Problem\_Set\_4

Sief Salameh

5/20/2023

# Section One:

# Part A:

library(tm)

## Loading required package: NLP

library(tidyverse)

## ── Attaching packages ─────────────────────────────────────── tidyverse 1.3.1 ──

## ✔ ggplot2 3.4.2 ✔ purrr 1.0.1  
## ✔ tibble 3.2.1 ✔ dplyr 1.1.1  
## ✔ tidyr 1.3.0 ✔ stringr 1.5.0  
## ✔ readr 2.1.2 ✔ forcats 0.5.1

## Warning: package 'ggplot2' was built under R version 4.1.2

## Warning: package 'tibble' was built under R version 4.1.2

## Warning: package 'tidyr' was built under R version 4.1.2

## Warning: package 'readr' was built under R version 4.1.2

## Warning: package 'purrr' was built under R version 4.1.2

## Warning: package 'dplyr' was built under R version 4.1.2

## Warning: package 'stringr' was built under R version 4.1.2

## ── Conflicts ────────────────────────────────────────── tidyverse\_conflicts() ──  
## ✖ ggplot2::annotate() masks NLP::annotate()  
## ✖ dplyr::filter() masks stats::filter()  
## ✖ dplyr::lag() masks stats::lag()

texts <- file.path("~/Downloads/Machine\_Learning/Problem\_Set\_Four/SimpleText\_auto")  
  
docs <- VCorpus(DirSource(texts))

# Part B:

clean\_corpus <- function(corpus) {  
 corpus <- tm\_map(corpus, content\_transformer(tolower))  
  
 corpus <- tm\_map(corpus, removePunctuation)  
  
 corpus <- tm\_map(corpus, removeNumbers)  
  
 corpus <- tm\_map(corpus, removeWords, stopwords("english"))  
  
 academic\_words <- c(  
 "table", "figure", "results", "analyze",  
 "concept", "construct", "data", "define", "evidence", "framework",  
 "hypothesis", "interpret", "methodology", "perspective", "principle",  
 "quantitative", "research", "significant", "theory", "variable",  
 "validity", "conclude", "critique", "figure", "model", "analysis", "can",  
 "since", "therefore", "first", "state", "within", "use", "using", "similar",  
 "used", "shown", "shows", "however", "give", "given", "also", "compared",  
 "found", "fig", "two", "present", "well", "may", "time", "different", "one",  
 "study", "show", "will", "due", "thus", "let", "see", "number", "based",  
 "set", "left"  
 )  
  
 corpus <- tm\_map(corpus, removeWords, academic\_words)  
  
 corpus <- tm\_map(corpus, stripWhitespace)  
  
 return(corpus)  
}  
  
cleaned\_docs <- clean\_corpus(docs)

**In Part B, I transformed all the words to lowercase to help standardize the document. Then, I removed all punctuation, numbers, and stopwords. Additionally, I created a set of custom words to filter out. I repeatedly ran the code and generated word clouds to visualize the non-necessary words that were appearing. I would then take those words and further filter them out through a cyclical process. These words are common academic jargon that, in my opinion, do not contribute to distinguishing the stem topic. Lastly, I removed all whitespace and executed the function to clean the words in the documents.**

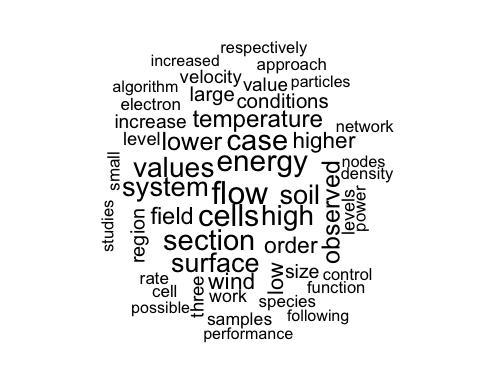
# Part C:

library(wordcloud)

## Loading required package: RColorBrewer

## Warning: package 'RColorBrewer' was built under R version 4.1.2

word\_matrix <- DocumentTermMatrix(cleaned\_docs)  
  
freq\_words <- findFreqTerms(word\_matrix, 50)  
  
freq\_table <- data.frame(  
 word = freq\_words,  
 freq = colSums(as.matrix(word\_matrix[, freq\_words]))  
)  
  
freq\_table <- freq\_table[order(freq\_table$freq, decreasing = TRUE), ]  
  
wordcloud(freq\_table$word, freq\_table$freq, scale = c(2, 0.1), max.words = 50, random.order = FALSE, rot.per = 0.2)



freq\_table %>% head(50)

## word freq  
## flow flow 893  
## cells cells 853  
## energy energy 830  
## section section 808  
## case case 806  
## values values 781  
## high high 774  
## system system 758  
## soil soil 753  
## surface surface 744  
## lower lower 696  
## observed observed 691  
## temperature temperature 665  
## field field 651  
## order order 627  
## wind wind 624  
## higher higher 615  
## low low 594  
## conditions conditions 556  
## large large 551  
## region region 537  
## size size 529  
## increase increase 516  
## value value 508  
## three three 500  
## velocity velocity 499  
## level level 493  
## levels levels 485  
## work work 478  
## samples samples 469  
## electron electron 454  
## rate rate 454  
## small small 454  
## cell cell 452  
## species species 452  
## power power 451  
## approach approach 450  
## function function 443  
## control control 441  
## density density 441  
## network network 441  
## studies studies 441  
## respectively respectively 437  
## nodes nodes 427  
## particles particles 426  
## possible possible 426  
## algorithm algorithm 424  
## performance performance 424  
## following following 422  
## increased increased 421

