DSA1101 Midterm Notes

Contents

1 R

1.1 Data Types

1.1.1 Vectors

```
  c(1,2,3,4,5)
```

```
## [1] 1 2 3 4 5
```

```
1:5
```

```
## [1] 1 2 3 4 5
```

```
seq(1,9,2)
```

```
## [1] 1 3 5 7 9
```

1.1.2 Matrices

```
  matrix(1:6, nrow = 2, ncol = 3, byrow = TRUE)
```

```
##      [,1] [,2] [,3]
## [1,]    1    2    3
## [2,]    4    5    6
```

1.1.3 Data Frames

```
  data.frame(
    id = 1:3,
    name = c('Tom', 'Mary', 'Peter'),
    age = c(26,30,25),
    marital_status = c('married','divorced','single'),
    stringsAsFactors = TRUE
  )
```

```
##   id  name age marital_status
## 1  1   Tom  26        married
## 2  2  Mary  30       divorced
## 3  3 Peter  25         single
```

1.2 Logical Vectors

```
  random_permutation_one_to_ten <- sample(1:10, 10, replace=FALSE)
random_permutation_one_to_ten
```

```
##  [1] 10  1  9  5  6  8  7  3  2  4
```

```
random_permutation_one_to_ten > 5
```

```
##  [1]  TRUE FALSE  TRUE FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
```

1.3 Logical Operators

A	B	A **AND** B	A **OR** B
TRUE	TRUE	TRUE	TRUE
TRUE	FALSE	FALSE	TRUE
FALSE	TRUE	FALSE	TRUE
FALSE	FALSE	FALSE	FALSE

A	**NOT** A
TRUE	FALSE
FALSE	TRUE

1.3.1 Logical Operators in R

Operator	Description
&	Element-wise AND
|	Element-wise OR
&&	First element AND
||	First element OR
!	NOT

1.4 Conditionals

1.4.1 Conditional Statements

```
  x <- 10
if (x > 20) {
  print('x is bigger than 20')
} else if (x > 10) {
  print('x is bigger than 10')
} else {
  print('x is smaller than or equal to 10')
}

## [1] "x is smaller than or equal to 10"
```

1.5 Reading CSV Files

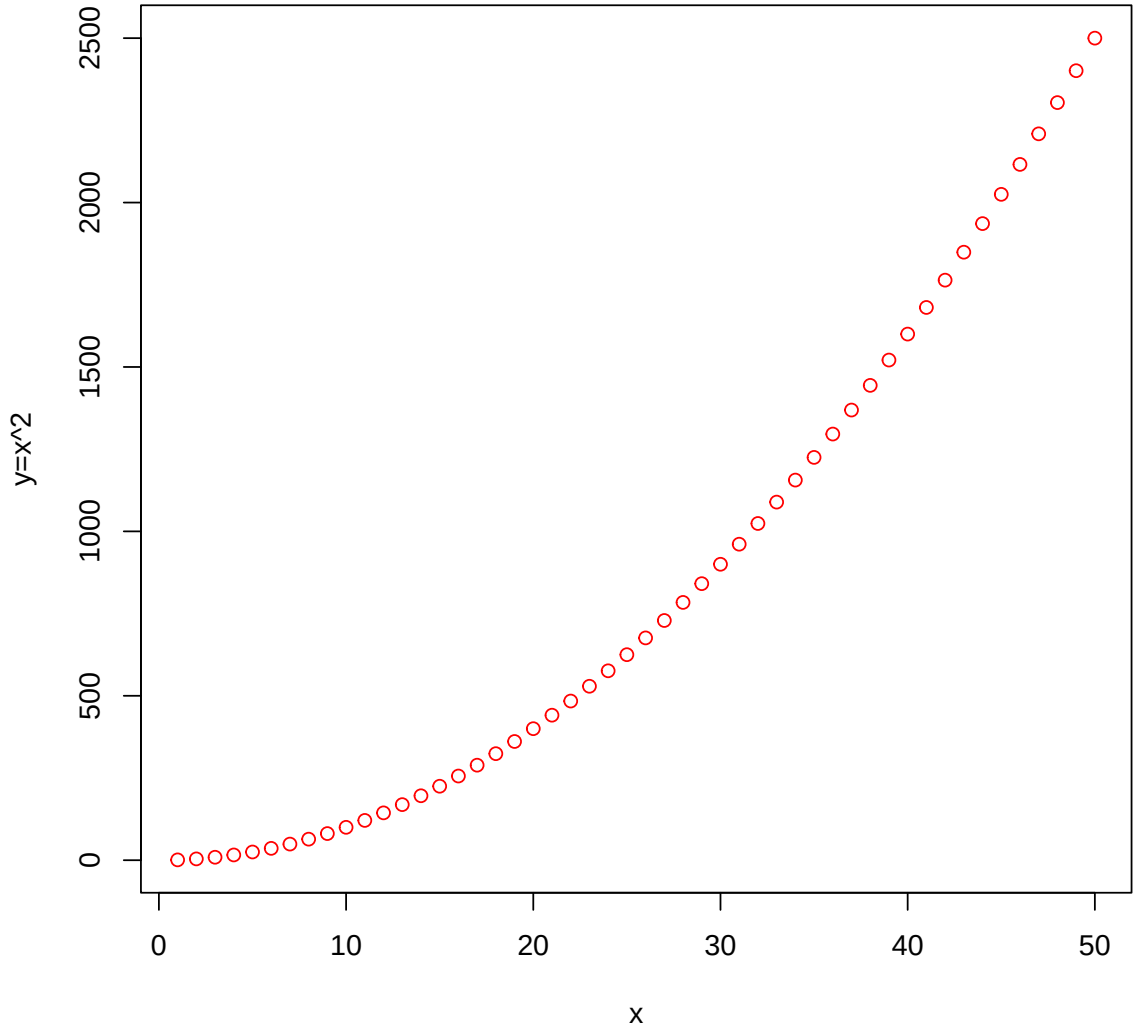
```
  read.csv('data.csv', header = TRUE, sep = ',', dec = '.',
           stringsAsFactors = TRUE)

##   id   name age occupation
## 1  1  James  28    Fireman
## 2  2 Evelyn  27 Technician
## 3  3  Laura  34    Teacher
```

