ST4061 - Statistical Methods for Machine Learning II
ST6041 - Machine Learning and Statistical Analytics II

CA1 Answer document

Question 1

<table>
<tr><th>Question item</th><th>Answer</th></tr>
<tr><td>(a)</td><td>
<table>
<tr><th></th><th>Length</th><th>Width</th><th>Leaf</th><th>Curve</th><th>Prediction</th></tr>
<tr><td>Obs1</td><td>4.5</td><td>2.3</td><td>1.3</td><td>0.3</td><td>Young</td></tr>
<tr><td>Obs2</td><td>5.0</td><td>3.5</td><td>4.3</td><td>0.3</td><td>Intermediate</td></tr>
<tr><td>Obs3</td><td>6.1</td><td>3.0</td><td>4.9</td><td>1.8</td><td>Mature</td></tr>
<tr><td>Obs4</td><td>7.2</td><td>3.0</td><td>5.8</td><td>1.9</td><td>Mature</td></tr>
<tr><td>Obs5</td><td>5.1</td><td>3.8</td><td>2.5</td><td>0.4</td><td>Intermediate</td></tr>
</table>
</td></tr>
<tr><td>(b)</td><td>Misclassification rate = 100*(3/5) = 60%</td></tr>
<tr><td>(c)</td><td>i) Sum of all elements in the confusion matrix provided = 42

ii) 100*(Sum of all leading diagonal elements/Total) = 100*((8 + 12 + 9)/42) = 69.04%

