Programming Assignment 2

Jake Sciotto
EN685.621 - Algorithms for Data Science
770-286-8893
Johns Hopkins University

August 17, 2020

Problem 1 - Machine Learning

1. Data Cleansing (use the iris_data_for_cleansing.csv)

   After modifying the column names, we can see that we have a varying count of features per column.

	sepal-length	sepal-width	petal-length	petal-width	feature-5	feature-6	class
count	147.000000	147.000000	147.000000	149.000000	148.000000	148.000000	150.000000
mean	5.845578	3.053741	3.751701	1.205369	1.375071	3.523958	2.000000
std	0.834958	0.437188	1.760151	0.761292	0.135347	0.900837	0.819232
min	4.300000	2.000000	1.000000	0.100000	1.120668	1.976229	1.000000
25%	5.100000	2.800000	1.600000	0.300000	1.270109	2.745057	1.000000
50%	5.800000	3.000000	4.300000	1.300000	1.382528	3.441192	2.000000
75%	6.400000	3.300000	5.100000	1.800000	1.473925	4.364670	3.000000
max	7.900000	4.400000	6.900000	2.500000	1.685578	5.104838	3.000000

   Figure 1: Original data exploration

   This is due to several NaN values. I filled these in with the median.

2. Generate two sets of features from the original 4 features to end up with a total of 8 features

   Using the results from Programming Assignment 1, we are going to generate the extra two features from the top ranked features.

   The general feature generation algorithm was as follows:

   (a) Calculate the mean, standard deviation, and covariance matrix for two features (in this case petal length and petal width
   (b) Generate 50 new observations per class per feature
   (c) Calculate the new means and standard deviations
   (d) Perform z-score normalization on these new features
   (e) Multiply by petal-length/petal-width covariance matrix
   (f) Add back the mean of the original data to scale it appropriately

	sepal-length	sepal-width	petal-length	petal-width	feature-5	feature-6	feature-7	feature-8	class
0	5.100000	3.500000	1.400000	0.200000	1.611281	2.981148	1.688530	0.216784	1
1	4.900000	3.000000	1.400000	0.200000	1.295847	2.210908	1.450339	0.226244	1
2	4.700000	3.053741	1.300000	0.200000	1.685578	3.114562	1.435315	0.287895	1
3	4.600000	3.100000	1.500000	0.200000	1.546064	2.714977	1.614887	0.294221	1
4	5.000000	3.600000	1.400000	0.200000	1.501464	2.815603	1.735079	0.241314	1
5	5.400000	3.900000	1.700000	0.400000	1.325835	2.235803	1.612376	0.264735	1
6	4.600000	3.400000	1.400000	0.300000	1.529386	2.792032	1.448460	0.237832	1
7	5.000000	3.400000	1.500000	0.200000	1.331028	2.278499	1.324568	0.243768	1
8	4.400000	2.900000	1.400000	0.200000	1.147873	1.976229	1.717279	0.201962	1
9	4.900000	3.100000	1.500000	0.100000	1.401505	2.434789	1.490861	0.233069	1
10	5.400000	3.700000	1.500000	0.200000	1.340574	2.380203	1.376924	0.262190	1
11	4.800000	3.400000	1.600000	0.200000	1.259047	2.296553	1.725666	0.282007	1
12	4.800000	3.000000	3.751701	0.100000	1.526365	2.794599	1.325299	0.262883	1
13	4.300000	3.000000	1.100000	0.100000	1.314645	2.262849	1.561354	0.302939	1
14	5.800000	4.000000	1.200000	0.200000	1.495580	2.590192	1.700619	0.280801	1
15	5.700000	4.400000	1.500000	0.400000	1.403405	2.429602	1.701930	0.260934	1
16	5.400000	3.900000	1.300000	0.400000	1.651607	3.045064	1.553559	0.305849	1
17	5.100000	3.500000	1.400000	0.300000	1.181882	2.088136	1.570059	0.217882	1
18	5.700000	3.800000	1.700000	0.300000	1.432863	2.460808	1.476348	0.245577	1
19	5.100000	3.800000	1.500000	0.300000	1.193632	2.108733	1.691248	0.298073	1
20	5.400000	3.400000	1.700000	0.200000	1.426801	3.523958	1.525907	0.263781	1