1.6 Data Visualisation

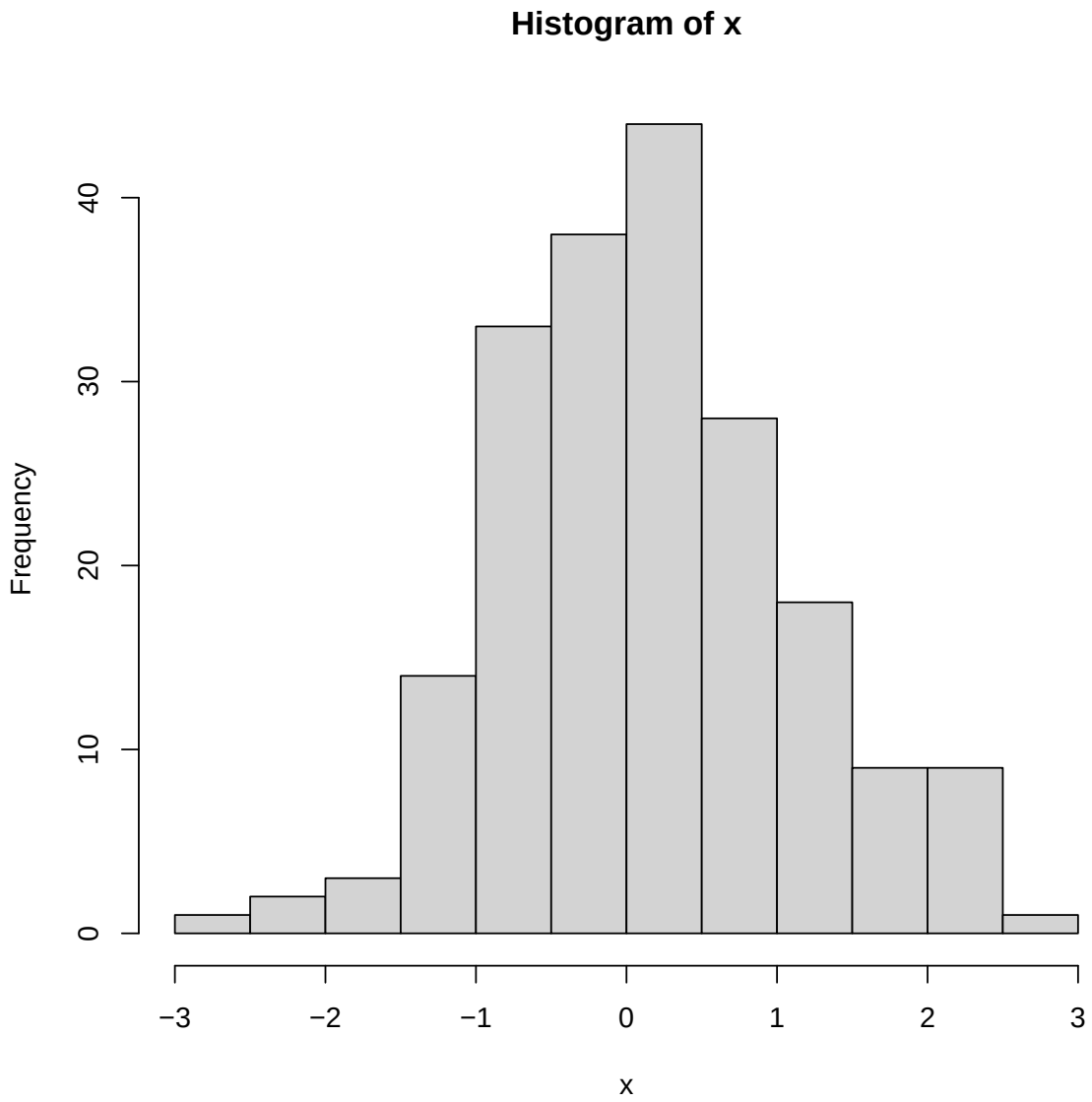
1.6.1 Scatter Plots

```
  x <- 1:50
y <- x^2
plot(x = x, y = y, xlab = 'x', ylab = 'y=x^2', col = 'red')
```