iii) 100*(8/(8+3+1)) = 66.67%

iv) 100*((0+3)/(0+3+9)) = 25%</td></tr>
</table>

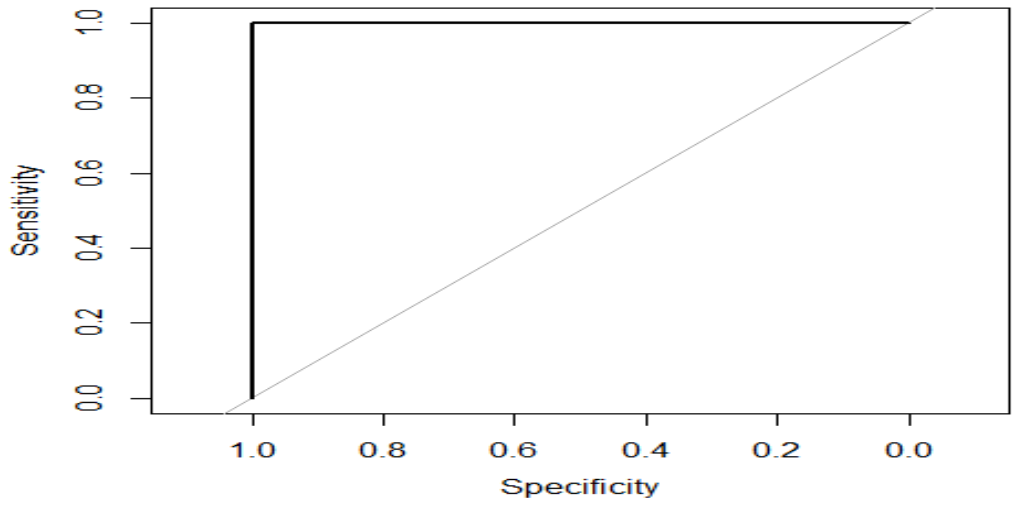
Question 2

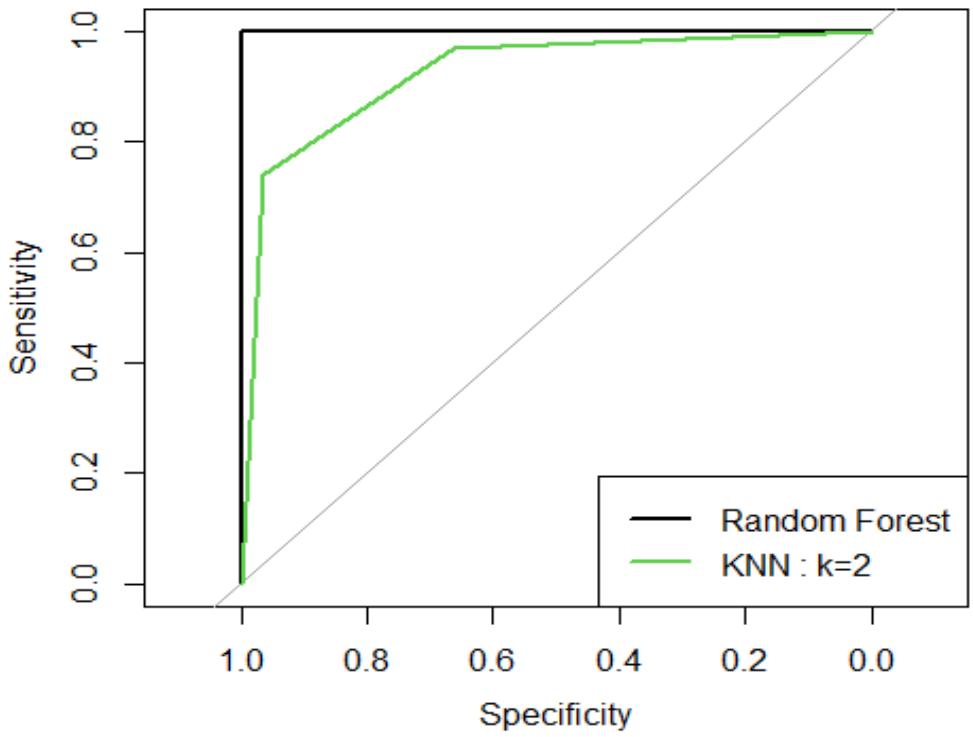
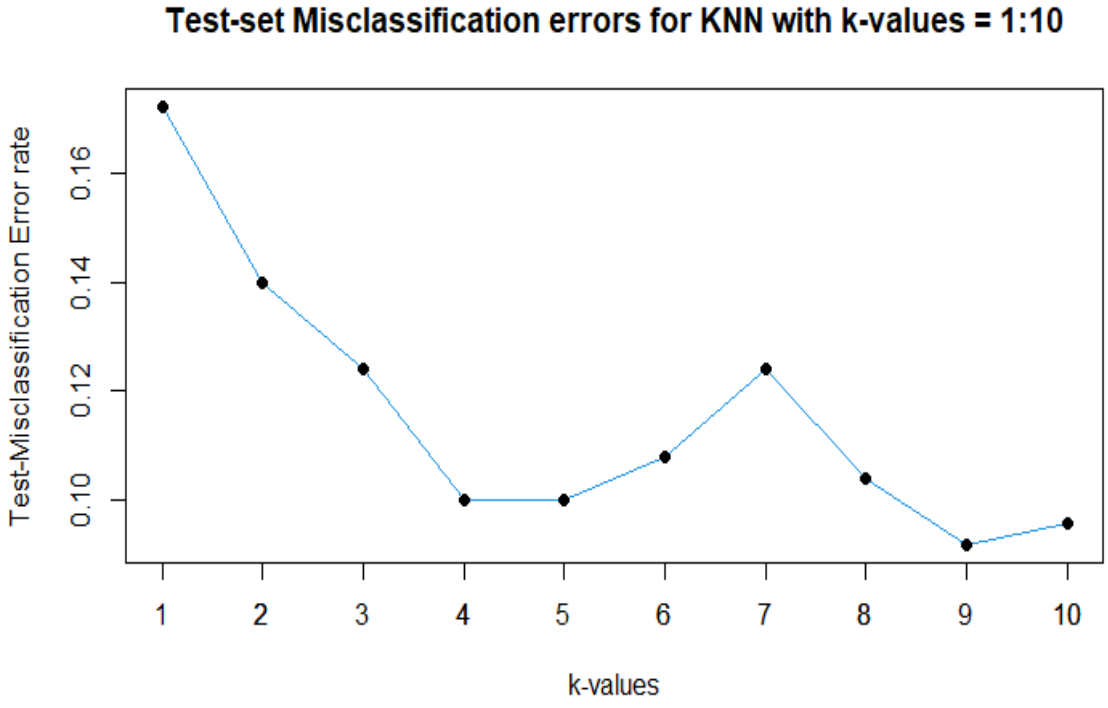
Question item	Answer
(a)	It's a **regression** problem as the dependent variable is salary (in log scale here) which is a range of numbers (continuous variable) and not a finite discrete set of values.
(b)	It is a **generalized linear model fit with elastic-net regularisation**. As alpha = 0.5, it is an exact 50-50 mix of L1 (Lasso) and L2 (Ridge) regression penalty applied to the loss (mean square error) function during the training of the model.
(c)	Type of Cross-Validation applied to mod2 is K-fold Cross-Validation and K=10 where dataset is divided into 10 folds using *folds=cut(1:n,K,labels=FALSE)* command. So, it's 10-fold CV. Also, *cv.glmnet()* command uses a 10-fold CV internally to compute the optimum lambda value and is passed as a parameter for each of the 'K' mod2 fits.