# Part D:

library(topicmodels)

## Warning: package 'topicmodels' was built under R version 4.1.2

k\_values <- c(2, 3, 5, 8, 10)  
  
for (k in k\_values) {  
 lda\_model <- LDA(word\_matrix, k)  
 cat("LDA Model with k = ", k, "\n")  
  
 top\_freq\_words <- terms(lda\_model, 10)  
  
 for (i in 1:k) {  
 cat("Topic", i, ": ")  
 cat(paste(top\_freq\_words[i, ], collapse = ", "))  
 cat("\n")  
 }  
  
 cat("------------------------\n\n")  
}

## LDA Model with k = 2   
## Topic 1 : cells, flow  
## Topic 2 : system, energy  
## ------------------------  
##   
## LDA Model with k = 3   
## Topic 1 : energy, cells, soil  
## Topic 2 : temperature, flow, nodes  
## Topic 3 : surface, cell, wind  
## ------------------------  
##   
## LDA Model with k = 5   
## Topic 1 : energy, cells, wind, soil, case  
## Topic 2 : electron, cell, turbine, algorithm, order  
## Topic 3 : flow, expression, particles, nodes, temperature  
## Topic 4 : electrons, gene, high, memory, theorem  
## Topic 5 : field, human, power, algorithms, section  
## ------------------------  
##   
## LDA Model with k = 8   
## Topic 1 : surface, temperature, cells, memory, network, cells, electron, flow  
## Topic 2 : film, particles, neurons, clusters, nodes, surface, energy, wind  
## Topic 3 : temperature, energy, expression, cluster, tephra, cell, electrons, soil  
## Topic 4 : flows, atmosphere, mitochondrial, nodes, connectivity, culture, field, depth  
## Topic 5 : lddos, heating, cell, algorithm, area, tinnitus, turbine, beach  
## Topic 6 : approach, solar, levels, system, cost, human, theorem, surface  
## Topic 7 : layer, observations, mutation, program, cafrep, hipscs, soil, concentration  
## Topic 8 : samples, thermosphere, patients, module, scaling, rpe, power, velocity  
## ------------------------  
##   
## LDA Model with k = 10   
## Topic 1 : nodes, particles, surface, turbine, energy, area, energy, clusters, soil, cells  
## Topic 2 : maximal, temperature, film, power, electron, network, region, cluster, root, cell  
## Topic 3 : order, wind, layer, algorithm, approach, soil, electron, flow, elements, expression  
## Topic 4 : module, epoxy, mitochondrial, performance, electrons, values, plasma, beach, values, gene  
## Topic 5 : node, atmosphere, rpe, turbines, flow, tephra, field, coherent, operators, genes  
## Topic 6 : system, observations, mtdna, memory, flows, flow, observed, pressure, soils, culture  
## Topic 7 : output, polymer, devices, speed, lddos, scaling, thermosphere, dependence, concentration, human  
## Topic 8 : clock, models, sediment, algorithms, feature, cost, wake, program, theorem, neurons  
## Topic 9 : input, heating, samples, work, features, vertical, electrons, surface, roots, differentiation  
## Topic 10 : lemma, particle, high, java, normal, equations, density, sets, case, growth  
## ------------------------

**LDA Model with k = 2:**

**Topic 1: This topic indicates a correlation with biological and ecological terms, such as flow and cells.**

**Topic 2: This topic includes words like systems and energy, representing a potential correlation with numerical or quantitative terminology related to energy.**

**LDA Model with k = 3:**

**Topic 1: This topic mentions words such as soil, energy, and cells, possibly indicating a focus on biological systems or dynamics.**

**Topic 2: This topic involves words like temperature, flow, and nodes, suggesting a correlation with thermal science and case studies related to energy.**

**Topic 3: This topic includes words such as surface, cell, and wind, which indicates a correlation related to topics such as experimental measurements, sections in scientific papers, and electron-related phenomena.**

**LDA Model with k = 5:**

**Topic 1: This topic includes words like energy, cells, wind, soil, and case which could indicate a relationship to energy systems, cellular biology, and mathematical theorems.**

**Topic 2: This topic involves words like electron, cell, turbine, algorithm, and order suggesting a correlation to topics such as fluid flow, cellular biology, and mathematical lemmas.**

**Topic 3: This topic includes words such as flow, expression, particles, nodes, and temperature indicating a potential focus on scientific topics related to network systems, human factors, and ordered structures.**

**Topic 4: This topic involves words like electrons, gene, high, memory, and theorem suggesting a focus on topics such as temperature effects, gene expression, plant biology, and performance evaluation.**

