Problem 1 - Spam Filter Classifiers

(a) Classification Error and Comments

Least Sum of Squares Classifier - Accuracy: **91.35%**

Bayes Classifier using Parzen Windowing - Accuracy: **88.48%**

Support Vector Machines (SVM) - Accuracy: **91.7%**

The classifiers all obtained approximately 90% classification accuracy. On this merit alone, it is hard to declare any one classifier as superior to another, though it is worth noting that both SVM and the Bayes classifier could theoretically be improved with parameter tuning, while the results of the LS classifier are fixed.

In a practical setting, it may also be relevant to note time to convergence for these three algorithms. The LS classifier does not iteratively improve, and therefore runs instantaneously. The Bayes classifier is more computationally intensive, and takes longer to produce results than the LS classifier. The SVM classifier takes a significant amount of time, which is the main drawback to this otherwise superior classifier.

One additional metric of interest with regards to the performance of these classifiers is the "false positive" error rate, which is reported below in the confusion matrices and discussed further in the following suggestions for improvement.

(b) Confusion Matrices

LSS	**Classified "SPAM"**	**Classified "NOT SPAM"**	**Accuracy**
"SPAM"	807	93 (4.04%)	**89.7%**
"NOT SPAM"	106 (4.61%)	1295	**92.4%**
Proportion/Total	39.67%	60.33%	**91.35%**

Bayes / Parzen	**Classified "SPAM"**	**Classified "NOT SPAM"**	**Accuracy**
"SPAM"	699	201 (8.74%)	**77.66%**
"NOT SPAM"	64 (2.78%)	1337	**95.43%**
Proportion/Total	33.16%	66.84%	**88.48%**

SVM	**Classified "SPAM"**	**Classified "NOT SPAM"**	**Accuracy**
"SPAM"	771	129 (5.61%)	**85.67%**
"NOT SPAM"	62 (2.69%)	1339	**95.57%**
Proportion/Total	36.2%	63.8%	**91.7%**

(c) Suggestions for Improvement

i. Decrease False Positive Rate
The practical application of the classifier must be taken into consideration. The cost of a misclassification is not necessarily equal across both classes. Intuitively, it is a much larger issue to label and thus discard an email as "spam" that is in reality "not spam". For that reason, with these classifiers it may be possible to weight the cost of a "false positive" as higher than that of a "false negative". This would require deriving a new optimisation expression for the cost function, and may be difficult with SVM and the LS classifier. The Bayes classifier could easily be implemented with a new threshold ϵ that awards a reasonable classification preference to the prior estimate for an email to be "not spam".

ii. (SVM) Optimise Pruning Function
In the case of the SVM classifier, a function "Prune" is set up to eliminate data points that are unlikely to be support vectors affecting the outcome of the classification. The algorithm looks at the "nearest" k = 10 neighbours to a data point, and "prunes" (removes) it from the training dataset if all 10 neighbours are of the same class. k = 10 was selected somewhat arbitrarily, and in testing greatly reduced the time to convergence of the SMO algorithm without significantly reducing classifier accuracy. This algorithm could be trained with different values of k to optimise decreasing convergence time while controlling for the cost to the classifier accuracy, which should theoretically be possible to reduce to zero (no change in accuracy).

iii. Dimensionality Reduction
There are 57 response variables (features) in the dataset. The dataset is computationally expensive, especially in the case of the SVM classifier, which produces the best accuracy and lowest false positive error rate. The "Prune" function may also suffer from the "curse of dimensionality" in that datapoints are relatively spread out in the 57-dimension feature space. Using a method such as principal component analysis (PCA) or Fisher's discriminant ratio (FDR) to reduce the dimension of the dataset may decrease convergence time without significantly affecting classifier accuracy. The "Prune" function could then be removed from the SVM classifier, potentially offsetting any cost to classifier accuracy as a result of dimensionality reduction.