1.6.2 Histogram

```r
  n <- 200
x <- rnorm(n)
hist(x = x, breaks = ceiling(sqrt(n)), col = 'lightgray')
```



1.7 Iteration

1.7.1 For Loop

```r
  for (i in 1:5) {
    print(i)
  }
```

```
## [1] 1
## [1] 2
## [1] 3
## [1] 4
## [1] 5
```

1.7.2 While Loop

```
  i = 1
while (i <= 5) {
  print(i)
  i <- i + 1
}

## [1] 1
## [1] 2
## [1] 3
## [1] 4
## [1] 5
```

1.7.3 Repeat Loop

```
  i = 1
repeat {
  print(i)
  i <- i + 1
  if (i == 6) break
}

## [1] 1
## [1] 2
## [1] 3
## [1] 4
## [1] 5
```

2 Statistical Measures

2.1 Mean

$$\text{mean}(\mathbf{x}) = \overline{x} = \frac{1}{N}\sum_{i=1}^{N} x_i$$

2.2 Median

$$\text{median}(\mathbf{x}) = \begin{cases} x_{(N+1)/2} & \text{if } N \text{ is odd} \\ \frac{x_{N/2}+x_{N/2+1}}{2} & \text{if } N \text{ is even} \end{cases}$$

2.3 Sample Variance

$$\text{var}(\mathbf{x}) = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \overline{x})^2$$

2.4 Sample Standard Deviation

$$\text{sd}(\mathbf{x}) = \sqrt{\text{var}(\mathbf{x})}$$

2.5 Sample Covariance

$$\text{cov}(\mathbf{x}, \mathbf{y}) = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \overline{x})(y_i - \overline{y})$$

2.6 Sample Correlation Coefficient

$$\text{cor}(\mathbf{x}, \mathbf{y}) = r_{xy} = \frac{\text{cov}(\mathbf{x}, \mathbf{y})}{\text{sd}(\mathbf{x})\,\text{sd}(\mathbf{y})}$$

2.7 Location and Scale Changes to Statistical Measures

Statistical Measure	**Location Changes** $\mathbf{x} + b,\ \mathbf{y} + c$	**Scale Changes** $a\mathbf{x},\ d\mathbf{y}$
mean	variant $\text{mean}(\mathbf{x}) + b$	variant $a \cdot \text{mean}(\mathbf{x})$
median	variant	variant
var	invariant $\text{var}(\mathbf{x})$	variant $a^2 \cdot \text{var}(\mathbf{x})$
sd	invariant $\text{sd}(\mathbf{x})$	variant $\lvert a\rvert \cdot \text{sd}(\mathbf{x})$
cov	invariant $\text{cov}(\mathbf{x}, \mathbf{y})$	variant
cor	invariant $\text{cor}(\mathbf{x}, \mathbf{y})$	invariant $\text{cor}(\mathbf{x}, \mathbf{y})$ if $ad >$ 0 else if $ad < 0$ then $-\text{cor}(\mathbf{x}, \mathbf{y})$

3 Diagnostics of Classifiers

3.1 Confusion Matrix

		Predicted Class	
		Positive	Negative
Actual Class	Positive	True Positive (TP)	False Negative (FN)
	Negative	False Positive (FP)	True Negative (TN)

