Use the data set OJ which is part of the ISLR package to answer the following questions.

(a) Create a training set containing a random sample of 800 observations, and a test set containing the remaining observations.

Solution:

Below is the code to create a random sample of training and test set. The training set has 800 points and the test set has 270 points.

```
> set.seed(1)
> train<-sample(nrow(oj),800)
> train.oj<-oj[train,]
> test.oj<-oj[-train,]
```

(b) Fit a support vector classifier to the training data using cost=0.01, with Purchase as the response and the other variables as predictors. Use the summary() function to produce summary statistics, and describe the results obtained.

Solution:

Below is the code for fitting a support vector classifier using a cost value of 0.01 and kernel function as linear with Purchase as a response variable.

```
> library(e1071)
> svmfit<-svm(Purchase~.,data=train.oj,kernel="linear",cost=0.01)
> summary(svmfit)

Call:
svm(formula = Purchase ~ ., data = train.oj, kernel = "linear",
    cost = 0.01)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
       cost:  0.01
      gamma:  0.05555556

Number of Support Vectors:  432

 ( 215 217 )


Number of Classes:  2

Levels:
 CH MM
```

The Support Vector Classifier created 432 support vectors out of the 800 training points. Out of these 432 support vectors 215 belong to the CH class and 217 belong to the MM class.

(c) What are the training and test error rates?

Solution:

Training error rate code is shown below:

```
> train.pred <- predict(svmfit, train.oj)
> table(train.oj$Purchase, train.pred)
    train.pred
      CH  MM
  CH 439  55
  MM  78 228
> mean(train.oj$Purchase!=train.pred) #misclassification rate
[1] 0.16625
```

This misclassification rate or error rate of support vector classifier on the training set is 16.6% which indicates that the classifier wrongly classifies the purchase 16.6% of the times.

Testing error rate code is shown below:

```
> test.pred <- predict(svmfit, test.oj)
> table(test.oj$Purchase, test.pred)
    test.pred
      CH  MM
  CH 141  18
  MM  31  80
> mean(test.oj$Purchase!=test.pred) #misclassification rate
[1] 0.1814815
```

This misclassification rate or error rate of support vector classifier on the test set is 18.14% which indicates that the classifier wrongly classifies the purchase 18.14% of the times.

(d) Use the tune() function to select an optimal cost. Consider values in range 0.01 to 10.

Solution:

The code using tune() function to select an optimal cost is given below:

```
> set.seed(2)
> tune.train.out<-tune(svm,Purchase~.,data=train.oj,kernel="linear", ranges=list(
cost=c(0.01,0.05,0.1,0.5,1,5,10)))
> summary(tune.train.out)

Parameter tuning of ‘svm’:

- sampling method: 10-fold cross validation

- best parameters:
 cost
  0.1

- best performance: 0.1625

- Detailed performance results:
   cost   error dispersion
1  0.01 0.17125 0.05172376
2  0.05 0.16500 0.04594683
3  0.10 0.16250 0.04787136
4  0.50 0.16625 0.03998698
5  1.00 0.16500 0.03670453
6  5.00 0.16625 0.03586723
7 10.00 0.16750 0.03917553
```

The best cost value is 0.1 with an error of 0.16250

(e) Compute the training and test error rates using this new value for cost.

Solution:

Training error rate:

```
> svmfit.linear<-svm(Purchase~.,data=train.oj,kernel="linear",cost=0.1)
> summary(svmfit.linear)

Call:
svm(formula = Purchase ~ ., data = train.oj, kernel = "linear",
    cost = 0.1)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
       cost:  0.1
      gamma:  0.05555556

Number of Support Vectors:  343

 ( 171 172 )


Number of Classes:  2

Levels:
 CH MM



> train.pred.linear <- predict(svmfit.linear, train.oj)
> table(train.oj$Purchase, train.pred.linear)
    train.pred.linear
      CH  MM
  CH 438  56
  MM  71 235
> mean(train.oj$Purchase!=train.pred.linear) #misclassification rate
[1] 0.15875
```

Test error rate:

```
> test.pred.linear <- predict(svmfit.linear, test.oj)
> table(test.oj$Purchase, test.pred.linear)
    test.pred.linear
      CH  MM
  CH 140  19
  MM  32  79
> mean(test.oj$Purchase!=test.pred.linear) #misclassification rate
[1] 0.1888889
```

We see that the training error using the new cost value has decreased to 15.9 while the test error has increased slightly to 18.9

(f) Repeat (b-e) using svm with polynomial kernel with degree =2.