Problem 2 - Fisher Discriminant Analysis Classifier

(a) Linear FDA

i. Classification Error: **12.56%** (87.44% Accuracy)

ii. Parameters:
The weight vector $\boldsymbol{w}$ is determined by Fisher's linear discriminant:

$$\boldsymbol{w} = (\boldsymbol{\mu_1} - \boldsymbol{\mu_2})^T \boldsymbol{S_w}^{-1} \tag{1}$$

Where $\boldsymbol{\mu_1},\ \boldsymbol{\mu_2}$ represent the class means and $\boldsymbol{S_w}^{-1}$ is the inverse of the within-class scatter matrix. Such a ratio seeks to maximise between-class variance and minimise within-class variance. As a result, the weight vector is such that the projection of $\boldsymbol{x}$ onto $\boldsymbol{w}$ allows for optimal linear separation of the two classes based on a threshold c in a reduced dimension space (for 2 features, dimension of the projection = 1).

The threshold value c is determined by:

$$c = \boldsymbol{w}^T(\frac{1}{2}(\boldsymbol{\mu_1} + \boldsymbol{\mu_2})) \tag{2}$$

This estimate determines the "halfway" point between the class means as projected onto the weight vector, which is a reasonable estimate for the discriminant if the distributions of the classes are expected to be similar. More optimal thresholds may be generated with additional assumptions or constraints regarding the dataset.

iii. Plot: *See Figure 1 - following page*

(b) Principal Component Analysis
The first linear principal component is plotted as the purple line in Figure 1. It is parallel to the discriminant line, which is orthogonal to the projection vector. Intuitively, the weight vector is such that the projection of $\boldsymbol{x}$ onto $\boldsymbol{w}$ optimises class separability - in other words, $\boldsymbol{w}$ has direction to maximise between-class variance and minimise within-class variance.

Conversely, the first principal component of the dataset is the eigenvector associated with the largest eigenvalue of the matrix $\boldsymbol{S_w}^{-1}\boldsymbol{S_b}$ which represents the greatest amount of variance of the dataset. This vector has direction which minimises class separability, which is orthogonal to the vector that optimises class separability and therefore also orthogonal to the projection vector.

The first principal component is therefore parallel to the discriminant line, but is plotted without an intercept (threshold) as only the direction of the vector is of interest in terms of dimensionality reduction.

