**Homework5**

Gouthamrajan Nadarajan

**Problem 1:**

**Part (a)**

The variance of the features of the input dataset (each column) is calculated below. Here we see that the variance of the features is very different and calculating the mean of the variances, I found this to be 929.67. Below I show the variance of each of the predictors.

Graphical user interface, text

Description automatically generated

Now calculating the principal components for the dataset, I show the variance for the principal components in the graph below.

Chart, histogram

Description automatically generated

Next calculating the overall percentage of variance that each principal component captures, I show the graph below. **Here I found that you need at least 5 principal components to capture 80% of the variance.**

Chart, histogram

Description automatically generated

Here, I next show biplots using the first two principal components:

Chart

Description automatically generated

Because this is not very interpretable, I also graph the first two principal components plotted against each other. I also graph the third and the fourth principal components plotted against each other as well. Here each color represents its own class.

Chart, scatter chart, bubble chart

Description automatically generated

Next, I graph principal component 3 vs. principal component 4. Here you can see that there is not as clear separation between the classes as we saw in the previous figure. This tells us that as expected, using the principal components that capture the most variance (PC1 and PC2), separates the data the best.

Chart, scatter chart

Description automatically generated

Below I show a graph of the first three principal components. Here with this added dimension, we again see good separation among the classes.

Chart, surface chart

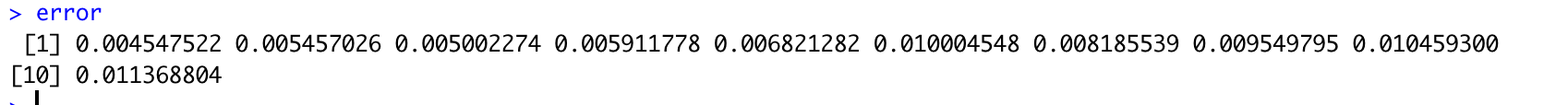
Description automatically generated

**Part (b)**

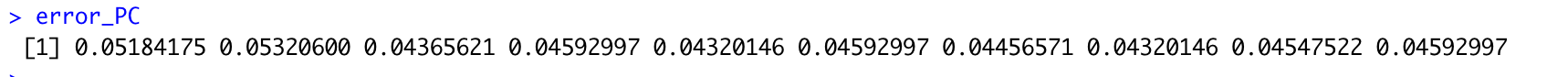
I next divided the dataset into a training and testing set using an 80:20 train/test split.

First performing KNN classification on the raw data, I first did a parameter search over values of K from 1 through 10.

Below I show the error for each K value below using the raw data. **I found that the model trained with k=1 had the lowest error. The error for this model is 0.0045**



I then compared this with a KNN model trained on the dataset of the first 5 principal components. Here I used the same train/test split and again did a parameter search using values of k from 1 through 10. I show the errors below



**Here I found that the model with k=5 had the lowest error. This model had an error of 0.043201.** Here we see that this model has a higher error than the model trained on the entire dataset. However, this is understandable if you consider we are only using 1/3 of the total amount of information in this case when compared to the KNN model fitted over the entire raw dataset.

**Part (C)**

In this part, I also explored fitting a Random Forest model. I trained the random forest using 100 trees and used the same train/test split as in part b.

Training the random forest on the raw data, I found the training accuracy to be 100% and I found the testing accuracy to be 99.23%. I show the confusion matrix on the holdout testing set below:

A picture containing text

Description automatically generated

Now comparing this to a random forest trained on the first 5 principal components, I found that I can get a perfect 100% accuracy on both the training and testing sets. we see the following confusion matrix results on the testing dataset:

A picture containing text

Description automatically generated

Comparing across the different models, I found that a random forest trained on the principal components that captures 80% of the variance performs on par to a model trained on the entire dataset. This tells me that we do not actually need the entire dataset for classification. Using principal components, we are essentially able to derive features that are most discriminative (have the highest variance) and using that yields on par or even superior classification depending on the classifying algorithm. For the KNN algorithm, I found that the classifier trained on the principal components performs poorly compared to the classifier trained on all the data. This could be attributed to KNN using a much similar non-parametric method to classify the data which does not allow it to take advantage of the highly discriminative PC derived features. This is the opposite to what I had saw with the random forest classifier indicating that not all classifiers perform better with principal component derived features. This reinforces the “no free lunch theorem”.

**Problem 2:**

For this problem, I divided the dataset using an 80:20 train/test split. I also converted the non-numeric variables to numeric values.

