ISYE 6740 HW 5

Mohammed Khalid Siddiqui

October 22, 2024

1 Comparing multi-class classifiers for handwritten digits classification.

1. The below tables show the respective output for precision, recall, and F1-Score for each of the classes in the handwritten digits dataset with the classification models used as columns.

Digit	SVM	RBF	KNN	LGM	MLP
0	0.936832	0.965139	0.949070	0.952286	0.948156
1	0.956373	0.974848	0.917886	0.962738	0.973822
2	0.925743	0.959804	0.982143	0.928215	0.945259
3	0.884652	0.927481	0.922255	0.905512	0.916587
4	0.910521	0.961224	0.957974	0.936928	0.935743
5	0.890315	0.959584	0.930023	0.895402	0.921700
6	0.945361	0.967842	0.962129	0.942089	0.959916
7	0.940535	0.964567	0.923892	0.934975	0.965932
8	0.918860	0.961094	0.970252	0.882474	0.923958
9	0.925358	0.942886	0.879812	0.911417	0.945399

Table 1: Precision for each digit using SVM, RBF, KNN, LGM, and MLP classifiers.

Digit	SVM	RBF	KNN	LGM	MLP
0	0.983673	0.988776	0.988776	0.977551	0.970408
1	0.985022	0.990308	0.994714	0.978855	0.983260
2	0.906008	0.948643	0.906008	0.902132	0.937016
3	0.918812	0.962376	0.939604	0.910891	0.946535
4	0.942974	0.959267	0.905295	0.937882	0.949084
5	0.855381	0.931614	0.923767	0.873318	0.923767
6	0.957203	0.973904	0.981211	0.950939	0.949896
7	0.923152	0.953307	0.932879	0.923152	0.937743
8	0.860370	0.938398	0.870637	0.878850	0.910678
9	0.896928	0.932607	0.928642	0.917740	0.926660

Table 2: Recall for each digit using SVM, RBF, KNN, LGM, and MLP classifiers.

Digit	SVM	RBF	KNN	LGM	MLP
0	0.959681	0.976815	0.968516	0.964753	0.959153
1	0.970486	0.982517	0.954757	0.970730	0.978518
2	0.915769	0.954191	0.942540	0.914988	0.941119
3	0.901408	0.944606	0.930848	0.908193	0.931320
4	0.926463	0.960245	0.930890	0.937405	0.942366
5	0.872499	0.945392	0.926884	0.884222	0.922732
6	0.951245	0.970864	0.971576	0.946494	0.954879
7	0.931762	0.958904	0.928364	0.929026	0.951629
8	0.888653	0.949610	0.917749	0.880658	0.917270
9	0.910921	0.937718	0.903568	0.914568	0.935936

Table 3: F1-score for each digit using SVM, RBF, KNN, LGM, and MLP classifiers.

Below are the respective confusion matrices for the five classification algorithms performance on the test dataset:

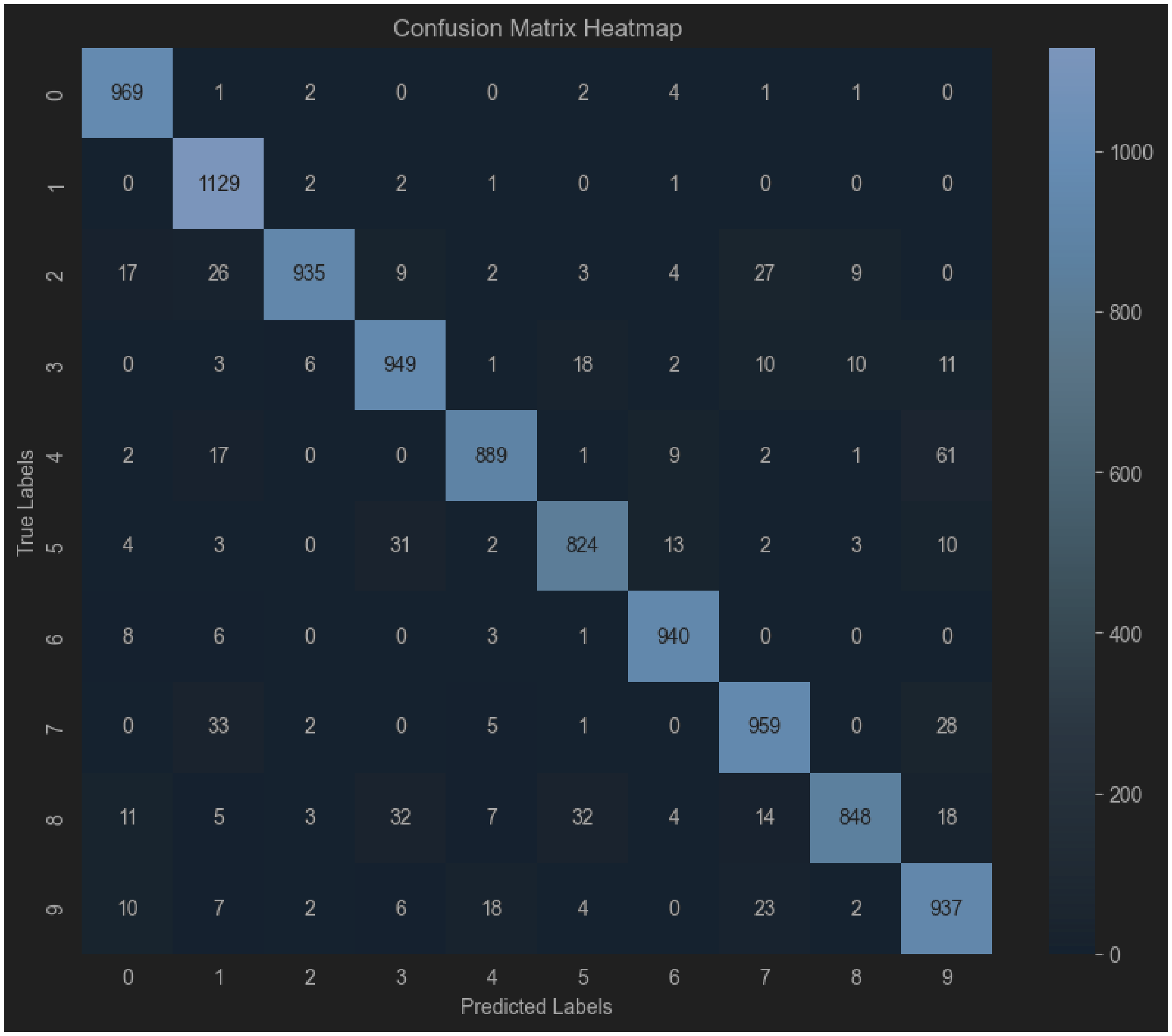
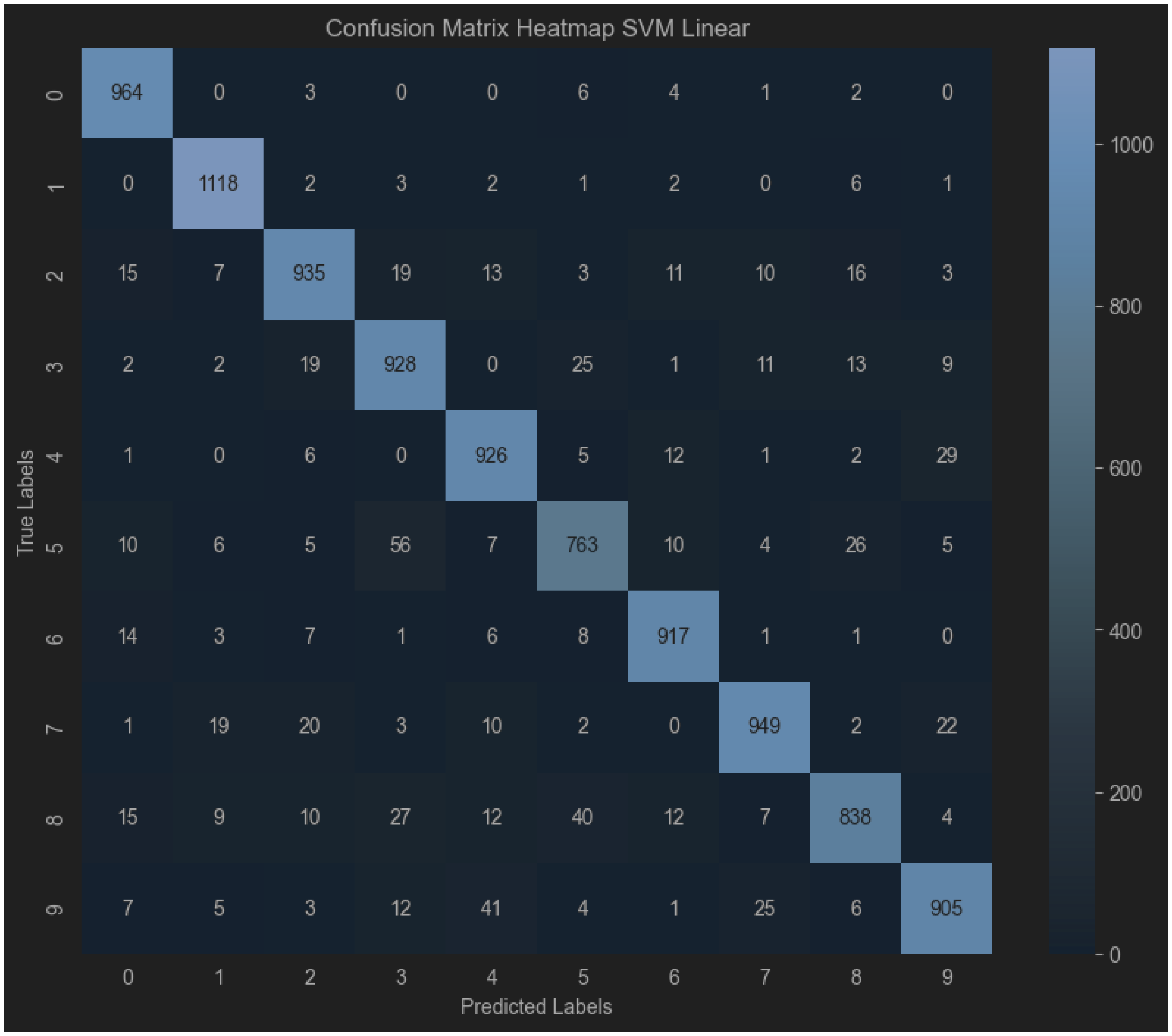