**Topic 5: This topic includes words such as field, human, power, algorithms, and section which might correlate to topic areas related to fluid velocity, cultural aspects, modular systems, and maximum values.**

**LDA models with topic sizes 8 and 10 both yield similar topic selections, and the distribution of words is identical to that of smaller LDA sizes. They do not generate unique or distinct word patterns that help distinguish the topic theme or subject matter.**

**Based on my opinion, a k-size of 5 yields the most meaningful coherence for each topic. This is because k-sizes 2 and 3 are too small and do not reveal significant information regarding word meaning or the context of word distribution. On the other hand, k-sizes 8 and 10 provide too much information and overwhelm the reader with noisy words. Thus, a k-size of 5 is an effective measure to distinguish between each topic theme and the quantity of words that each topic encompasses.**

# Part E:

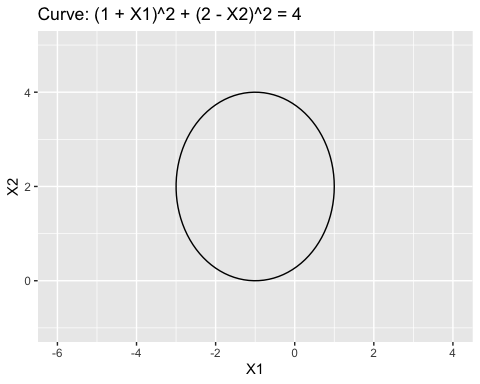
set.seed(123)  
  
k\_values <- c(2, 3, 5, 8, 10)  
fold <- 10  
  
best\_model <- NULL  
  
best\_perplexity <- Inf  
  
for (k in k\_values) {  
 fold\_perplexities <- c()  
  
 for (i in 1:fold) {  
  
 # Creating training and testing sets  
  
 set.seed(123)  
 train\_indices <- sample(seq\_len(nrow(word\_matrix)),  
 size = floor((fold - 1) / fold \* nrow(word\_matrix)), replace = FALSE  
 )  
  
 training\_data <- word\_matrix[train\_indices, ]  
 test\_data <- word\_matrix[-train\_indices, ]  
  
 # Training the LDA model  
  
 lda\_model.cv <- LDA(training\_data, k)  
  
 # Calculating the perplexity on test set  
  
 perplexity <- perplexity(lda\_model.cv, newdata = test\_data)  
 fold\_perplexities <- c(fold\_perplexities, perplexity)  
 }  
  
 avg\_perplexity <- mean(fold\_perplexities)  
  
 if (avg\_perplexity < best\_perplexity) {  
 best\_perplexity <- avg\_perplexity  
 best\_model <- LDA(word\_matrix, k)  
 }  
}  
  
# Presenting the topics from the best model  
  
cat("Best LDA Model:\n")  
cat("Number of Topics (k):", best\_model$k, "\n")  
cat("Perplexity:", best\_perplexity, "\n")  
  
top\_freq\_words <- terms(best\_model, 10)  
  
for (i in 1:best\_model$k) {  
 cat("Topic", i, ": ")  
 cat(paste(top\_freq\_words[i, ], collapse = ", "))  
 cat("\n")  
}

**This code was to computationally extensive and took too long to run in R. Therefore, I did not provide the output. I did include the code however, to demonstrate the different steps in achieving the optimization of the hyperparameters of the LDA model using 10-fold cross-validation.**

# Section Two

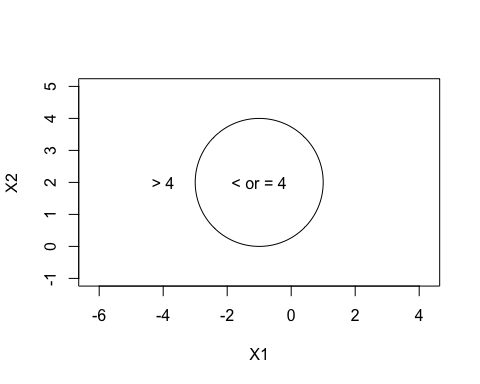
# Part A:

library(ggplot2)  
  
theta <- seq(0, 2 \* pi, length.out = 100)  
X1 <- -1 + 2 \* cos(theta)  
X2 <- 2 + 2 \* sin(theta)  
  
plot\_one <- data.frame(X1, X2)  
  
ggplot(plot\_one, aes(X1, X2)) +  
 geom\_path() +  
 xlim(-6, 4) +  
 ylim(-1, 5) +  
 xlab("X1") +  
 ylab("X2") +  
 ggtitle("Curve: (1 + X1)^2 + (2 - X2)^2 = 4")