3.2 Accuracy

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

3.3 True Positive Rate

$$\text{TPR} = \frac{TP}{TP + FN}$$

3.4 False Positive Rate / Type I Error Rate

$$\text{FPR} = \frac{FP}{FP + TN}$$

3.5 False Negative Rate / Type II Error Rate

$$\text{FNR} = \frac{FN}{TP + FN}$$

3.6 True Negative Rate

$$\text{TNR} = \frac{TN}{TN + FP}$$

3.7 Precision

$$\text{Precision} = \frac{TP}{TP + FP}$$

3.7.1 Remarks

1. Precision is useful when costly actions will be followed up on the data predicted to be positive,
2. because precision gives the proportion of actual positives among those predicted to be positive
3. For example, if an insurance company wants to predict potential customers interested in purchasing insurance, and the cost to try to sell an insurance to a potential customer is non-trivial (e.g. insurance agent has to house visit the customer).

3.8 N-Fold Cross Validation

3.8.1 Algorithm

1. The entire dataset is randomly split into N datasets of approximately equal size.
2. $N-1$ of these datasets are treated as the training dataset, while the remaining one is the test dataset. A measure of the model error is obtained.
3. This process is repeated across the various combinations of N datasets taken $N-1$ at a time.
4. The observed N models errors are averaged across the N folds

3.9 ROC Curve (TPR vs FPR Trade-off)

1. Graph of True Positive Rate (TPR) against False Positive Rate (FPR)
2. As TPR increases, FPR tend to increase as well
   (a) Increasing TPR may be a double-edged sword
3. TPR = FPR = 0 means binary classifier classifies everything as negative
4. TPR = FPR = 1 means binary classifier classifies everything as positive

3.10 Bias-Variance tradeoff

1. error = bias^2 + variance + irreducible error
2. As variance increases, bias decreases, and vice versa

3.11 Calculation Intensive Exam Question & Solution

Midterm Q6. Consider the following confusion matrix for a classifier

		Predicted Class	
		Positive	Negative
Actual Class	Positive	20	75
	Negative	140	55

The false negative rate (FNR) of the classifier is _____ (round to 3 decimal places).

Solution

1. Copy paste the following code:

```r
gcmfv <- function(tp, fn, fp, tn) {
  # generates confusion matrix from values
  return (matrix(c(tp, fn, fp, tn), nrow = 2, ncol = 2, byrow = TRUE))
}

tp <- function(m) {
  # true positive from confusion matrix m
  return (m[1, 1])
}

fn <- function(m) {
  # false negative from confusion matrix m
  return (m[1, 2])
}

fp <- function(m) {
  # false positive from confusion matrix m
  return (m[2, 1])
}

tn <- function(m) {
  # true negative from confusion matrix m
  return (m[2, 2])
}

accuracy <- function(m) {
  return ((tp(m)+tn(m))/(tp(m)+tn(m)+fp(m)+fn(m)))
}

tpr <- function(m) {
  return (tp(m)/(tp(m)+fn(m)))
}

fpr <- function(m) {
  return (fp(m)/(fp(m)+tn(m)))
}

fnr <- function(m) {
```

```
  return (fn(m)/(fn(m)+tp(m)))
}

tnr <- function(m) {
  return (tn(m)/(tn(m)+fp(m)))
}

precision <- function(m) {
  return (tp(m)/(tp(m)+fp(m)))
}
```

2. Create Confusion Matrix

```
confusion.matrix <- gcmfv(tp=20, fn=75, fp=140, tn=55)
```

3. Get the metric you need

```
fnr(confusion.matrix)

## [1] 0.7894737
```

4 Supervised Learning

4.1 K-nearest Neigbours

4.1.1 Description

1. Given training set of size M, N feature values, and 1 binary outcome (0 or 1), we have a table of feature values x_{ij} and a vector of outcome y_i for $1 \leq i \leq M,\ 1 \leq j \leq N$
2. Given any test point x^*, with N feature values x_j^*, for $1 \leq j \leq N$, calculate euclidean distance of x^* to each training point x_i, i.e. for $1 \leq i \leq M$, $\text{distance}_i = \sqrt{\sum_{j=1}^{N}(x_{ij} - x_j^*)^2}$
3. Given chosen value k, the k-nearest neighbours/training points x_i to x^*, denoted $N_k(x^*)$, is the set of the first k x_i in the sequence of x_i sorted by increasing distance_i
4. $\hat{Y}(x^*) = \frac{1}{k} \sum_{x_i \in N_k(x^*)} y_i$

5. The predicted outcome for $x^* = y^* = \begin{cases} 1 & \hat{Y} > \sigma \\ 0 & \hat{Y} < \sigma \end{cases}$ where σ is the threshold. $\sigma = 0.5$ in the majority rule.