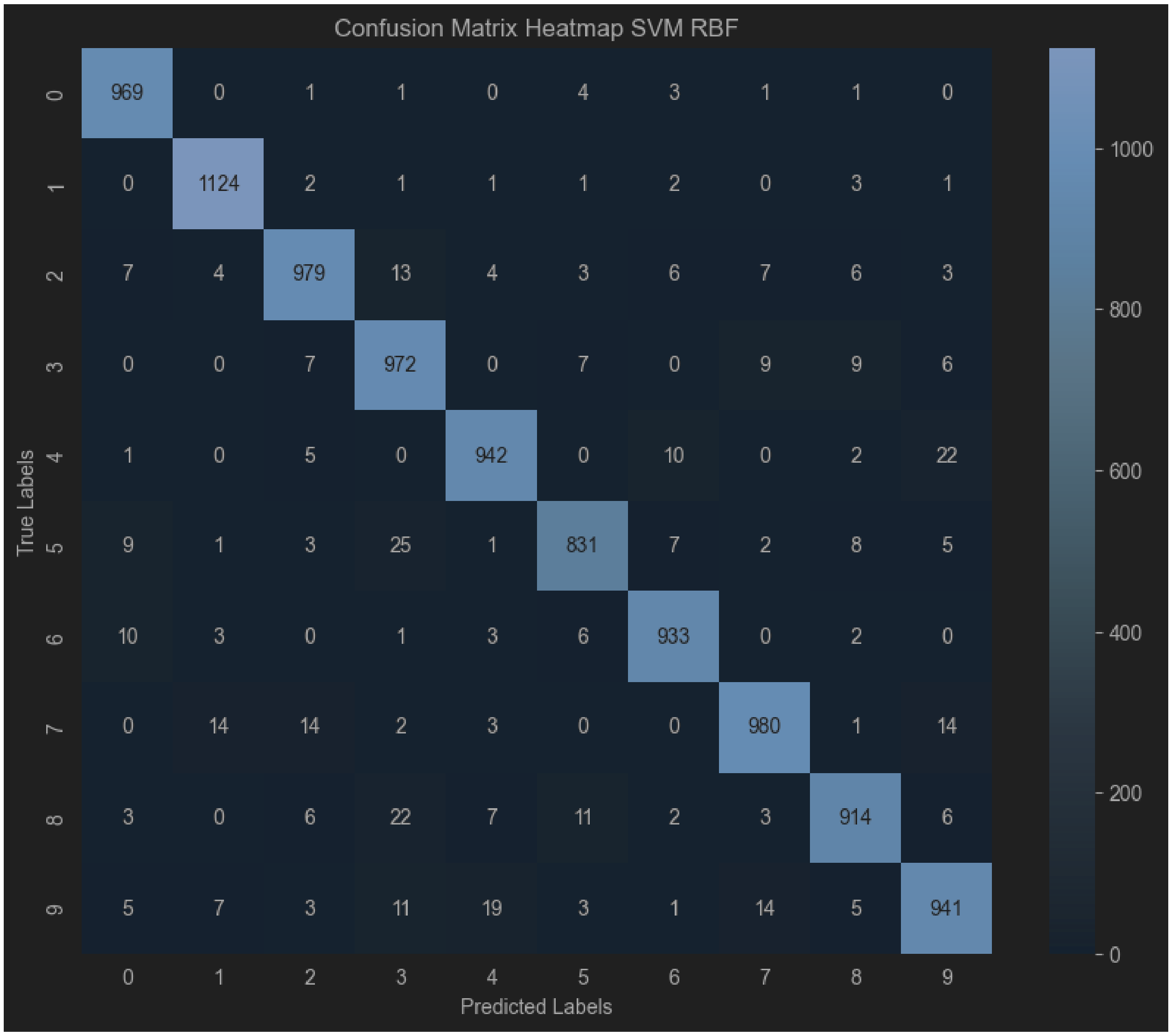


