Binary Classification

Defining true/false positives

In general, Positive = identified and negative = rejected. Therefore:

- True positive = correctly identified
- False positive = incorrectly identified
- True negative = correctly rejected
- False negative = incorrectly rejected

Medical testing example:

- True positive = Sick people correctly diagnosed as sick
- False positive= Healthy people incorrectly identified as sick
- True negative = Healthy people correctly identified as healthy
- False negative = Sick people incorrectly identified as healthy.

Steps in building a prediction

1. Find the right data
2. Define your error rate
3. Split data into:
    - Training Set
    - Testing Set
    - Validation Set(optional)
4. On the training set pick features
5. On the training set pick prediction function
6. On the training set cross-validate
7. If no validation - apply 1x to test set
8. If validation - apply to test set and refine
9. If validation - apply 1x to validation

Question 1

Which of the following (pick one) is **NOT** a step in building a prediction model?

Options

(i) Defining the error rate

(ii) Picking features

(iii) Obtaining the right data

(iv) Estimating test set accuracy with training-set accuracy.

(v) (N0) Selecting features with the test set.

Cross Validation

Bias Variance Trade-off *http://scott.fortmann-roe.com/docs/BiasVariance.html*

- In a prediction problem, a model is usually given a dataset of known data on which training is run (*training dataset*), and a dataset of unknown data (or *first seen data/ testing dataset*) against which testing the model is performed.
- Cross-validation is mainly used in settings where the goal is prediction, and one wants to estimate how accurately a predictive model will perform in practice.
- The goal of cross validation is to define a dataset to "test" the model in the training phase (i.e., the validation dataset), in order to limit problems like overfitting, give an insight on how the model will generalize to an independent data set (i.e., an unknown dataset, for instance from a real problem), etc.
- Cross-validation is important in guarding against testing hypotheses suggested by the data (called "Type III errors"), especially where further samples are hazardous, costly or impossible to collect

K-fold cross validation

- In k-fold cross-validation, the original data set is randomly partitioned into k equal size subsamples.
- Of the k subsamples, a single subsample is retained as the validation data for testing the model, and the remaining k - 1 subsamples are used as training data.
- The cross-validation process is then repeated k times (the folds), with each of the k subsamples used exactly once as the validation data.
- The k results from the folds can then be averaged (or otherwise combined) to produce a single estimation.

- The advantage of this method over repeated random sub-sampling is that all observations are used for both training and validation, and each observation is used for validation exactly once.

Choosing K - Bias and Variance

In general, when using k-fold cross validation, it seems to be the case that:

- A larger k will produce an estimate with smaller bias but potentially higher variance (on top of being computationally expensive)
- A smaller k will lead to a smaller variance but may lead to a a biased estimate.

Leave-One-Out Cross-Validation

- As the name suggests, leave-one-out cross-validation (LOOCV) involves using a single observation from the original sample as the validation data, and the remaining observations as the training data.
- This is repeated such that each observation in the sample is used once as the validation data.
- This is the same as a K-fold cross-validation with K being equal to the number of observations in the original sampling, i.e. **K=n**.

Question 2

- **Part A** If K is small in a K-fold cross validation is the bias in the estimate of out-of-sample (test set) accuracy smaller or bigger?
- **Part B** If K is small is the variance in the estimate of out-of-sample (**test set**) accuracy smaller or bigger.
- **Part C** Is K large or small in **leave one out** cross validation?

Options

(i) The bias is larger and the variance is smaller. Under leave one out cross validation K is equal to the sample size.

(ii) (No) The bias is larger and the variance is smaller. Under leave one out cross validation K is ***equal to one***.

(iii) (No) The bias is smaller and the variance is bigger. Under leave one out cross validation K is equal to one.