4.1.2 Choice of σ

1. As σ increases,
   (a) TP, TPR, FP, and FPR decreases or stays the same
   (b) TN, TNR, FN, and FNR increases or stays the same
2. As σ decreases,
   (a) TP, TPR, FP, and FPR increases or stays the same
   (b) TN, TNR, FN, and FNR decreases or stays the same
3. See Diagnostics of Classifiers ROC Curve for more info

4.1.3 Choice of k

1. when k increases, the variance decreases, but bias increases
2. when k decreases, the variance increases, but bias decreases
3. See Diagnostics of Classifiers Bias-Variance Tradeoff for more info

4.1.4 Prediction Surface

1. Boundaries can be curvy
2. Can be not axis-aligned

4.1.5 Standardising Variables

1. Any variable with a larger scale than others will have a larger effect on the Euclidean distance
2. To prevent this problem, we can standardise our data so that all our variables will have mean of zero and standard deviation of one with the scale function in R

```r
data <- cbind(
    1:5,
    seq(100, 500, 100),
    sample(0:1, 5, replace = TRUE)
)
data
```

```
##      [,1] [,2] [,3]
## [1,]    1  100    1
## [2,]    2  200    0
## [3,]    3  300    1
## [4,]    4  400    0
## [5,]    5  500    1
```

```r
data[, 1:2] = scale(data[, 1:2])
data
```

```
##            [,1]       [,2] [,3]
## [1,] -1.2649111 -1.2649111    1
## [2,] -0.6324555 -0.6324555    0
## [3,]  0.0000000  0.0000000    1
## [4,]  0.6324555  0.6324555    0
## [5,]  1.2649111  1.2649111    1
```

4.1.6 R Implementation

```r
library(class)
data <- matrix(
    c(1, 1, 0,
      1, 2, 0,
      2, 1, 0,
      2, 2, 0,
      2, 3, 0,
      8, 8, 1,
      9, 8, 1,
      8, 7, 1,
      9, 9, 1,
      9, 7, 1), nrow = 10, byrow = TRUE
```

```
)
train <- sample(1:10, 5, replace = FALSE)
train.x <- data[train, 1:2]
train.y <- data[train, 3]
cbind(train.x, train.y)

##          train.y
## [1,] 2 3       0
## [2,] 8 8       1
## [3,] 1 1       0
## [4,] 1 2       0
## [5,] 9 8       1

test.x <- data[-train, 1:2]
test.y <- data[-train, 3]
cbind(test.x, test.y)

##          test.y
## [1,] 2 1      0
## [2,] 2 2      0
## [3,] 8 7      1
## [4,] 9 9      1
## [5,] 9 7      1

knn.pred <- knn(train.x, test.x, train.y, k=3)
confusion.matrix <- table(test.y, knn.pred)
confusion.matrix

##       knn.pred
## test.y 0 1
##      0 2 0
##      1 0 3
```

4.1.7 Calculation Intensive Exam Questions & Solutions

4.1.7.1 Euclidean Distances, $\hat{Y}$, and Prediction for 1 Test Data Point

Adapted from Midterm Q17-18. Suppose we have a training set of 5 data points with binary value outcome = c(1,1,0,1,0), x_1 = c(1,2,1,3,3), and x_2 = c(3,2,1,3,1). Using the 3-nearest neighbors classifier and the majority, what is the **fitted outcome value** $\hat{Y}$ and the **predicted outcome value** of (x_1^*,x_2^*) = (2,4)?

Solution

1. Copy paste the following code:

```
distance <- function(m, t) {
  # returns numbered table of euclidean distance of t to each
  # training point in m
  # each column in m is feature variable except last column is
  # outcome y
  if (length(t) != ncol(m)-1) {
    print("test data does not match number of feature variables")
    return
  }
  cd <- function(p1, p2) {
    return (sqrt(sum((p1-p2)^2)))
  }

  table <- data.frame(id = 1:nrow(m))
  colnames(m) <- c(paste("x_", 1:(ncol(m)-1), sep = ""), "y")
  table <- cbind(table, m)
  dist <- rep(1, times<-nrow(m))
  for (r in 1:nrow(m)) {
    dist[r] <- cd(m[r, 1:(ncol(m)-1)], t)
  }
  table <- cbind(table, dist)
  return (table)
}

sorts <- function(d) {
  # sorts the table output of distance function in increasing
  # euclidean distance
  return (d[order(d[, ncol(d)]), ])
}

y_hat <- function(s, k) {
 # calculate y-hat from table output of sorts function given value of k
 return (sum(s[1:k, ncol(s)-1])/k)
}
```

```
predict <- function(y, s) {
 # return predicted class given y-hat y and threshold value s (sigma)
 # assumes y !<- s, i.e. no tie
 return (if (y > s) 1 else 0)
}
```