Solution:

```
> svmfit.poly<-svm(Purchase~.,data=train.oj,kernel="polynomial",cost=0.01,degree=2)
> summary(svmfit.poly)

Call:
svm(formula = Purchase ~ ., data = train.oj, kernel = "polynomial", cost = 0.01,
    degree = 2)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  polynomial
       cost:  0.01
     degree:  2
      gamma:  0.05555556
     coef.0:  0

Number of Support Vectors:  620

 ( 306 314 )


Number of Classes:  2

Levels:
 CH MM



> train.pred.poly <- predict(svmfit.poly, train.oj)
> table(train.oj$Purchase, train.pred.poly)
    train.pred.poly
      CH  MM
  CH 494   0
  MM 306   0
> mean(train.oj$Purchase!=train.pred.poly) #misclassification rate
[1] 0.3825

> test.pred.poly <- predict(svmfit.poly, test.oj)
> table(test.oj$Purchase, test.pred.poly)
    test.pred.poly
      CH  MM
  CH 159   0
  MM 111   0
> mean(test.oj$Purchase!=test.pred.poly) #misclassification rate
[1] 0.4111111
```

The SVM used 620 support vectors out of 800 data points. Out of the 620 support vectors 306 are CH class and 314 are MM class. Using cost=0.01 degree=2 and a polynomial kernel, the training error rate is 38.25 %. The test error rate is 41.11%.

We use the tune function to find the optimum cost value associated with a polynomial kernel function.

```
> tune.train.poly<-tune(svm,Purchase~.,data=train.oj,kernel="polynomial",degree=2, ranges=list(cost=c(
0.01,0.05,0.1,0.5,1,5,10)))
> summary(tune.train.poly)

Parameter tuning of 'svm':

- sampling method: 10-fold cross validation

- best parameters:
 cost
   10

- best performance: 0.17125

- Detailed performance results:
   cost   error dispersion
1  0.01 0.38250 0.04377975
2  0.05 0.33875 0.05084358
3  0.10 0.32000 0.04609772
4  0.50 0.21250 0.03864008
5  1.00 0.19500 0.04456581
6  5.00 0.17375 0.04387878
7 10.00 0.17125 0.04715886
```

We see that the best performance is obtained for cost=10. So we use this cost function to create the SV classifier and try to ascertain the train and test error.

```
> svmfit.polybest<-svm(Purchase~.,data=train.oj,kernel="polynomial",degree=2,cost=tune.train.poly$best
.parameters$cost)
> summary(svmfit.polybest)

Call:
svm(formula = Purchase ~ ., data = train.oj, kernel = "polynomial", degree = 2,
    cost = tune.train.poly$best.parameters$cost)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  polynomial
       cost:  10
     degree:  2
      gamma:  0.05555556
     coef.0:  0

Number of Support Vectors:  342

 ( 170 172 )


Number of Classes:  2

Levels:
 CH MM
```

The Support Vector Machine created 342 support vectors out of the 800 training points. Out of these 342 support vectors 170 belong to the CH class and 172 belong to the MM class.

We can try to find the training and test error for this classifier to check if there are any performance improvements over the linear kernel using the best cost value. The code is shown below.

```
> train.pred.poly <- predict(svmfit.polybest, train.oj)
> table(train.oj$Purchase, train.pred.poly)
    train.pred.poly
      CH  MM
  CH 450  44
  MM  72 234
> mean(train.oj$Purchase!=train.pred.poly) #misclassification rate
[1] 0.145
>
> test.pred.poly <- predict(svmfit.polybest, test.oj)
> table(test.oj$Purchase, test.pred.poly)
    test.pred.poly
      CH  MM
  CH 140  19
  MM  31  80
> mean(test.oj$Purchase!=test.pred.poly) #misclassification rate
[1] 0.1851852
```