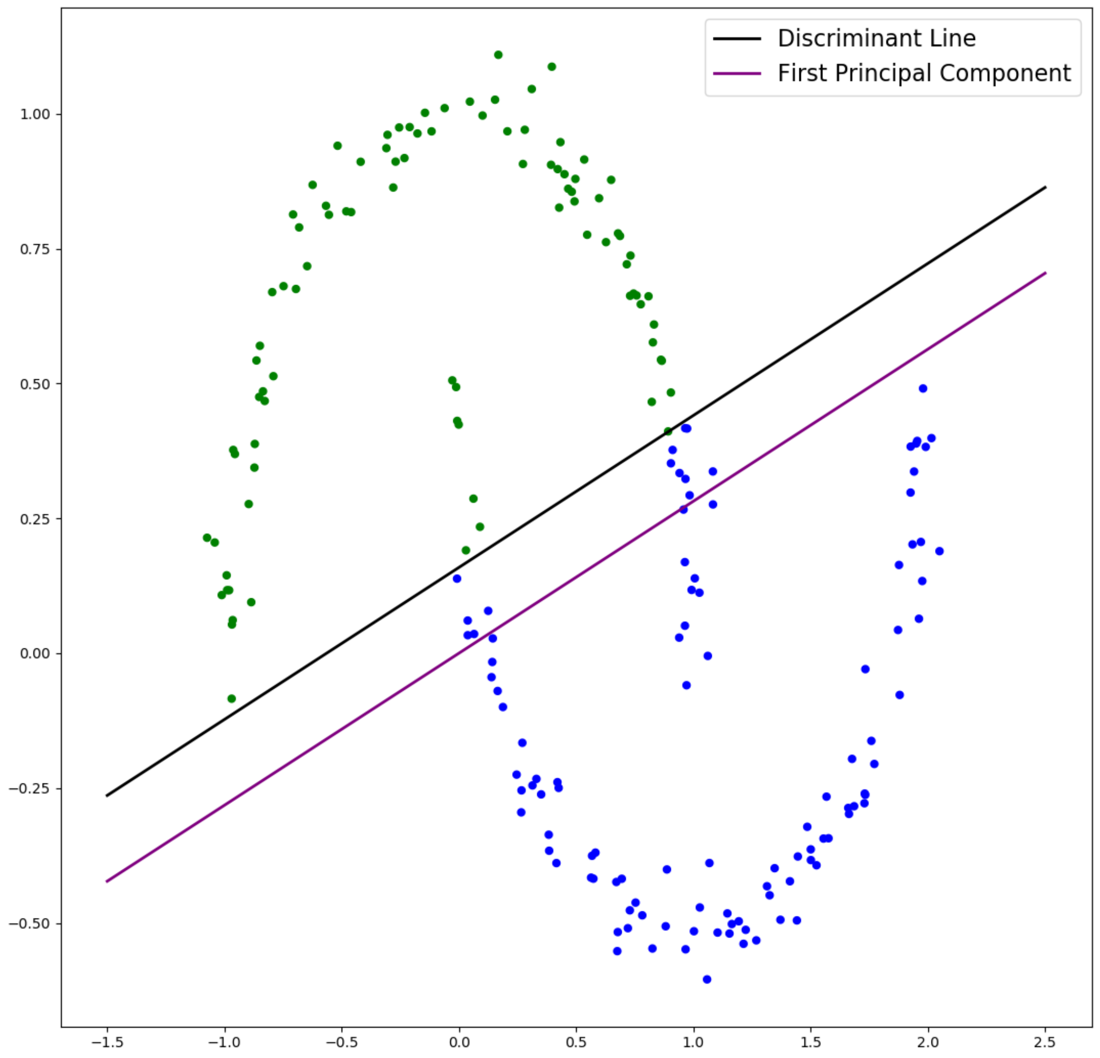

Figure 1: Fisher Discriminant Analysis for ”Moons” dataset

(c) Kernel Trick

The feature space of the ”Moons” dataset is 2-dimensional. If the data were linearly separable, a weight vector $\boldsymbol{w}$ could be generated such that the inner product $\boldsymbol{w^T x}$ separates data points above or below a threshold w_0 according to that point's classification.

This dataset is not linearly separable in the feature space, so the kernel trick is used to extend the classifier's ability to create non-linear decision boundaries. It accomplishes this by establishing a transformation $\Phi(\boldsymbol{x})$ which maps $\boldsymbol{x}$ to a higher dimension, the Hilbert space $\boldsymbol{\mathcal{H}}$. It is difficult or perhaps impossible to compute this transformation directly, but it is possible to compute inner products between vectors in $\mathcal{H}$, which is all that is needed for the sake of the classifier. The problem then becomes finding a weight vector $\boldsymbol{\alpha}$ such that each data point can be classified based on a weighted sum of inner products between the training data set $\boldsymbol{x_i}$ and each new data point $\boldsymbol{x_t}$:

$$\boldsymbol{\hat{w}^T \phi(x_t)} = \sum_{i=1}^{N} \alpha_i k(\boldsymbol{x_i}, \boldsymbol{x_t}) \tag{3}$$

The goal of the FDA classifier remains to project the data points onto a weighting vector that maximises the Fisher Discriminant Ratio. This means maximising between-class variance with respect to a fixed value for within-class variance. This can be accomplished in a manner analogous to the eigendecomposition problem of non-kernalised FDA, but here the matrices $\boldsymbol{S_w}$ and $\boldsymbol{S_b}$ are respectively replaced with $\boldsymbol{N}$ and $\boldsymbol{M}$:

$$\boldsymbol{N} = \boldsymbol{K}_1(\boldsymbol{I}_{N_1} - \mathbf{1}_{\boldsymbol{N_1}})\boldsymbol{K}_1^T + \boldsymbol{K}_2(\boldsymbol{I}_{N_2} - \mathbf{1}_{\boldsymbol{N_2}})\boldsymbol{K}_2^T \tag{4}$$

$$\boldsymbol{M} = (\boldsymbol{m}_1 - \boldsymbol{m}_2)(\boldsymbol{m}_1 - \boldsymbol{m}_2)^T \tag{5}$$

$\boldsymbol{N}$ is now a function of kernal matrices $\boldsymbol{K_1}$ and $\boldsymbol{K_2}$, and $\boldsymbol{M}$ is a function of kernalised mean vectors for the two classes. The essential functions of the original matrices are preserved, with $\boldsymbol{N}$ representing the within-class variance and $\boldsymbol{M}$ representing the between-class variance of the dataset in $\boldsymbol{\mathcal{H}}$. The derivation of these two matrices thus uses the kernel trick to accomplish the desired eigendecomposition resulting in the weight vector $\boldsymbol{\alpha}$.

