step = 5000,
                    stepmax = 1e+08,
                    threshold = 0.05)

  pred = function(nn, dat) {
      result = compute(nn, dat[,-ncol(dat)])$net.result[,1]
      result = round(result, 0)
      return(mean(result != dat$class))
  }

  train_error = pred(spam, standspamtrain)

  test_error = pred(spam, standspamtest)

  autoe.cverr = sapply(folds, function(fold){

    cv = neuralnet(formula,
                    data=standspamtrain,
                    hidden = c(2, 2),
                    linear.output=FALSE,
                    lifesign='full',
                    lifesign.step = 5000,
                    stepmax = 1e+08,
                    threshold = 0.05)

    pred_acc = pred(cv, standspamtrain[fold$test,])

    return(pred_acc)
  })

  cv_error = mean(autoe.cverr)
```

```
  return(c(train_error, cv_error, test_error))

}

neural_net_layer4 = function() {

  spam = neuralnet(formula,
                   data=standspamtrain,
                   hidden = c(2, 3),
                   linear.output=FALSE,
                   lifesign='full',
                   lifesign.step = 5000,
                   stepmax = 1e+08,
                   threshold = 0.05)

  pred = function(nn, dat) {
      result = compute(nn, dat[,-ncol(dat)])$net.result[,1]
      result = round(result, 0)
      return(mean(result != dat$class))
  }

  train_error = pred(spam, standspamtrain)

  test_error = pred(spam, standspamtest)

  autoe.cverr = sapply(folds, function(fold){

    cv = neuralnet(formula,
                   data=standspamtrain,
                   hidden = c(2, 3),
                   linear.output=FALSE,
                   lifesign='full',
                   lifesign.step = 5000,
                   stepmax = 1e+08,
                   threshold = 0.05)

    pred_acc = pred(cv, standspamtrain[fold$test,])

    return(pred_acc)
  })

  cv_error = mean(autoe.cverr)

  return(c(train_error, cv_error, test_error))

}

nn_layer1_errors = neural_net_layer1()
nn_layer2_errors = neural_net_layer2()
nn_layer3_errors = neural_net_layer3()
nn_layer4_errors = neural_net_layer4()

nntable = matrix(0, nrow = 4, ncol = 2)
```

```
nntable[1,] = nn_layer1_errors[2:3]
nntable[2,] = nn_layer2_errors[2:3]
nntable[3,] = nn_layer3_errors[2:3]
nntable[4,] = nn_layer4_errors[2:3]

rownames(nntable) = c("1 Hidden Layer, 2 Nodes", "1 Hidden Layer, 3 Nodes", "2 Hidden Layers, [2,2] Nod

colnames(nntable) = c("Cross-Validation Error", "Test Error")

kable(nntable, caption = "Neural Net: CV and Test Error Rates for the Standardized Spam Datasets")

y_names = colnames(imbstandspamtrain19)[ncol(imbstandspamtrain19)]
x_names = colnames(imbstandspamtrain19)[-ncol(imbstandspamtrain19)]
formula = paste(paste(y_names, collapse='+'),'~',paste(x_names, collapse='+'))
folds = cv_partition(imbstandspamtrain19$class, num_folds = 5)

neural_net_layer1 = function() {

  spam = neuralnet(formula,
                   data=imbstandspamtrain19,
                   hidden = c(3),
                   linear.output=FALSE,
                   lifesign='full',
                   lifesign.step = 5000,
                   stepmax = 1e+05)

  pred = function(nn, dat) {
      result = compute(nn, dat[,-ncol(dat)])$net.result[,1]
      result = round(result, 0)
      return(mean(result != dat$class))
  }

  train_error = pred(spam, imbstandspamtrain19)

  test_error = pred(spam, imbstandspamtest19)

  autoe.cverr = sapply(folds, function(fold){

    cv = neuralnet(formula,
                   data=imbstandspamtrain19,
                   hidden = c(3),
                   linear.output=FALSE,
                   lifesign='full',
                   lifesign.step = 5000,
                   stepmax = 1e+05)

    pred_acc = pred(cv, imbstandspamtrain19[fold$test,])

    return(pred_acc)
  })

  cv_error = mean(autoe.cverr)
```

```
  return(c(train_error, cv_error, test_error))

}

neural_net_layer2 = function() {

  spam = neuralnet(formula,
                   data=imbstandspamtrain19,
                   hidden = c(4),
                   linear.output=FALSE,
                   lifesign='full',
                   lifesign.step = 5000,
                   stepmax = 1e+05)

  pred = function(nn, dat) {
      result = compute(nn, dat[,-ncol(dat)])$net.result[,1]
      result = round(result, 0)
      return(mean(result != dat$class))
  }

  train_error = pred(spam, imbstandspamtrain19)

  test_error = pred(spam, imbstandspamtest19)

  autoe.cverr = sapply(folds, function(fold){

    cv = neuralnet(formula,
                   data=imbstandspamtrain19,
                   hidden = c(4),
                   linear.output=FALSE,
                   lifesign='full',
                   lifesign.step = 5000,
                   stepmax = 1e+05)

    pred_acc = pred(cv, imbstandspamtrain19[fold$test,])

    return(pred_acc)
  })

  cv_error = mean(autoe.cverr)

  return(c(train_error, cv_error, test_error))