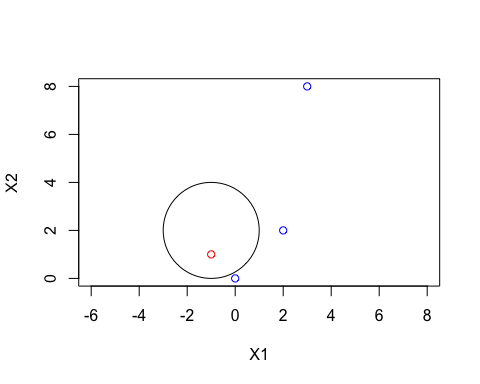
# Part B:

plot(NA, NA,  
 type = "n", xlim = c(-6, 4), ylim = c(-1, 5), asp = 1,  
 xlab = "X1", ylab = "X2"  
)  
  
symbols(c(-1), c(2), circles = c(2), add = TRUE, inches = FALSE)  
  
text(c(-1), c(2), "< or = 4")  
text(c(-4), c(2), "> 4")



# Part C:

plot(c(0, -1, 2, 3), c(0, 1, 2, 8),  
 col = c("blue", "red", "blue", "blue"),  
 type = "p", asp = 1, xlab = "X1", ylab = "X2"  
)  
symbols(c(-1), c(2), circles = c(2), add = TRUE, inches = FALSE)



(0, 0) would be classified blue since it is outside the circle.

(-1, 1) would be classified red since it is inside the circle.  
(2, 2) would be classified blue since it is outside of the circle.

(3, 8) would be classified blue since it is outside of the circle.

# Part D:

**By expanding the equation of the decision boundary, (1 + X1)² + (2 - X2)² = 4, we can obtain:**

**(1 + X1)² + (2 - X2)² = 4**

**1 + 2X1 + X1² + 4 - 4X2 + X2² = 4**

**X1² + X2² + 2X1 - 4X2 + 1 = 0**

**This is a quadratic equation in terms of X1 and X2. However, when we introduce the additional variables Y1 = X1² and Y2 = X2², we can rewrite the equation as:**

**Y1 + Y2 + 2X1 - 4X2 + 1 = 0**

**When we expand the equation to this form, it becomes linear in terms of X1, X1², X2, and X2². The linear coefficients are 2 and -4, multiplying X1 and X2, respectively, while Y1 and Y2 serve as additional terms.**

# Section Three:

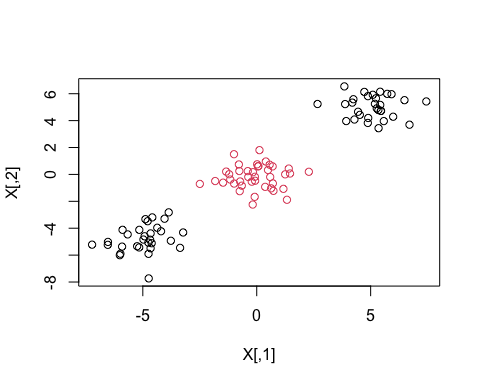
library(e1071)

## Warning: package 'e1071' was built under R version 4.1.2

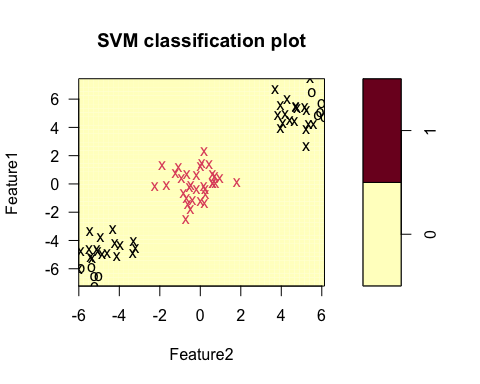
library(mlbench)

## Warning: package 'mlbench' was built under R version 4.1.2

set.seed(321)  
  
shift <- 5  
  
X <- matrix(rnorm(100 \* 2), ncol = 2)  
  
X[1:30, ] <- X[1:30, ] + shift  
  
X[31:60, ] <- X[31:60, ] - shift  
  
y <- c(rep(0, 60), rep(1, 40))  
  
simulated\_data <- data.frame(Feature1 = X[, 1], Feature2 = X[, 2], Class = as.factor(y))  
  
plot(X, col = y + 1)



train\_function <- sample(100, 80)  
  
training\_data <- simulated\_data[train\_function, ]  
  
test\_data <- simulated\_data[-train\_function, ]  
  
# Fitting the data with a Support Vector Classifier  
  
svm\_linear <- svm(Class ~ .,  
 data = training\_data, kernel = "linear",  
 scale = FALSE  
)  
  
plot(svm\_linear, data = training\_data)



summary(svm\_linear)

##   
## Call:  
## svm(formula = Class ~ ., data = training\_data, kernel = "linear",   
## scale = FALSE)  
##   
##   
## Parameters:  
## SVM-Type: C-classification   
## SVM-Kernel: linear   
## cost: 1   
##   
## Number of Support Vectors: 70  
##   
## ( 33 37 )  
##   
##   
## Number of Classes: 2   
##   
## Levels:   
## 0 1

table(predicted = svm\_linear$fitted, truth = training\_data$Class)

## truth  
## predicted 0 1  
## 0 47 33  
## 1 0 0

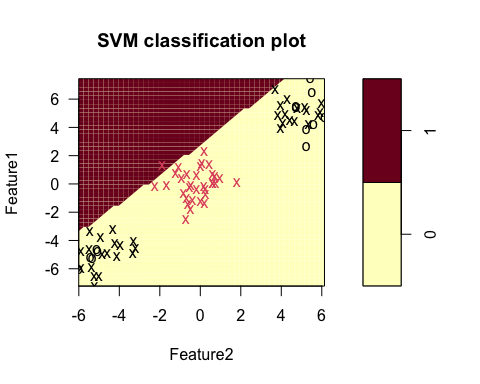
Error rate = Number of incorrect predictions / Total number of predictions

33/80 = .4125 or 41.25%

The support vector classifier labels all training points as class zero, indicating its inability to distinguish between the different classes in the training set. Consequently, it assigns all instances to class zero, disregarding their actual class labels.