(iv) The bias is smaller and the variance is bigger. Under leave one out cross validation K is equal to the sample size,

Logistic Regression

Logistic regression is useful when you are predicting a binary outcome from a set of continuous predictor variables.

```
# Logistic Regression
# where F is a binary factor and
# x1-x3 are continuous predictors

fit <- glm(F~x1+x2+x3,data=mydata,family=binomial())

summary(fit) # display results

confint(fit) # 95% CI for the coefficients

exp(coef(fit)) # exponentiated coefficients

exp(confint(fit)) # 95% CI for exponentiated coefficients

predict(fit, type="response") # predicted values

residuals(fit, type="deviance") # residuals
```

Type III Errors

- Type III error is related to hypotheses suggested by the data, if tested using the data set that suggested them, are likely to be accepted even when they are not true.
- This is because circular reasoning would be involved: something seems true in the limited data set, therefore we hypothesize that it is true in general, therefore we (wrongly) test it on the same limited data set, which seems to confirm that it is true.
- Generating hypotheses based on data already observed, in the absence of testing them on new data, is referred to as post hoc theorizing.
- The correct procedure is to test any hypothesis on a data set that was not used to generate the hypothesis.

Definitions

Accuracy Rate

The accuracy rate calculates the proportion ofobservations being allocated to the **correct** group by the predictive model. It is calculated as follows:

$$\frac{\text{Number of Correct Classifications}}{\text{Total Number of Classifications}}$$

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

Misclassification Rate

The misclassification rate calculates the proportion ofobservations being allocated to the **incorrect** group by the predictive model. It is calculated as follows:

$$\frac{\text{Number of Incorrect Classifications}}{\text{Total Number of Classifications}}$$

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

Question 3

Load the South Africa Heart Disease Data and create training and test sets with the following code:

```
install.packages("ElemStatLearn")
library(ElemStatLearn)
data(SAheart)

set.seed(8484)
train = sample(1:dim(SAheart)[1],
    size=dim(SAheart)[1]/2,replace=F)
trainSA = SAheart[train,]
testSA = SAheart[-train,]
```

Then fit a logistic regression model with *Coronary Heart Disease* (chd) as the outcome and *age at onset, current alcohol consumption, obesity levels, cumulative tabacco, type-A behavior*, and *low density lipoprotein cholesterol* as predictors.
Calculate the misclassification rate for your model using this function and a prediction on the "response" scale:

```
missClass = function(values,prediction){
  sum(((prediction > 0.5)*1) != values)/length(values)
  }
```

What is the misclassification rate on the training set? What is the misclassification rate on the test set?

```
head(SAheart)

lr1 <- glm(chd ~ age + alcohol + obesity +
      tobacco + typea + ldl, data=trainSA,
      family="binomial")

lr1.train.predict <- predict(lr1, type="response")

missclass.lr1.train <- missClass(trainSA$chd,
    lr1.train.predict)

lr1.test.predict <- predict(lr1, newdata=testSA,
   type="response")

missclass.lr1.test <- missClass(testSA$chd,
   lr1.test.predict)
```

```
> table(floor(2*lr1.test.predict),testSA$chd)

     0   1
  0 117  34
  1  38  42

> table(floor(2*lr1.train.predict),trainSA$chd)

     0   1
  0 124  40
```

```
  1  23  44

> missclass.lr1.test
[1] 0.3116883
>
> missclass.lr1.train
[1] 0.2727273
```

Question 4

Load the olive oil data using the commands:

```
install.packages("pgmm")
library(pgmm)
data(olive)
olive = olive[,-1]
```

These data contain information on 572 different Italian olive oils from multiple regions in Italy.
(Areas: (1) North Apulia, (2) Calabria, (3) South Apulia, (4) Sicily, (5) Inland Sardinia, (6) Coastal Sardinia, (7) East Liguria, (8) West Liguria, and (9) Umbria)

```
> table(olive$Area)

  1   2   3   4   5   6   7   8   9
 25  56 206  36  65  33  50  50  51
```

Fit a classification tree where **Area** is the outcome variable. Then predict the value of area for the following data frame using the tree command with all defaults.

```
library(tree)
head(olive)
```

```
> head(olive)
  Area Palmitic Palmitoleic Stearic Oleic Linoleic Linolenic
1    1     1075          75     226  7823      672        36
2    1     1088          73     224  7709      781        31
3    1      911          54     246  8113      549        31
4    1      966          57     240  7952      619        50
5    1     1051          67     259  7771      672        50
6    1      911          49     268  7924      678        51
  Arachidic Eicosenoic
1        60         29
2        61         29
3        63         29
4        78         35
5        80         46
6        70         44
```