}

neural_net_layer3 = function() {

  spam = neuralnet(formula,
                   data=imbstandspamtrain19,
                   hidden = c(2, 2),
                   linear.output=FALSE,
                   lifesign='full',
                   lifesign.step = 5000,
```

```
                    stepmax = 1e+08,
                    threshold = 0.05)

  pred = function(nn, dat) {
      result = compute(nn, dat[,-ncol(dat)])$net.result[,1]
      result = round(result, 0)
      return(mean(result != dat$class))
  }

  train_error = pred(spam, imbstandspamtrain19)

  test_error = pred(spam, imbstandspamtest19)

  autoe.cverr = sapply(folds, function(fold){

    cv = neuralnet(formula,
                    data=imbstandspamtrain19,
                    hidden = c(2, 2),
                    linear.output=FALSE,
                    lifesign='full',
                    lifesign.step = 5000,
                    stepmax = 1e+08,
                    threshold = 0.05)

    pred_acc = pred(cv, imbstandspamtrain19[fold$test,])

    return(pred_acc)
  })

  cv_error = mean(autoe.cverr)

  return(c(train_error, cv_error, test_error))

}

neural_net_layer4 = function() {

  spam = neuralnet(formula,
                    data=imbstandspamtrain19,
                    hidden = c(2, 3),
                    linear.output=FALSE,
                    lifesign='full',
                    lifesign.step = 5000,
                    stepmax = 1e+08,
                    threshold = 0.05)

  pred = function(nn, dat) {
      result = compute(nn, dat[,-ncol(dat)])$net.result[,1]
      result = round(result, 0)
      return(mean(result != dat$class))
  }

  train_error = pred(spam, imbstandspamtrain19)
```

```
  test_error = pred(spam, imbstandspamtest19)

  autoe.cverr = sapply(folds, function(fold){

    cv = neuralnet(formula,
                   data=imbstandspamtrain19,
                   hidden = c(2, 3),
                   linear.output=FALSE,
                   lifesign='full',
                   lifesign.step = 5000,
                   stepmax = 1e+08,
                   threshold = 0.05)

    pred_acc = pred(cv, imbstandspamtrain19[fold$test,])

    return(pred_acc)
  })

  cv_error = mean(autoe.cverr)

  return(c(train_error, cv_error, test_error))

}

nn_layer1_errors = neural_net_layer1()
nn_layer2_errors = neural_net_layer2()
nn_layer3_errors = neural_net_layer3()
nn_layer4_errors = neural_net_layer4()

nntable = matrix(0, nrow = 4, ncol = 2)
nntable[1,] = nn_layer1_errors[2:3]
nntable[2,] = nn_layer2_errors[2:3]
nntable[3,] = nn_layer3_errors[2:3]
nntable[4,] = nn_layer4_errors[2:3]

rownames(nntable) = c("1 Hidden Layer, 2 Nodes", "1 Hidden Layer, 3 Nodes", "2 Hidden Layers, [2,2] Nod

colnames(nntable) = c("Cross-Validation Error", "Test Error")

kable(nntable, caption = "Neural Net: CV and Test Error Rates for the Imbalanced 1:9 Spam Dataset")

### DECISION TREES CODE SECTION ###

spamtraintree = rpart(class~., data=standspamtrain)
rpart.plot(spamtraintree, type = 4, extra = 1, clip.right.labs = F)

spamtesttree = rpart(class~., data=standspamtest)
rpart.plot(spamtesttree, type = 4, extra = 1, clip.right.labs = F)

# Choose cp using cross-validation
control = rpart.control(cp = 0.000, xxval = 100, minsplit = 2)
spamtrees = rpart(class ~ ., data = standspamtrain, control = control)
plotcp(spamtrees)
```

```
imbspamtraintree = rpart(class~., data=imbstandspamtrain19)
rpart.plot(imbspamtraintree, type = 4, extra = 1, clip.right.labs = F)

imbspamtesttree = rpart(class~., data=imbstandspamtest19)
rpart.plot(imbspamtesttree, type = 4, extra = 1, clip.right.labs = F)

# Choose cp using cross-validation
control = rpart.control(cp = 0.000, xxval = 100, minsplit = 2)
imbspamtrees = rpart(class ~ ., data = imbstandspamtrain19, control = control)
plotcp(imbspamtrees)
```