This observation implies that the support vector classifier fails to capture the underlying patterns and characteristics of the data. It suggests that the classifier struggles to identify an appropriate decision boundary or effectively separate the classes. As a result, the model is deemed ineffective on this specific training set as it fails to provide meaningful predictions or accurate classification.

# Fitting the data with polynomial kernel  
  
svm\_poly <- svm(Class ~ .,  
 data = training\_data, kernel = "polynomial",  
 scale = FALSE  
)  
  
plot(svm\_poly, data = training\_data)



table(predicted = svm\_poly$fitted, truth = training\_data$Class)

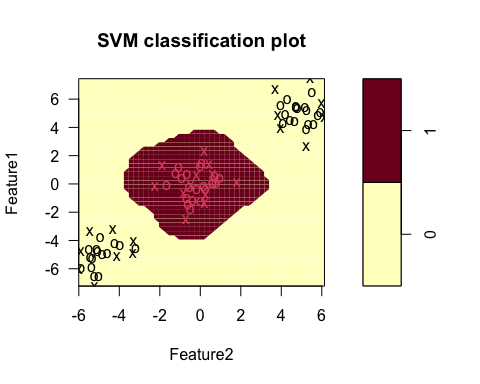
## truth  
## predicted 0 1  
## 0 47 32  
## 1 0 1

Error rate = Number of incorrect predictions / Total number of predictions

32/80 = .40 or 40%

The error rate corrects for one observation using the polynomial kernel model

# Fitting the data with radial kernel  
  
svm\_radial <- svm(Class ~ .,  
 data = training\_data, kernel = "radial",  
 scale = FALSE  
)  
  
plot(svm\_radial, data = training\_data)



table(predicted = svm\_radial$fitted, truth = training\_data$Class)

## truth  
## predicted 0 1  
## 0 47 0  
## 1 0 33

Error rate = Number of incorrect predictions / Total number of predictions

0/80 = 0%

The error rate for the radial kernel model is much lower than the svm and polynomial models. In fact, the error rate for the radial kernel is 0.

# Comparing the test errors for the 3 kernels:  
  
linear\_pred <- predict(svm\_linear, test\_data)  
  
table(predicted = linear\_pred, truth = test\_data$Class)

## truth  
## predicted 0 1  
## 0 13 7  
## 1 0 0

Error rate = Number of incorrect predictions / Total number of predictions

7/20 = .35 or 35%

poly\_pred <- predict(svm\_poly, test\_data)  
  
table(predicted = poly\_pred, truth = test\_data$Class)

## truth  
## predicted 0 1  
## 0 12 7  
## 1 1 0

Error rate = Number of incorrect predictions / Total number of predictions

8/20 = .4 or 40%

radial\_pred <- predict(svm\_radial, test\_data)  
  
table(predicted = radial\_pred, truth = test\_data$Class)

## truth  
## predicted 0 1  
## 0 13 0  
## 1 0 7

Error rate = Number of incorrect predictions / Total number of predictions

0/20 = 0%

**The Radial Kernel model generates lower error rates for both the training data and the test data.**

# Section Four

# Part A:

library(ISLR)  
  
data(Auto)  
  
median\_mpg <- median(Auto$mpg)  
  
Auto$mpg\_binary <- ifelse(Auto$mpg > median\_mpg, 1, 0)  
  
Auto$mpg\_binary <- as.factor(Auto$mpg\_binary)

# 

# Part B:

library(e1071)  
  
set.seed(10)  
  
cost\_tune <- tune(svm, mpg\_binary ~ ., data = Auto, kernel = "linear", ranges = list(cost = c(.001, 0.01, 0.1, 1, 10, 100, 1000)))  
  
summary(cost\_tune)

##   
## Parameter tuning of 'svm':  
##   
## - sampling method: 10-fold cross validation   
##   
## - best parameters:  
## cost  
## 1  
##   
## - best performance: 0.007628205   
##   
## - Detailed performance results:  
## cost error dispersion  
## 1 1e-03 0.091730769 0.03990003  
## 2 1e-02 0.073974359 0.03067293  
## 3 1e-01 0.050897436 0.03153169  
## 4 1e+00 0.007628205 0.01228382  
## 5 1e+01 0.020384615 0.02019157  
## 6 1e+02 0.033205128 0.01737168  
## 7 1e+03 0.033205128 0.01737168

**In the linear model, the lowest error of 0.007628205 was achieved when the cost parameter was set to 1, indicating that this parameter value yielded the best performance for the support vector (SVM) machine. The dispersion represents the variability or spread of errors across different cost values.**

