HOMEWORK 1

Question 2.1

Describe a situation or problem from your job, everyday life, current events, etc., for which a classification model would be appropriate. List some (up to 5) predictors that you might use.

I'm currently working on a project to forecast traffic utilization for a telecom company and could apply a classification model to differentiate traffic patterns. Predictors that could be used include:

1. Time-series predictor – Auto-regressive integrated moving average (ARIMA) models have been used by researchers to predict yearly growth in network traffic
2. Neural Networks-Based Predictors – Deep learning algorithms have been used for traffic prediction in wireless mesh networks
3. Wavelet Transform-Based Predictors – Used for multi-scale prediction as the naturally transforms the network wavelet into multiple resolutions depending on the size and range of the wave

Question 2.2

1. *Using the support vector machine function ksvm contained in the R package kernlab, find a good classifier for this data. Show the equation of your classifier, and how well it classifies the data points in the full data set.*

Multiple test cases were run on the ksvm model using C values ranging from 0.001 to 10000 but did not result in a better training error score. The model returns a training error of : 0.136086 which signifies the probability of prediction errors. The equation of the classifier is: -0.0005(A1) - 0.01(A2) - 0.008(A3) + 0.01(A8) + 0.5(A9) - 0.002(A10) - 0.001(A11) + 0.0003(A12) - 0.2(A14) + 0.06(A15) - 40.4 = 0

Note: I created multiple SVM models in Python using the sklearn library and was able to produce an accuracy score of 99.4% using the Radial basis function kernel. The code is included below the R solution.

```
> d0 = read.table('credit_card_data-headers.txt', header=TRUE)
> d0[,11] = replace(d0[,11], d0[,11] == 0, -1)
> d0[11] = as.factor(d0[,11])
> colnames(d0)[11] = 'class'
> predictors = as.matrix(d0[,1:10]);   classes = as.matrix(d0[,11])
>
> library(kernlab)
> model = ksvm(predictors, classes, type="C-svc", kernel="vanilladot", C=0.2, scaled=TRUE)
 Setting default kernel parameters
> model
Support Vector Machine object of class "ksvm"
```

SV type: C-svc (classification)
parameter : cost C = 0.2

Linear (vanilla) kernel function.

Number of Support Vectors : 196

Objective Function Value : -36.2851
Training error : 0.136086

```
#Calculate the weights & constant
> w = colSums(predictors[model1@SVindex,1:10] * model1@coef[[1]])
> w
           A1            A2            A3            A8            A9
-5.115191e-04 -9.931841e-03 -8.163952e-03  1.086537e-02  5.013006e-01
          A10           A11           A12           A14           A15
-1.628945e-03 -1.024945e-03 -2.559994e-04 -2.016443e-01  5.587158e+02
> a0 <- sum(w*predictors[1,1:10]) - model1@b
> a0
[1] -40.44358
```

Python Solution using the same data:

```
from sklearn.svm import SVC
import pandas as pd
import numpy as np

#Reading in the data
file_loc = (r'C:\Users\Mitchell.Ramey\Documents\credit_card_data-headers.txt')
file = open(file_loc)
file = file.read()
df = pd.read_csv(file_loc, delim_whitespace=True)

#Seperating the Classifier/Predictors
clssfr = df.iloc[:,10:]
clssfr2 = df.loc[:,'R1']
predictors = df.loc[:,'A1':'A15']


#Fitting the model
clf = SVC(gamma='scale', kernel = 'linear')
clf_rbf_auto = SVC(gamma = 'auto')
clf_rbf_scale = SVC(gamma = 'scale')
Y = np.array(clssfr)
Y = np.reshape(Y, (654,))
y = Y
```

```
X = np.array(predictors)
model_linear = clf.fit(X, Y)
model_rbf_auto = clf_rbf_auto.fit(X,Y)
model_rbf_scale = clf_rbf_scale.fit(X,Y)

# Perform classification on samples in X.
y_prediction = clf.predict(X)


# Returns the mean accuracy on the given test data and labels.
linear_training_error = model_linear.score(X, Y)
print(linear_training_error)
#0.8516819571865444
rbf_score = model_rbf_auto.score(X,Y)
print(rbf_score)
#0.9938837920489296
rbf2_score = model_rbf_scale.score(X,Y)
print(rbf2_score)
#0.6605504587155964


# Weights assigned to the features
print('w = ',clf.coef_)

#Constant in the decision function
print('b = ',clf.intercept_)

###
print('Indices of support vectors = ', clf.support_)
print('Support vectors = ', clf.support_vectors_)
print('Number of support vectors for each class = ', clf.n_support_)
print('Coefficients of the support vector in the decision function = ', np.abs(clf.dual_coef_))
```

2. *Using the k-nearest-neighbors classification function `kknn` contained in the R kknn package, suggest a good value of k, and show how well it classifies that data points in the full data set. Don't forget to scale the data (`scale=TRUE` in `kknn`).*

Model1 has the best prediction accuracy with a value of 89.3%

```
> data = d0
> set.seed(9)
> rowidx = sample(1:nrow(data), round(.8*nrow(data)), replace=FALSE)
> X_train = data[rowidx,];   test = data[-rowidx,]
>
> library(kknn)
> model1 = kknn(class~., X_train, test, na.action=na.omit, k=11, distance=1, kernel='inv',scale=TRUE)
> model2 = kknn(class~., X_train, test, na.action=na.omit, k=11, distance=2, kernel='inv',scale=TRUE)
```

```
> model3 = kknn(class~., X_train, test, na.action=na.omit, k=11, distance=3, kernel='inv',scale=TRUE)
> model4 = kknn(class~., X_train, test, na.action=na.omit, k=11, distance=4, kernel='inv',scale=TRUE)
>
> model1accuracy = sum(model1$fitted.values ==  test[,11]) / length(test[,11])
> model2accuracy = sum(model2$fitted.values ==  test[,11]) / length(test[,11])
> model3accuracy = sum(model3$fitted.values ==  test[,11]) / length(test[,11])
> model4accuracy = sum(model4$fitted.values ==  test[,11]) / length(test[,11])
>
> model1accuracy
[1] 0.8931298
> model2accuracy
[1] 0.8854962
> model3accuracy
[1] 0.8778626
> model4accuracy
[1] 0.8778626
```