Figure 2: New features

3. Perform Feature Preprocessing

   Upon examination of the features, I found that there were obvious outliers in the sepal width class. I initially chose to replace these with the median, however, upon further thinking and examination I chose to leave them. They comprise a very small amount of the dataset (less than 5%). There are many options for dealing with these outliers, including replacement or removal. Replacement could be done with the mean, but not without affecting summary statistics. It is my belief that they are fine as is.

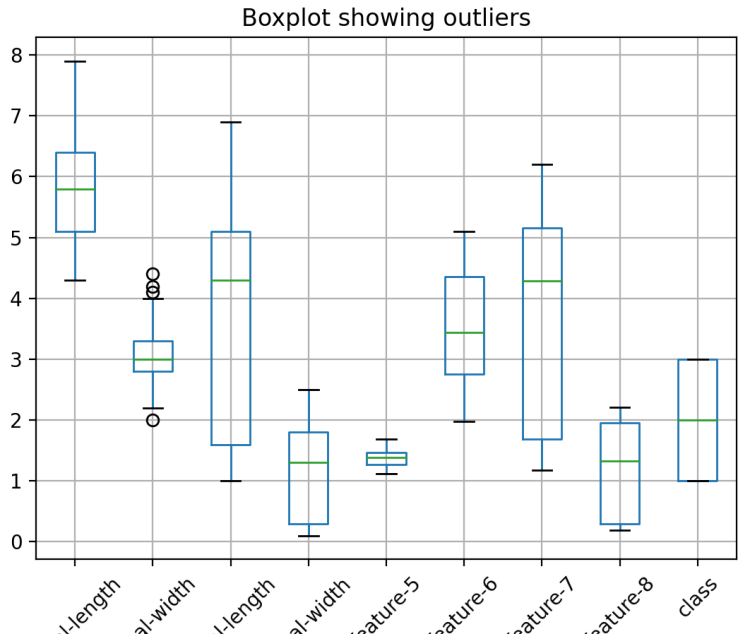

Figure 3: New features

4. Rank the 6 set of features to determine which are the two top features

   For this programming assignment, I have used the Bhattacharyya distance to perform the feature ranking. The method employed draws samples from a continuous probability density function and performs a rough integration over a number of steps, returning the Bhattacharyya coefficient. From this, we can take the log and return the negative value to get the Bhattacharyya distance. The feature ranking is below.

```
Bhattacharyya distance for: feature-8 68.29041728813566
Bhattacharyya distance for: feature-7 40.54541799254838
Bhattacharyya distance for: petal-length 2.1153577088274553
Bhattacharyya distance for: petal-width 2.096844926034446
Bhattacharyya distance for: feature-6 1.0588166875365528
Bhattacharyya distance for: sepal-length 0.4831027934510407
Bhattacharyya distance for: sepal-width 0.4143419397210582
Bhattacharyya distance for: feature-5 0.10826981585464973
```

Figure 4: Feature ranking using Bhattacharyya distance

5. Reduce the dimensionality to two features using PCA or Kernel PCA

   To reduce dimensionality, I used PCA. PCA is a technique from linear algebra that can be used to perform dimensionality reduction. Having a large number of a dimensions in a feature space can mean that the space's volume is very large, so the points that we have are often a small and non-representative example. Projection methods like PCA reduce the number of dimensions while preserving the structure and relationship between the variables.