# Part C:

# For the radial basis kernel (rbf):  
  
set.seed(09)  
  
cost\_tune\_rbf <- tune(svm, mpg\_binary ~ .,  
 data = Auto, kernel = "radial",  
 ranges = list(  
 cost = c(.001, 0.01, 0.1, 1, 10),  
 gamma = c(0.01, 0.1, 1, 5, 10, 100)  
 )  
)  
summary(cost\_tune\_rbf)

##   
## Parameter tuning of 'svm':  
##   
## - sampling method: 10-fold cross validation   
##   
## - best parameters:  
## cost gamma  
## 10 0.01  
##   
## - best performance: 0.02301282   
##   
## - Detailed performance results:  
## cost gamma error dispersion  
## 1 1e-03 1e-02 0.51096154 0.13709174  
## 2 1e-02 1e-02 0.51096154 0.13709174  
## 3 1e-01 1e-02 0.08929487 0.05558189  
## 4 1e+00 1e-02 0.07147436 0.04312562  
## 5 1e+01 1e-02 0.02301282 0.02549182  
## 6 1e-03 1e-01 0.51096154 0.13709174  
## 7 1e-02 1e-01 0.16839744 0.08975786  
## 8 1e-01 1e-01 0.07653846 0.05100638  
## 9 1e+00 1e-01 0.05615385 0.04294734  
## 10 1e+01 1e-01 0.02551282 0.02076457  
## 11 1e-03 1e+00 0.51096154 0.13709174  
## 12 1e-02 1e+00 0.51096154 0.13709174  
## 13 1e-01 1e+00 0.51096154 0.13709174  
## 14 1e+00 1e+00 0.06128205 0.05274896  
## 15 1e+01 1e+00 0.06134615 0.05006757  
## 16 1e-03 5e+00 0.54596154 0.03201429  
## 17 1e-02 5e+00 0.54596154 0.03201429  
## 18 1e-01 5e+00 0.54596154 0.03201429  
## 19 1e+00 5e+00 0.48205128 0.04602804  
## 20 1e+01 5e+00 0.47692308 0.05114030  
## 21 1e-03 1e+01 0.54596154 0.03201429  
## 22 1e-02 1e+01 0.54596154 0.03201429  
## 23 1e-01 1e+01 0.54596154 0.03201429  
## 24 1e+00 1e+01 0.50506410 0.03662576  
## 25 1e+01 1e+01 0.49743590 0.03803299  
## 26 1e-03 1e+02 0.54846154 0.02632839  
## 27 1e-02 1e+02 0.54846154 0.02632839  
## 28 1e-01 1e+02 0.54846154 0.02632839  
## 29 1e+00 1e+02 0.54846154 0.02632839  
## 30 1e+01 1e+02 0.54846154 0.02632839

# For the polynomial basis kernel (poly):  
  
set.seed(08)  
  
cost\_tune\_poly <- tune(svm, mpg\_binary ~ ., data = Auto, kernel = "polynomial", ranges = list(cost = c(.001, 0.01, 0.1, 1, 10), degree = c(1, 2, 3, 4, 5)))  
  
summary(cost\_tune\_poly)