2. Calculate Euclidean Distances to (2, 4)

```
dist_matrix <- distance(matrix(
  c(1,2,1,3,3,
    3,2,1,3,1,
    1,1,0,1,0), nrow = 5, byrow = FALSE
), c(2, 4))
dist_matrix
```

```
##   id x_1 x_2 y     dist
## 1  1   1   3 1 1.414214
## 2  2   2   2 1 2.000000
## 3  3   1   1 0 3.162278
## 4  4   3   3 1 1.414214
## 5  5   3   1 0 3.162278
```

3. Sort By Increasing Euclidean Distances

```
sorted_dist_matrix <- sorts(dist_matrix)
sorted_dist_matrix
```

```
##   id x_1 x_2 y     dist
## 1  1   1   3 1 1.414214
## 4  4   3   3 1 1.414214
## 2  2   2   2 1 2.000000
## 3  3   1   1 0 3.162278
## 5  5   3   1 0 3.162278
```

4. Calculate **fitted outcome value** $\hat{Y}$ based on value of k=3

```
y_hat_value <- y_hat(sorted_dist_matrix, k=3)
y_hat_value

## [1] 1
```

5. Calculate **predicted outcome value** based on majority rule $\sigma = 0.5$

```
predicted_value <- predict(y_hat_value, s=0.5)
predicted_value

## [1] 1
```

4.1.7.2 $\hat{Y}$ to Confusion Matrix for n Test Data Points

Adapted from Midterm Q14. Suppose we have k-nearest neighbours classifier for binary outcome Y. The table below shows the actual and fitted outcome for $n = 10$ test data points.

Actual Y	1	1	0	1	1	0	0	0	1	0
$\hat{Y}$	0.9	0.8	0.8	0.6	0.5	0.5	0.5	0.3	0.2	0.1

What is the True Positive Rate (TPR) when we predict $Y = 1$ if $\sigma > 0.7$

Solution

1. Copy paste the following code

```
predict <- function(y, s) {
  # return predicted class given y-hat y and threshold value s (sigma)
  # assumes y !<- s, i.e. no tie
  return (ifelse(y > s, 1, 0))
}
```

2. Obtain predictions for $\sigma = 0.7$

```
predictions <- predict(
  c(0.9, 0.8, 0.8, 0.6, 0.5, 0.5, 0.5, 0.3, 0.2, 0.1), s=0.7
)
predictions

##  [1] 1 1 1 0 0 0 0 0 0 0
```

3. Generate Confusion Matrix

```
table(c(1, 1, 0, 1, 1, 0, 0, 0, 1, 0), predictions)

##    predictions
##     0 1
##   0 4 1
##   1 3 2
```

4. Refer to Section 3.11 for metric calculations

4.2 Decision Tree

4.2.1 Graph

1. A graph consists of nodes (circles) and edges (lines) connecting the nodes.
2. A walk is a sequence of edges which joins a sequence of nodes
3. A trail is a walk where all edges are distinct
4. A cycle is a trail in which the only repeated nodes are the first and last nodes
5. An acyclic graph has no cycles

4.2.2 Tree

1. A tree is an acyclic graph.
2. A rooted tree has a root node.
3. Depth of node in a rooted tree = distance of node from root node
   (a) depth of root node = 0

4.2.3 Decision Tree

1. Is a rooted tree

4.2.4 Entropy

Given a outcome variable Y, with possible outcomes $y_1, y_2, \ldots, y_n$ which occur with purity / probability $P(y_1), P(y_2), \ldots, P(y_n)$, the entropy of Y is defined as:

$$D(Y) = -\sum_{i=1}^{n} P(y_i) \log_2 P(y_i)$$

4.2.5 Conditional Entropy

Given a feature variable X, with split outcome x_1, x_2 which occur with probability $P(x_1), P(x_2)$, the conditional entropy of Y given X is defined as:

$$D(Y|X) = \sum_{i=1}^{2} P(x_i) D(Y|X = x_i)$$

4.2.6 Decision Tree Algorithm: Entropy

1. Start at root node
2. Check for termination conditions, if any, e.g.:
   (a) Minimum purity threshold reached
   (b) Tree cannot be further split with the preset minimum purity threshold.
   (c) Any other stopping criterion is satisfied (such as the maximum depth of the tree).
3. Calculate entropy for current node (base entropy)
4. For each feature variable, for each split outcome, calculate conditional entropy.
5. Choose the feature variable and split outcome with the highest entropy reduction = base entropy - conditional entropy. Branch the current node by this choice.
6. Repeat Step 2-5 for each of the two branched nodes.