Question 3

Question item	Answer

(a)	Misclassification error rate from training data = **0%**
	`> prediction.conf`
	
	`rf.tree.preds Down  Up`
	`Down  486   0`
	`Up      0 514`

(b)	 `Area under the curve: 1`

(c)	 Area under the curve for KNN, k = 2: **0.9244**
(d)	

R code for Question 3

```
rm(list = ls())
require(ISLR)
require(class)
require(pROC)
library(randomForest)

x = Smarket[,-9]
y = Smarket$Direction
set.seed(4061)
train = sample(1:nrow(Smarket),1000)

## Question 3 i)

rf.tree = randomForest(y[train] ~ ., data = x[train,])
rf.tree
#summary(rf.tree)
rf.tree.preds = predict(rf.tree, x[train,], type = 'class')
prediction.conf = table(rf.tree.preds, y[train])
prediction.conf
missclass_rate = (1 - (sum(diag(prediction.conf))/sum(prediction.conf)))
missclass_rate

## Question 3 ii)
y_test_true = y[-train]
test_preds = predict(rf.tree, newdata=x[-train,], type='class')
#(rf.test.confusion = table(test_preds, y_test_true))
rftree.probs = predict(rf.tree, x[-train,], type="prob")
roc = roc(response=y_test_true, predictor=rftree.probs[,2])
auc = roc$auc
auc
plot(roc, col=1)

## Question 3 iii)
k = 2
knn.o = knn(x[train,], x[-train,], y[train], k)
knn.preds = as.numeric(knn.o == 'Up')
knn.p = attributes(knn(x[train,], x[-train,], y[train], k, prob=TRUE))$prob
new.probs = 1 - knn.p
final.knn.preds = ifelse(knn.preds == 1,knn.p, new.probs)
roc_knn = roc(y_test_true, final.knn.preds)
plot(roc_knn, add = TRUE, col = 75)
legend("bottomright", legend = c("Random Forest", "KNN : k=2"), col = c(1, 75), lty = 1,
lwd = 2)
auc_knn = roc_knn$auc
auc_knn
```

```
## Question 3 iv)
set.seed(4061)
M = 1000
train = sample(1:nrow(Smarket), M)

K = 10
test_class_errors = numeric(K)*NA
for(k in 1:K) {
  knn.o = knn(x[train,], x[-train,], y[train], k)
  confusion_mat = table(knn.o, y[-train])
  test_class_errors[k] = (1 - (sum(diag(confusion_mat))/sum(confusion_mat)))
}
test_class_errors
plot(seq(1:K), test_class_errors, xlim = c(1,10),
     xlab = "k-values", ylab = "Test-Misclassification Error rate",
     main = paste("Test-set Misclassification errors for KNN with k-values = 1:",K,sep=''),
     col = 4, type = 'l')
points(seq(1:K), test_class_errors, col=1, pch=20, cex = 1.4)
axis(side = 1, at = seq(1, 10, by = 1), labels = seq(1, 10, by = 1))
```