The following code shows how to fit a regression tree using the `tree()` command. Area is the outcome variable, using all the other variables as predictor variables.

```
olive.tree <- tree(Area ~ Palmitic +
     Palmitoleic + Stearic + Oleic + Linoleic +
     Linolenic + Arachidic + Eicosenoic,
     data=olive)
```

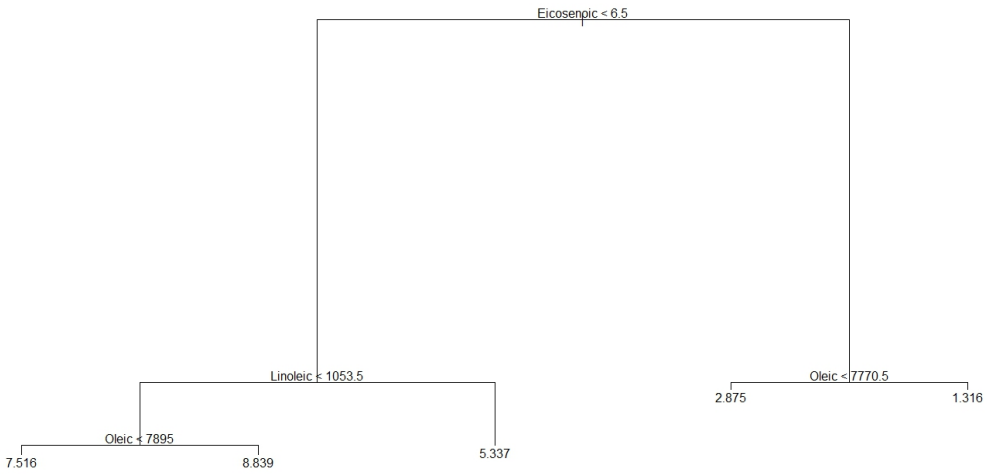

Figure 0.1:

```
plot(olive.tree)
text(olive.tree)

newdata = as.data.frame(t(colMeans(olive)))

predict(olive.tree, newdata)
```

Answer

2.875. It is strange because Region should be a qualitative variable - but tree is reporting the average value of Region as a numeric variable in the leaf predicted for newdata.

Question 5

Suppose that I fit and prune a tree to get the following diagram. What area would I predict for a new value of:

```
olive.tree <- tree(as.factor(Area) ~ Palmitic +
         Palmitoleic + Stearic + Oleic + Linoleic +
         Linolenic + Arachidic + Eicosenoic, data=olive)

plot(olive.tree); text(olive.tree)
```