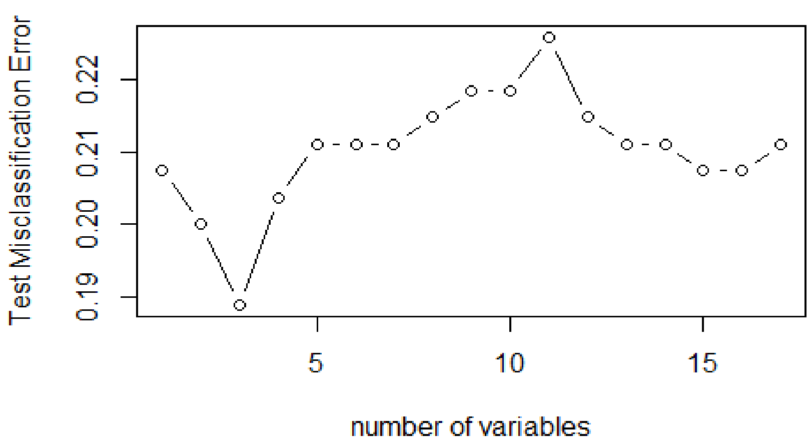
The training and test error are 14.5% and 18.5% respectively. We see that there is an improvement in both the training and test error as compared to the linear kernel model.

(g) Apply random forest to the dataset with Purchase as the response and the other variables as predictors and report the training and test error rates.

<u>Solution</u>:

For the random forest we first have to calculate the optimum value of m to select random predictors. We loop the mtry variable in the randomForest function to obtain a value for which the test misclassification error is the least. Below is the R implementation.

```
> set.seed(1234)
> result<-NULL
> for(k in 1:17){
+   rf.purchase <- randomForest(Purchase~.,data=train.oj,mtry=k,importance=TRUE)
+   rf.pred <- predict(rf.purchase, test.oj)
+   table(test.oj$Purchase,rf.pred)
+   error<-mean(test.oj$Purchase!=rf.pred)
+   result<-rbind(result,c(k,error))
+ }
> plot(result,xlab="number of variables",ylab="Test Misclassification Error", type="b")
```




```
> result
      [,1]      [,2]
 [1,]    1 0.2074074
 [2,]    2 0.2000000
 [3,]    3 0.1888889
 [4,]    4 0.2037037
 [5,]    5 0.2111111
 [6,]    6 0.2111111
 [7,]    7 0.2111111
 [8,]    8 0.2148148
 [9,]    9 0.2185185
[10,]   10 0.2185185
[11,]   11 0.2259259
[12,]   12 0.2148148
[13,]   13 0.2111111
[14,]   14 0.2111111
[15,]   15 0.2074074
[16,]   16 0.2074074
[17,]   17 0.2111111
```

From the above plot we see that for m=3 the test error is the least. Hence we take m=3 and calculate the training error rate.

```
> rf.purchase <- randomForest(Purchase~.,data=train.oj,mtry=3,importance=TRUE)
> rf.pred <- predict(rf.purchase, train.oj)
> table(train.oj$Purchase,rf.pred)
    rf.pred
      CH  MM
  CH 473  21
  MM  45 261
> mean(train.oj$Purchase!=rf.pred)
[1] 0.0825
```

The test error rate is:

```
> rf.pred <- predict(rf.purchase, test.oj)
> table(test.oj$Purchase,rf.pred)
    rf.pred
      CH  MM
  CH 139  20
  MM  33  78
> mean(test.oj$Purchase!=rf.pred)
[1] 0.1962963
```

The training and test error for random forest implementation is 8.25% and 19.6% respectively.

(h) Which method (random forest, support vector classifier or svm) is the best? Suggest the usage of these three methods for practice based on the above analyses.

<u>Solution</u>:

According to the results obtained SVM (kernel with degree 2) has the least misclassification error among the three methods tried. As far as the best method that can be used in practice, it ***highly depends on the dataset*** we are working on. Support vector ***works best when we can find the best kernel*** for the classifier. With the ***right kernel***, support vectors ***guarantees advantages in high-dimensional space and reduced overfitting***. However, results of support vector classifiers and machines cannot be easily comprehensible. Random forest results are ***easy to interpret, are non-parametric and hence outliers don't influence the results***. However, Random forests have been observed to ***overfit*** for some datasets with noisy classification/regression tasks. For example, in the above example, random forest gave the ***least training error (8.25%) and the highest test error (19.6%)***, which gives credence to the stated possible disadvantage.