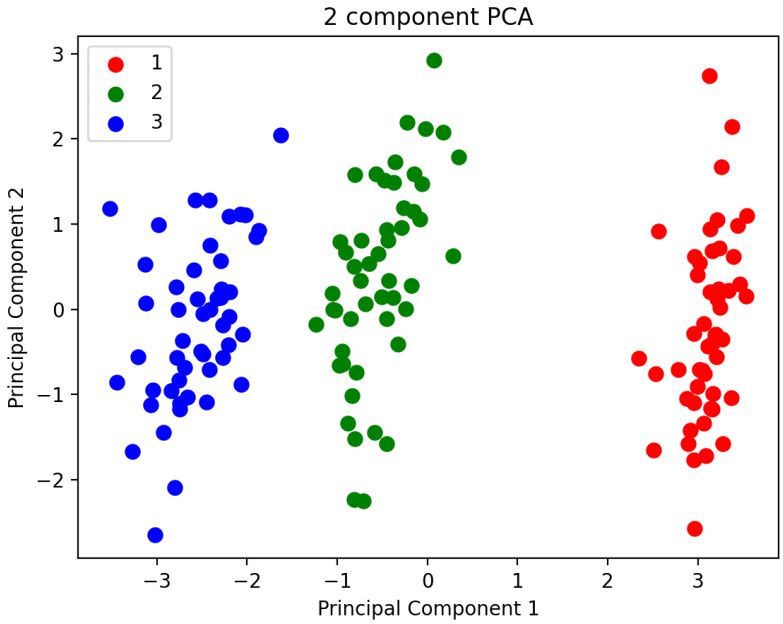

Figure 5: 2 Component PCA

6. Using the following Machine Learning techniques, classify the three class Iris data:

   (a) Expectation Maximization (Gaussian Mixture Model)

   As you can see, Expectation Maximization was fairly successful. To perform Expectation Maximization, I employed the use of the build in Gaussian Mixture Model class. The results (i.e. updated means and iterations to convergence) are written to the output file in the output folder included in the project. As you can see, the model did a decent job with the predictions (x marker) when plotted against the oridinal Iris data (circle marker). There is some overlap between the versicolor and virginica classes, but setosa is correct minus one in the bottom left corner of the graph.

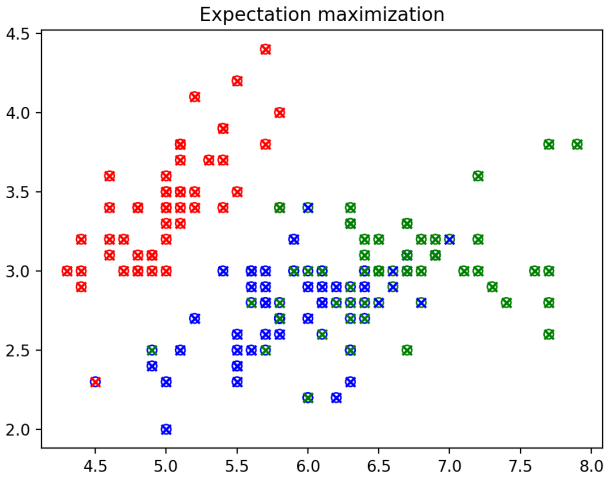

Figure 6: Expectation Maximization

   (b) Either Fisher Linear Discriminant (Linear Discriminant Analysis) or Parzen Window

   Here I used Linear Discriminant Analysis. Whereas PCA does not take into account any difference in class, LDA explicitly attempts to model the difference between classes of data. As you can see, we have a very clear separation of classes. Discriminant analysis is especially beneficial when groups are known a priori, like they are here. Simply put, LDA is classification.

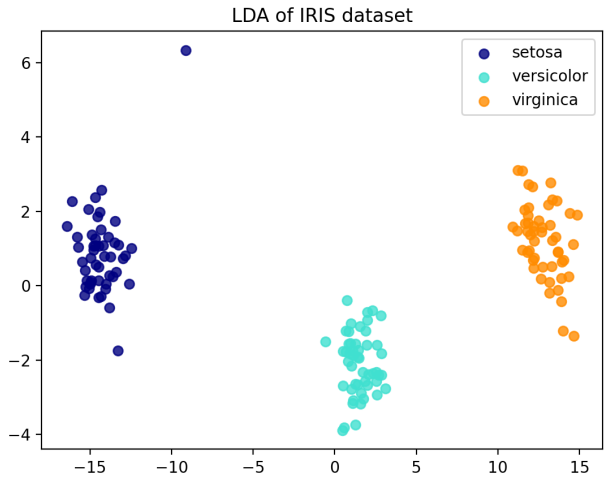

Figure 7: Linear Discriminant Analysis

(c) Neural Network Method (Probabilistic NN, Radial Basis Function or Feed Forward)