(d) Cost Function Optimisation
The cost function that must be maximised is:

$$J(\boldsymbol{\alpha}) = \frac{\boldsymbol{\alpha^T M \alpha}}{\boldsymbol{\alpha^T N \alpha}} \tag{6}$$

The optimisation problem requires the denominator to be fixed as a constant:

$$\boldsymbol{\alpha^T N \alpha} = c \tag{7}$$

By maximising (6) with this constraint, the resulting expression will represent the maximum ratio of between-class variance with respect to fixed within-class variance. By the method of Lagrange:

$$L = \boldsymbol{\alpha^T M \alpha} - \lambda(\boldsymbol{\alpha^T N \alpha} - c) \tag{8}$$

Taking the derivative of L with respect to $\boldsymbol{\alpha}$ and setting equal to 0:

$$\nabla_{\boldsymbol{\alpha}} L = 2\boldsymbol{M\alpha} - 2\lambda\boldsymbol{N\alpha} = 0 \tag{9}$$

$$\Rightarrow \boldsymbol{M\alpha} = \lambda\boldsymbol{N\alpha} \tag{10}$$

Applying matrix multiplication of $\boldsymbol{N^{-1}}$ to both sides of the equation:

$$\boldsymbol{N^{-1} M \alpha} = \lambda\boldsymbol{\alpha} \tag{11}$$

Which indicates that $\boldsymbol{\alpha}$ will be an eigenvector of $\boldsymbol{N^{-1} M}$ corresponding to eigenvalue λ. Multipling both sides of (10) by $\boldsymbol{\alpha^T}$:

$$\max_{\boldsymbol{\alpha}} \boldsymbol{\alpha^T M \alpha} = \max_{\boldsymbol{\alpha}} \lambda\boldsymbol{\alpha^T N \alpha} \tag{12}$$

By (7), the denominator expression of the original quotient (6) is equal to c:

$$\max_{\boldsymbol{\alpha}} \boldsymbol{\alpha^T M \alpha} = \max_{\boldsymbol{\alpha}} \lambda c \tag{13}$$

Since c is arbitrary, this demonstrates that the expression for the cost function is maximised by selecting the largest eigenvalue λ of $\boldsymbol{N^{-1} M}$, for which $\boldsymbol{\alpha}$ will be the corresponding eigenvector. This is the eigendecomposition problem of interest for generation of the weight vector $\boldsymbol{\alpha}$ for a FDA classifier.

(e) Kernel FDA Implementation

i. Kernel Choice and Parameters:
The KFDA classifier was implemented with the Gaussian and Polynomial kernels, reliably achieving 100% classifier accuracy for the "Moons" dataset over a variety of parameters and training set sizes.

To ensure robustness of the classifier, a random split is applied to the shuffled data. This allows training of a classifier that can deal with different distributions of subsets of the dataset as opposed to one that may be optimised towards a skewed sample. It also allows for deviation in the proportion of class labels for the training dataset which furthers the demand for classifier robustness.

The Gaussian kernel:

$$k(\boldsymbol{x}, \boldsymbol{x_i}) = \exp \frac{||\boldsymbol{x} - \boldsymbol{x_i}||^2}{2h^2} \tag{14}$$

Here, h is set to 0.75, though values between 0 and 1 reliably produced 100% accuracy. A range of values were tested between 0 and 20, with lower values producing fewer classification errors.

The polynomial kernel:

$$k(\boldsymbol{x}, \boldsymbol{x_i}) = (a\boldsymbol{x}^T\boldsymbol{x_i} + b)^p \tag{15}$$

There is an assumption made based on the shape of the data. The moons appear to be such that an odd-degree polynomial curve could be constructed to "manually" separate the data. For this reason, p is selected as 3. The default values of $a, b = 1$ reliably produced 100% accuracy and did not require extensive testing.

ii. Comments and Comparison to Linear FDA:

As expected for a dataset composed of non-linearly separable data, the kernalised version of the FDA classifier significantly out-performed the linear classifier. Linear FDA makes an honest attempt at optimising class separability of the data, but the limitations of the feature space will always restrict the classifier from achieving 100% accuracy.