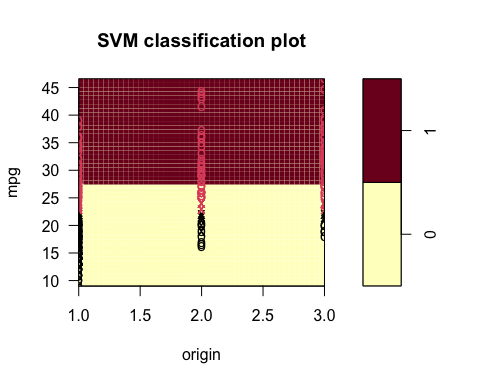
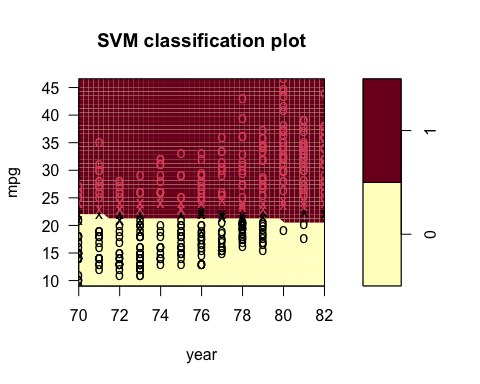
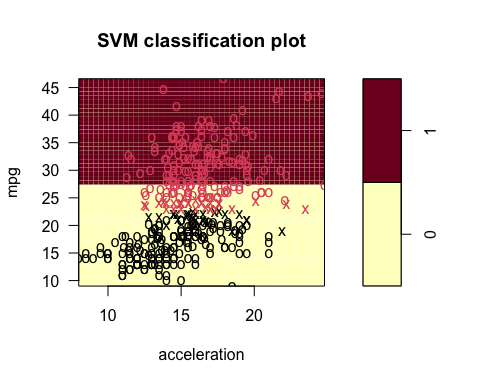
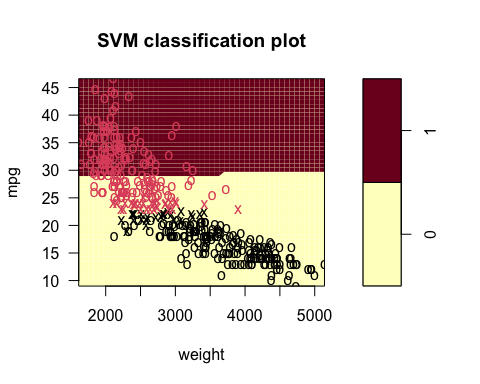
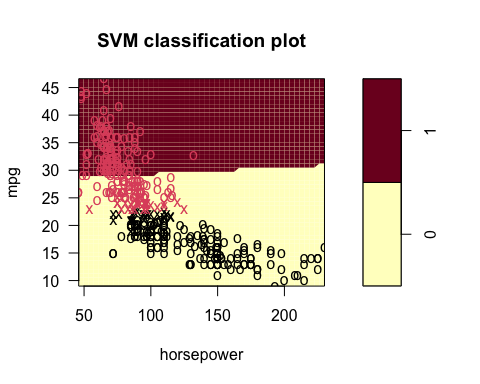
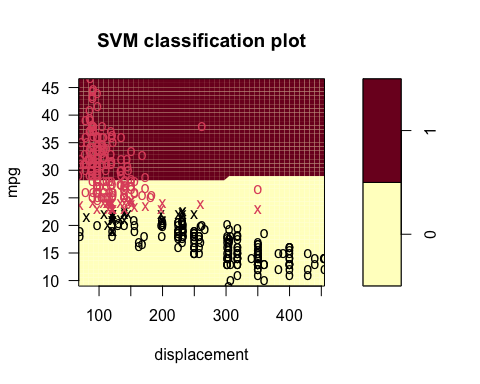
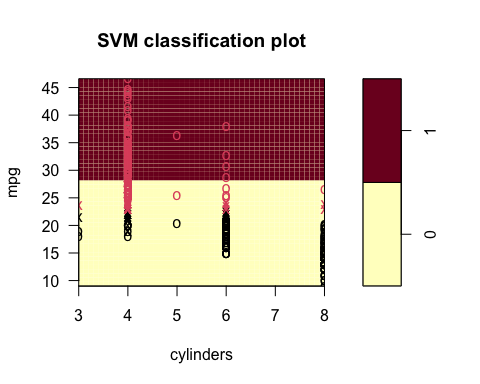
##   
## Parameter tuning of 'svm':  
##   
## - sampling method: 10-fold cross validation   
##   
## - best parameters:  
## cost degree  
## 10 1  
##   
## - best performance: 0.06371795   
##   
## - Detailed performance results:  
## cost degree error dispersion  
## 1 1e-03 1 0.56378205 0.05004495  
## 2 1e-02 1 0.56378205 0.05004495  
## 3 1e-01 1 0.16525641 0.10934527  
## 4 1e+00 1 0.08160256 0.03553445  
## 5 1e+01 1 0.06371795 0.04355118  
## 6 1e-03 2 0.56378205 0.05004495  
## 7 1e-02 2 0.56378205 0.05004495  
## 8 1e-01 2 0.56378205 0.05004495  
## 9 1e+00 2 0.56378205 0.05004495  
## 10 1e+01 2 0.50224359 0.12621940  
## 11 1e-03 3 0.56378205 0.05004495  
## 12 1e-02 3 0.56378205 0.05004495  
## 13 1e-01 3 0.56378205 0.05004495  
## 14 1e+00 3 0.56378205 0.05004495  
## 15 1e+01 3 0.56378205 0.05004495  
## 16 1e-03 4 0.56378205 0.05004495  
## 17 1e-02 4 0.56378205 0.05004495  
## 18 1e-01 4 0.56378205 0.05004495  
## 19 1e+00 4 0.56378205 0.05004495  
## 20 1e+01 4 0.56378205 0.05004495  
## 21 1e-03 5 0.56378205 0.05004495  
## 22 1e-02 5 0.56378205 0.05004495  
## 23 1e-01 5 0.56378205 0.05004495  
## 24 1e+00 5 0.56378205 0.05004495  
## 25 1e+01 5 0.56378205 0.05004495

**The parameter tuning process for the radial basis kernel model suggests that the SVM model performs best when the cost parameter is set to 10 and the gamma parameter is set to 0.01. This yields the lowest error of 0.02301282. The dispersion represents the variability or spread of errors across different cost and gamma values. It appears that the parameter combinations with lower values of cost and gamma generally result in higher error rates.**