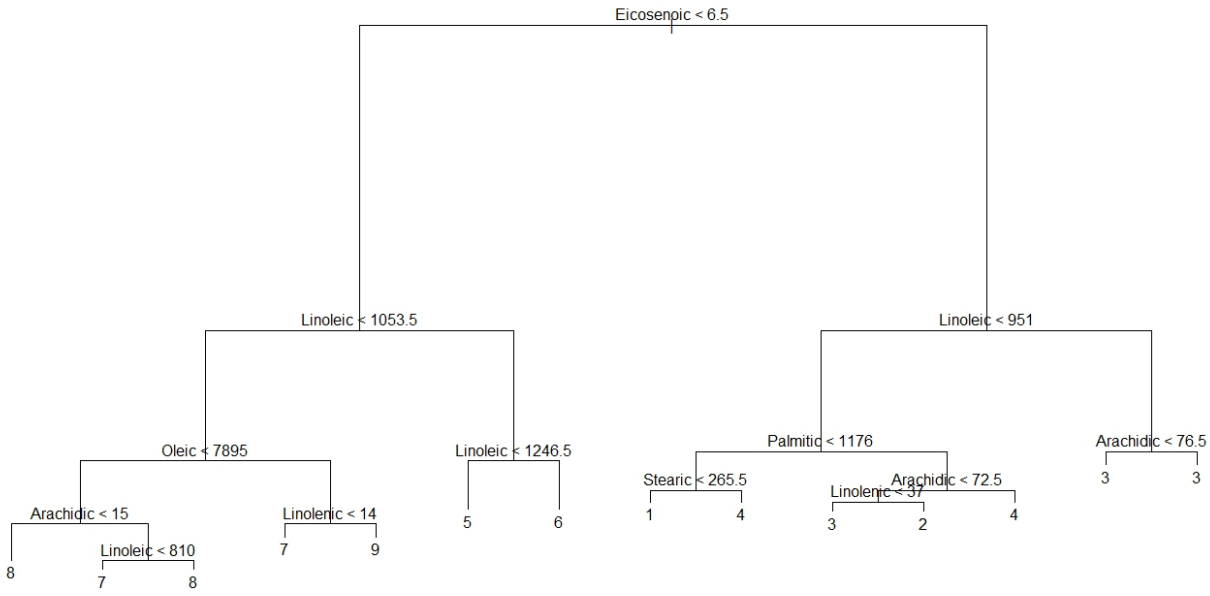

Figure 0.2:

The `prune.tree()` command determines a nested sequence of subtrees of the supplied tree by recursively snipping off the least important splits in the regression tree.

```
olive.pruned <- prune.tree(olive.tree,best=6)

plot(olive.pruned); text(olive.pruned)
```

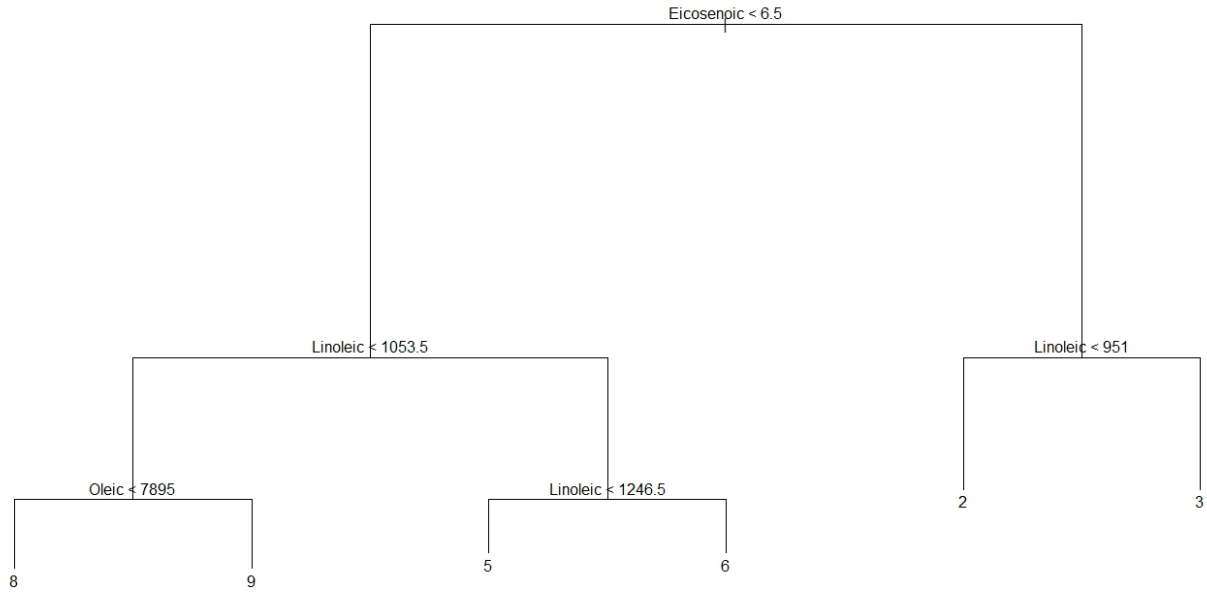

Figure 0.3:

```
newData = data.frame(Palmitic = 1200, Palmitoleic = 120,
          Stearic=200, Oleic=7000, Linoleic = 900,
          Linolenic = 32, Arachidic=60, Eicosenoic=6)

predict(olive.pruned, newData)
```

```
predict(olive.pruned, newData)
  1 2 3 4 5 6         7         8 9
1 0 0 0 0 0 0 0.4842105 0.5157895 0
```