For a nueral network method, I used a multi-layer perceptron classifier. An MLP is a class of feed-forward artifical neural networks. They consist of three layers of nodes - an input layer, hidden layer and an output layer. Except for the input nodes, each node uses an activation function and backpropagation for training. It works very well for distinguishing data that is not linearly separable. I used this one because I had some familiarity with it and how it performs. There are some complicated elements to it, and the math is not particularly complicated, but I will attempt to explain to the best of my understanding.

After splitting the data into training and testing data, there are a number of parameters that have to be chosen for the MLPClassifier method. I started with 50 iterations, which was more than enough. After that, picking the solver - I used stochastic gradient descent with a .1 learning rate, because SGD converges well and we do not want to end up on the other side of the function. The activation function is *tanh*, and the number of hidden layers is (10, 10, 10), so three layers of 10 neurons a piece. This was more than enough. An MLP classifier works very well with data that is linearly separable, and our Iris data is extremely separable (seen from LDA), so adjusting some parameters brought our classification in some cases to almost .99 accuracy.

(d) Support Vector Machine

For SVM, I used a linear kernel to achieve some separation. There was a little bit of mixing between the versicolor and virginica classes, like with expectation maximization. It worked well up to a certain point, but I believe some of the noise in the sepal-width class is responsible for the mix-up between virginica and versicolor classes. There are other kernels and functions for SVM that we could have used, but some of them are not as useful. For example, the sigmoid function is more suitable for binary classification problems. However, in this case we had three output classes and therefore a parametric model (linear SVM) was appropriate.

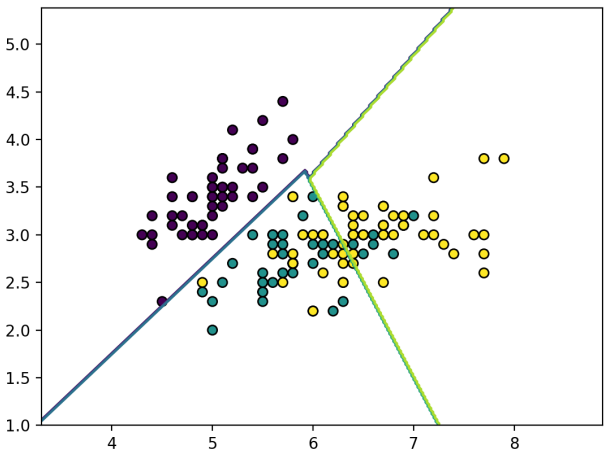

Figure 8: Support Vector Machine

Problem 2 - Game Theory

For this problem, I achieved two separate solutions. The first of which was a simple application of minimax, in which the algorithm builds a set of moves and returns either the max (if we are maximizing) or the min (if we are minimizing). The decisionMaker function is called when the first move is made, and the best move is returned to the function. The other was an adaptation of classmate Amit Poddar's solution from office hours. It is commented out, but it works nearly the same. No matter the choice the player makes, the algorithm chooses the top left square first as this is the result of a win propagating from the left most branch of the search tree. You can see the slight benefits in alpha beta pruning by swapping the comment on the following lines:

```
#F = decisionMaker(G, False, 0)
F = decisionMaker(G, 0, 1)
```

Figure 9: From TicTacToe code

When you initially run the TicTacToe code, you can see that there is a slight hang-up in the program execution when you make the first move. Alpha-beta pruning removes this slight hiccup by not removing some of the tree search options along the way.