**Fitting a neural network:**

Fitting the neural network data, I trained a model with 1 hidden layer. To find the best number of nodes to put in the hidden layer, I fit multiple models using a different number of nodes from 1 to 10. **For each model, I calculated the training and testing errors. The model that performed best on the test set was the model that had 3 nodes in the hidden layer which had a testing error (misclassification rate) of 0.13333.** Below, I show the training and testing errors for all the 10 different models:

A screenshot of a computer

Description automatically generated

Fitting the best neural network model (using 3 nodes in the hidden layer) again, I then generated the confusion matrix for the testing dataset. Note that because neural networks are not deterministic, when I fit the model for the second time, I did not get identical results as when I trained the neural networks the first time.

Text

Description automatically generated with medium confidence

**Fitting a CART model:**

I fitted a CART model and investigated different complexity parameters. I show a graph of error vs. complexity parameter below. I found that cp=0.063931229 had the lowest error when using 10 fold cross validation.



I then fitted a cart model using cp=0.063931229 and after pruning the model, I show the following tree structure visually below:



For this model, **I calculated the training and testing error to be 0.2331 and 0.2333 respectively.**

I show the confusion matrix for the CART model on the testing data below:

**Text

Description automatically generated**

**Fitting data using a Random Forest:**

Next using a Random Forest classifier, I decided to fit the data using the num\_trees = 1000.

**I fit the data and found that I got perfect accuracy on the training set (0 misclassification error) and an error of 0.133 on the test set.**

Generating the confusion matrix on the testing set I got below:

**Text

Description automatically generated**

Comparing across all three models, I found that the Random Forest and the Neural network both had the lowest test errors at 0.133 and the CART model had a higher testing error of 0.2333. This is to be expected as a single regression tree is not as generalizable as a random forest model. While the Random Forest and the Neural Net both performed similarly, I would prefer in most cases to use a Random Forest over the Neural network in this case as the random forest is much easier to tune and trained significantly faster on my computer than did the neural network.

**Problem 3**

**Part (a), (b), and (c):**

I first divided the dataset into a train/test dataset where 800 samples were used for training and 270 samples for testing. I show the summary of the fit below

Graphical user interface, text, application

Description automatically generated

Fitting a support vector machine using a linear kernel and cost of 0.01, we see that there are 654 support vectors which are the points along the decision boundary that influenced the shape of the boundary the most.

**Here for this SVM model, the misclassification error rate (1- Accuracy) for training and testing datasets was found to be 0.1825 and 0.1963 respectively. Below I show the confusion matrix for the testing dataset:**

Text, letter

Description automatically generated

**Part (d):**

Next tuning the SVM, we looked at values for cost for 0.001, 0.01, 0.1, 1, 3, 5, 7, and 10. Based on 10-fold cross validation, we found that the model with cost=0.01 performed best which had a average cross validation error of 0.134.

Graphical user interface, text

Description automatically generated

**Part(e):**

We next fit this model using our best cost of cost=0.01. **We found that the training and testing errors of our model was 0.1762 and 0.1667 respectively. Below, I show the confusion matrix on the test set for this model:**

Table

Description automatically generated

**Part (f):**

I then repeated this parameter search process for the cost parameter using an SVM trained with a radial kernel. Here I found that the model with cost=1 performed the best and had a average cross validation error of 0.130188.

Text

Description automatically generated

**Training the radial kernel SVM model with cost=1, I found the training and testing error to be 0.1525 and 0.1852 respectively**. Below is the confusion matrix for the testing dataset.

A picture containing table

Description automatically generated

**Part (g):**

Repeating this process for a support vector machine trained using a polynomial kernel using degree=2, I first performed a similar parameter search for the cost parameter. Here I found that cost=3 performed the best with an average cross validation error of 0.1459

Text

Description automatically generated

Training this SVM with cost=3**, I found that this model got a training and testing error of 0.1738 and 0.2296 respectively. Below I show the confusion matrix for the model on the testing data:**

Text

Description automatically generated

**Part (h):**

Comparing across all the models, the SVM with linear kernel performed best with a misclassification error of 0.1667, the SVM with the radial kernel performed well with misclassification error of 0.1852, and the SVM with polynomial kernel performed worst with 0.2296 testing error. This could be explained by the data being best distributed in such a way that the linear and radial basis functions can segment the data best rather than a nonlinear polynomial basis function.