4.2.7 Gini Index

Given a outcome variable Y, with possible outcomes $y_1, y_2, \ldots, y_n$ which occur with probability $P(y_1), P(y_2), \ldots, P(y_n)$, the Gini index of Y is defined as:

$$G(Y) = \sum_{i=1}^{n} P(y_i)(1 - P(y_i))$$

4.2.8 Conditional Gini Index

Given a feature variable X, with split outcome x_1, x_2 which occur with probability $P(x_1), P(x_2)$, the conditional Gini index of Y given X is defined as:

$$G(Y|X) = \sum_{i=1}^{2} P(x_i) G(Y|X = x_i)$$

4.2.9 Decision Tree Algorithm: Gini Index

1. Same as Decision Tree Algorithm for Entropy but replace Entropy with Gini Index.

4.2.10 Complexity Parameter C_p

1. Smaller values of C_p correspond to decision trees of larger sizes
2. Larger values of C_p correspond to decision trees of smaller sizes

4.2.11 Prediction Surface

1. Rectangular surfaces
2. Can only be axis-aligned

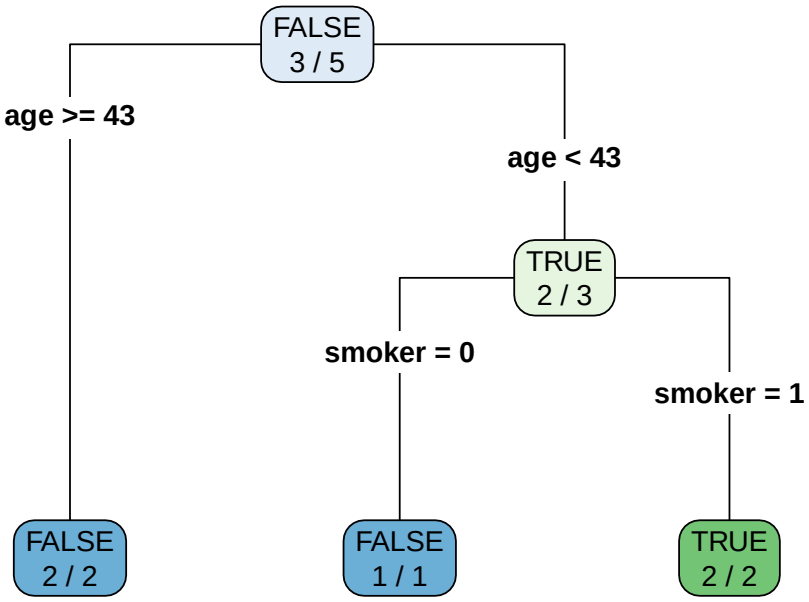
4.2.12 R Implementation

```
library(rpart)
library(rpart.plot)
data <- data.frame(
  id = 1:5,
  gender = c('M', 'M', 'F', 'M', 'F'),
  age = c(21, 33, 40, 60, 45),
  smoker = c(TRUE, FALSE, TRUE, TRUE, FALSE),
  bmi = c(22, 25, 28, 24, 26),
  diabetes = c(TRUE, FALSE, TRUE, FALSE, FALSE),
  stringsAsFactors = TRUE
)
data
```

```
##   id gender age smoker bmi diabetes
## 1  1      M  21   TRUE  22     TRUE
## 2  2      M  33  FALSE  25    FALSE
## 3  3      F  40   TRUE  28     TRUE
## 4  4      M  60   TRUE  24    FALSE
## 5  5      F  45  FALSE  26    FALSE
```

```
fit <- rpart(
  diabetes ~ gender + age + smoker + bmi,
  method = 'class',
  data = data,
  control = rpart.control(minsplit=1),
  parms = list(split = 'information')
)
```

```
rpart.plot(fit, type = 4, extra = 2, clip.right.labs = FALSE, varlen = 0,
           faclen = 0)
```



4.2.13 Calculation Intensive Exam Questions & Solutions

4.2.13.1 Entropy involving n outcomes

Adapted from Midterm Q2. Let X be the outcome variable with $n = 2$ possible outcomes, which occur with purity c(0.5, 0.5). Calculate the entropy of X.