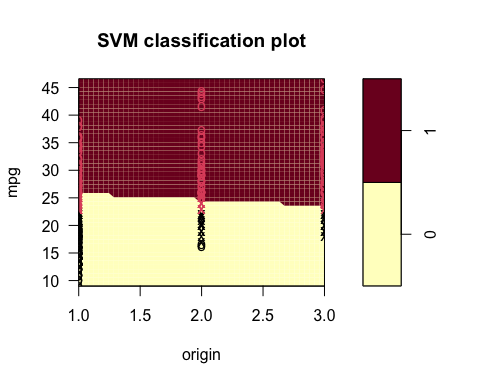
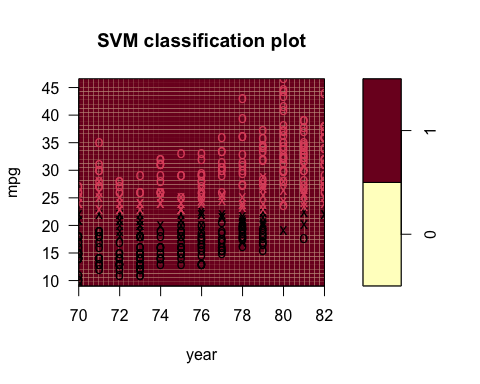
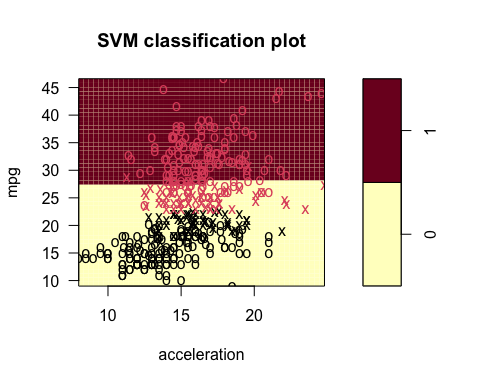
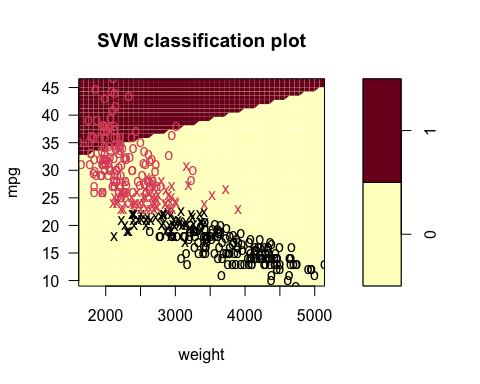
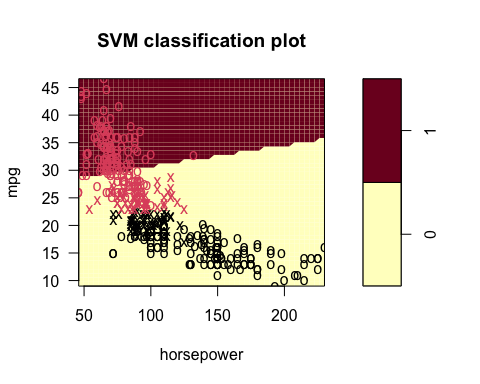
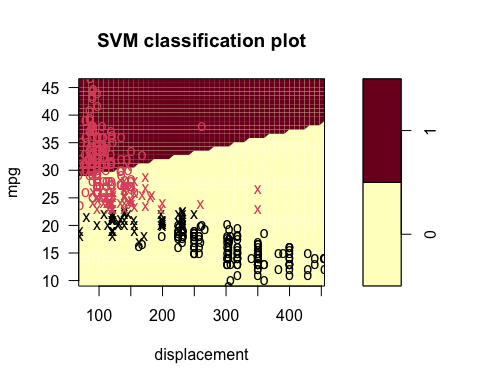
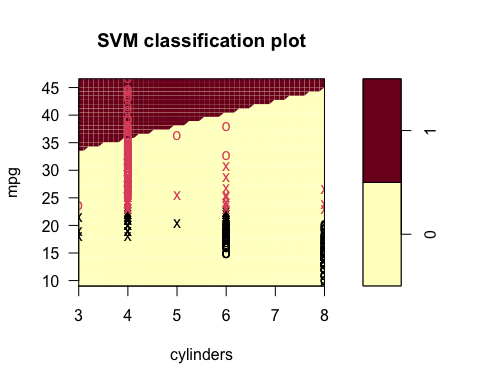
**The parameter tuning process for the polynomial basis kernel model suggests that the SVM model performs best when the cost parameter is set to 10 and the degree parameter is set to 1. This yields the lowest error of 0.06371795. The dispersion represents the variability or spread of errors across different cost and degree values.**

# Part D:

svm\_linear <- svm(mpg\_binary ~ ., data = Auto, kernel = "linear", cost = 1)  
  
svm\_poly <- svm(mpg\_binary ~ .,  
 data = Auto, kernel = "polynomial", cost = 10,  
 degree = 1  
)  
  
svm\_radial <- svm(mpg\_binary ~ .,  
 data = Auto, kernel = "radial", cost = 10,  
 gamma = 0.01  
)  
  
plotpairs <- function(fit) {  
 for (name in names(Auto)[!(names(Auto) %in% c("mpg", "mpg\_binary", "name"))])  
 {  
 plot(fit, Auto, as.formula(paste("mpg~", name, sep = "")))  
 }  
}  
  
plotpairs(svm\_linear)



plotpairs(svm\_poly)



plotpairs(svm\_radial)