Applying the kernel trick to the FDA classifier and using a reasonable kernel function produces desirable results. In addition to the achievement of 100% accuracy, the decision boundary for the dataset appears to yield relatively equivalent margins to the two distributions when using the Gaussian kernel. This result is theoretically optimal under the assumption that the cost of misclassification of new datapoints is equal for both classes.

iii. Plots: *Figures 2, 3*

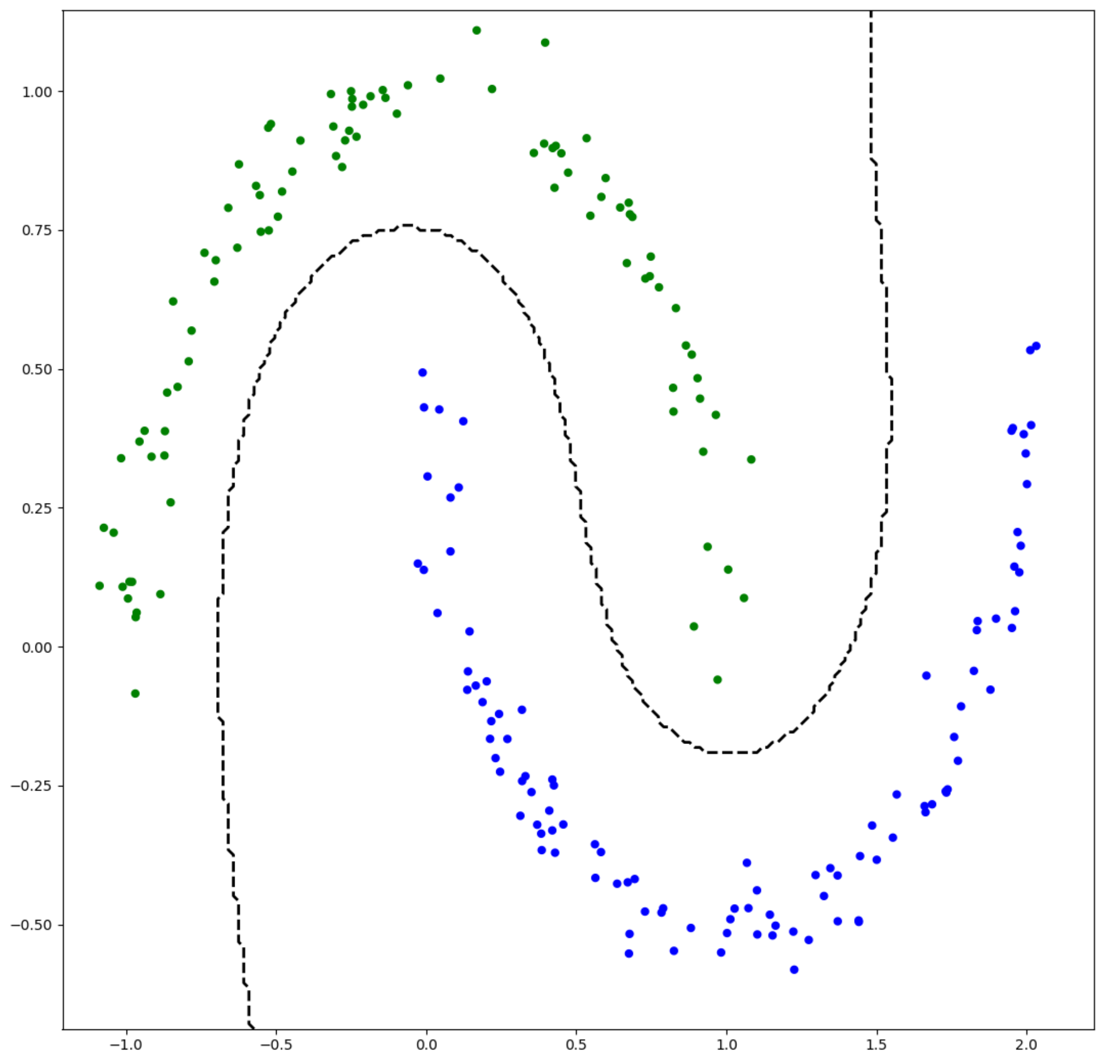

Figure 2: Kernalised FDA for "Moons" dataset, Gaussian kernel

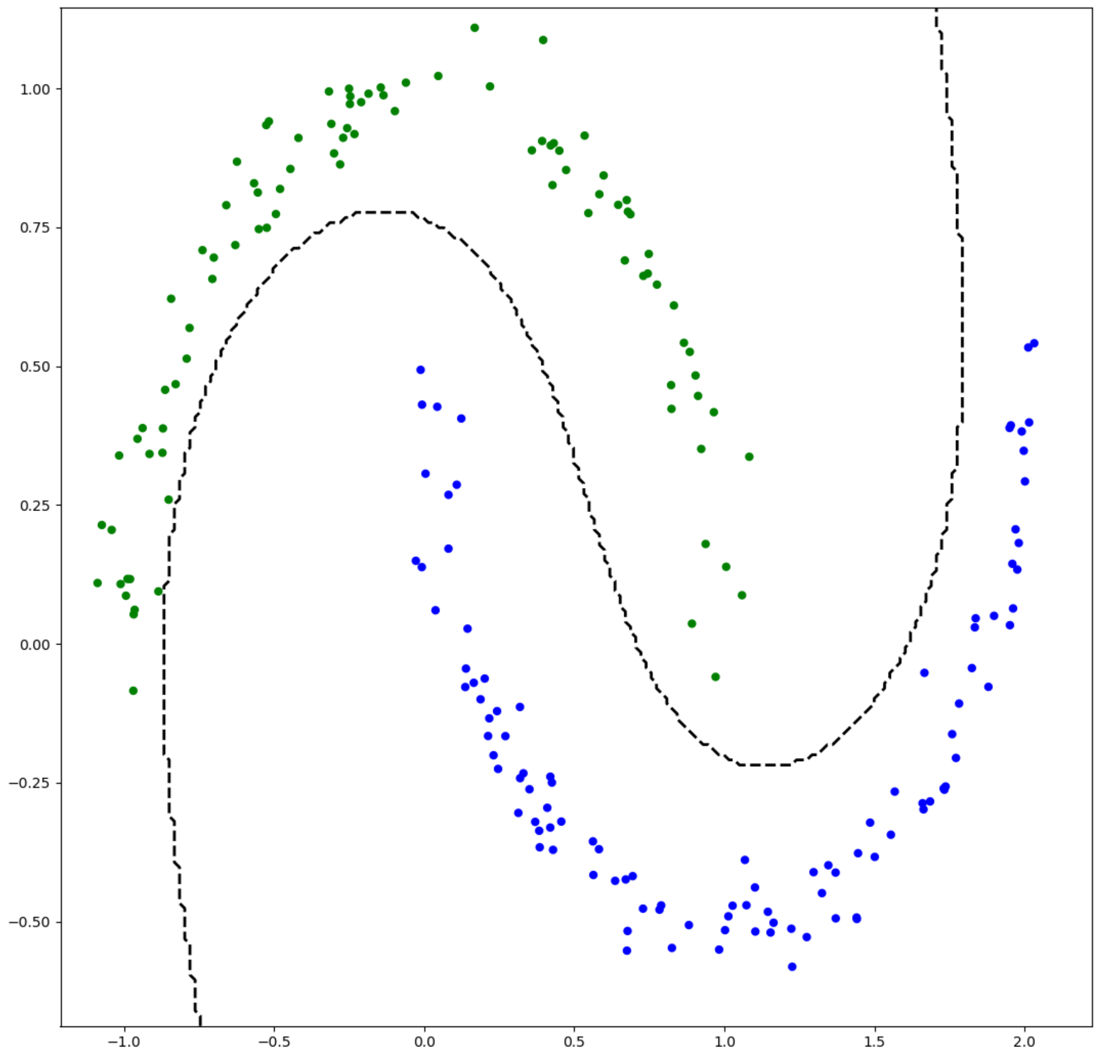

Figure 3: Kernalised FDA for "Moons" dataset, polynomial kernel