Solution.

1. Copy paste the following code

```r
entropy <- function(prob) {
  sum <- 0
  for (p in prob) {
    sum <- sum + p * log2(p)
  }
  return (-sum)
}
```

2. Calculate entropy

```r
entropy(c(0.5, 0.5))

## [1] 1
```

4.2.13.2 Gini Index involving n outcomes

Adapted from Midterm Q28. Let X be the outcome variable with $n = 2$ possible outcomes, which occur with purity c(1490/2201, 1-1490/2201). Calculate the Gini index of X.

Solution.

1. Copy paste the following code

```r
gini_index <- function(prob) {
  sum <- 0
  for (p in prob) {
    sum <- sum + p * (1-p)
  }
  return (sum)
}
```

2. Calculate Gini index

```r
gini_index(c(1490/2201, 1-1490/2201))

## [1] 0.4373668
```

4.3 Naive Bayes

4.3.1 Probability Laws

4.3.1.1 Bayes' Theorem

$$P(Y|X) = \frac{P(Y \cap X)}{P(X)} = \frac{P(X|Y) \times P(Y)}{P(X)}$$

4.3.1.2 Law of total probability

$$P(A) = P(A \cap B) + P(A \cap \neg B)$$

4.3.2 Naive Bayes

Suppose the categorical outcome variable Y takes on values in the set $\{y_1, y_2, \ldots, y_k\}$ and there are m feature variables $X_1, X_2, \ldots, X_m$. By Bayes Theorem, for $j = 1, 2, \ldots, k$,

$$\begin{aligned} &P(Y = y_j | X_1 = x_1, X_2 = x_2, \ldots, X_m = x_m) \\ &= \frac{P(X_1 = x_1, X_2 = x_2, \ldots, X_m = x_m | Y = y_j) \times P(Y = y_j)}{P(X_1 = x_1, X_2 = x_2, \ldots, X_m = x_m)} \end{aligned}$$

4.3.2.1 Assume Conditional Independence

$$\begin{aligned} &P(X_1 = x_1, X_2 = x_2, \ldots, X_m = x_m | Y = y_j) \\ &= P(X_1 = x_1 | Y = y_j) P(X_2 = x_2 | Y = y_j) \ldots P(X_m = x_m | Y = y_j) \\ &= \prod_{i=1}^{m} P(X_i = x_i | Y = y_j) \end{aligned}$$

4.3.2.2 Ignore Denominator

$$P(X_1 = x_1, X_2 = x_2, \ldots, X_m = x_m)$$

4.3.2.3 Finally

For $j = 1, 2, \ldots, k$,

$$\begin{aligned} &P(Y = y_j | X_1 = x_1, X_2 = x_2, \ldots, X_m = x_m) \\ &\propto P(Y = y_j) \times \prod_{i=1}^{m} P(X_i = x_i | Y = y_j) \end{aligned}$$

4.3.3 Numerical Underflow

To prevent probability scores from becoming too small to be accurately stored in a computer, we can take logarithm on both sides,

$$\log P(Y = y_j | X_1 = x_1, X_2 = x_2, \ldots, X_m = x_m)$$

$$\propto \log P(Y = y_j) + \sum_{i=1}^{m} \log P(X_i = x_i | Y = y_j)$$

4.3.4 R Implementation

```
library(e1071)
data <- data.frame(
  watch_lectures = c(TRUE, TRUE, FALSE, TRUE, FALSE, TRUE),
  does_tutorials = c(TRUE, FALSE, TRUE, FALSE, FALSE, TRUE),
  prior_exp = c(FALSE, FALSE, FALSE, TRUE, TRUE, FALSE),
  grades_for_dsa1101 = c('A-', 'B+', 'C', 'B+', 'B-', 'A'),
  stringsAsFactors = TRUE
)
train <- data[1:4,]
test <- data[5:6,]
model <- naiveBayes(grades_for_dsa1101 ~ watch_lectures + does_tutorials
                    + prior_exp, train, laplace = 0)
predict(model, test)

## [1] B+ A-
## Levels: A A- B- B+ C
```

4.3.5 Calculation Intensive Exam Questions & Solutions

```
table_to_naiveBayes <- function(features, feature_categories, )

## Error:  <text>:1:63:  unexpected ')'
## 1:  table_to_naiveBayes <- function(features, feature_categories, )
##                                                                   ^
```

4.4 Linear & Logistic Regression

4.4.1 Solving Simultaneous Equations

1. Use solve function in R

5 Unsupervised Learning

6 Big Data Techniques

6.1 MapReduce