Figure 1: KNN Confusion matrix



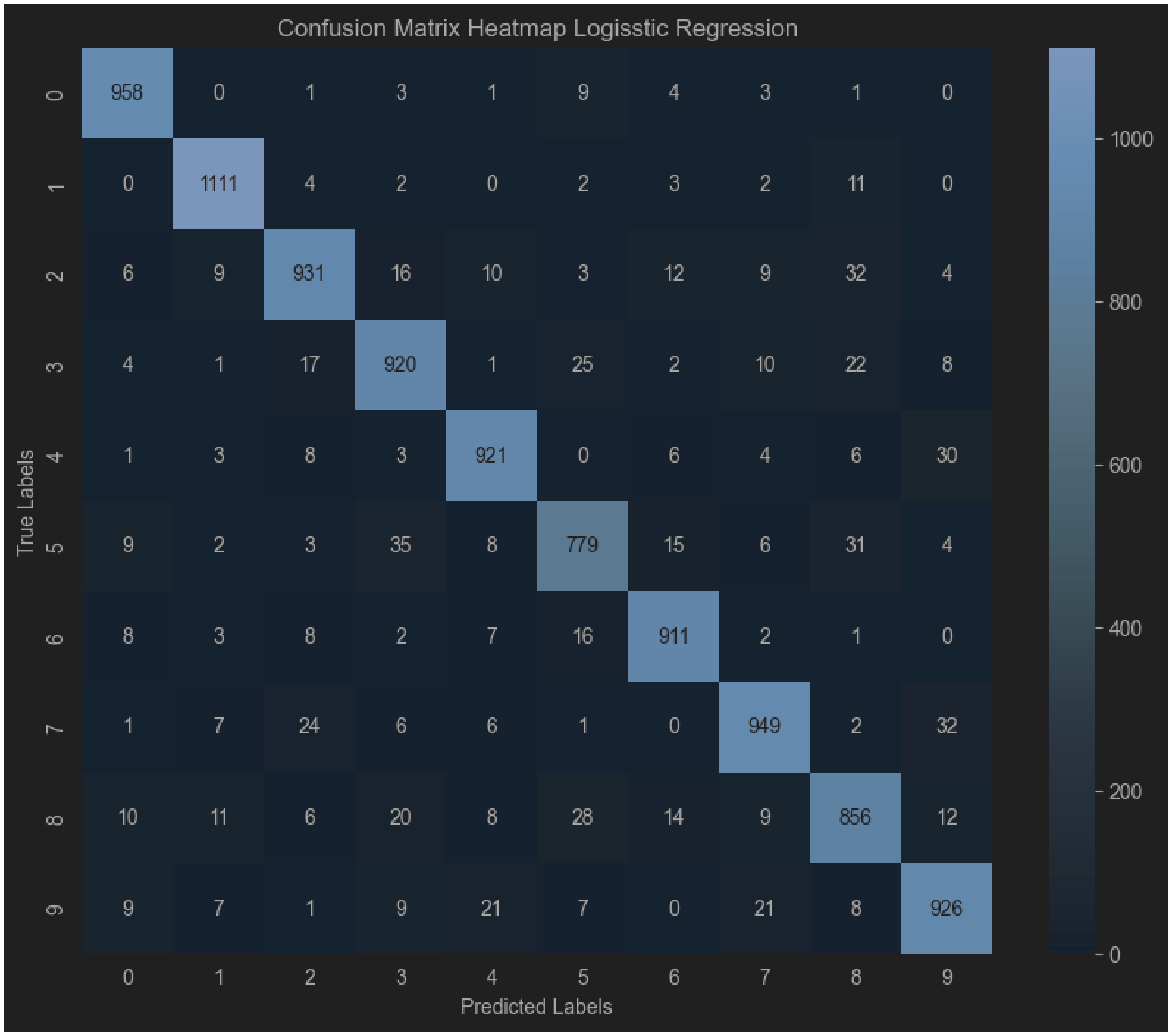
	0	1	2	3	4	5	6	7	8	9
0	964	0	3	0	0	6	4	1	2	0
1	0	1118	2	3	2	1	2	0	6	1
2	15	7	935	19	13	3	11	10	16	3
3	2	2	19	928	0	25	1	11	13	9
4	1	0	6	0	926	5	12	1	2	29
5	10	6	5	56	7	763	10	4	26	5
6	14	3	7	1	6	8	917	1	1	0
7	1	19	20	3	10	2	0	949	2	22
8	15	9	10	27	12	40	12	7	838	4
9	7	5	3	12	41	4	1	25	6	905

Figure 2: SVM Confusion Matrix



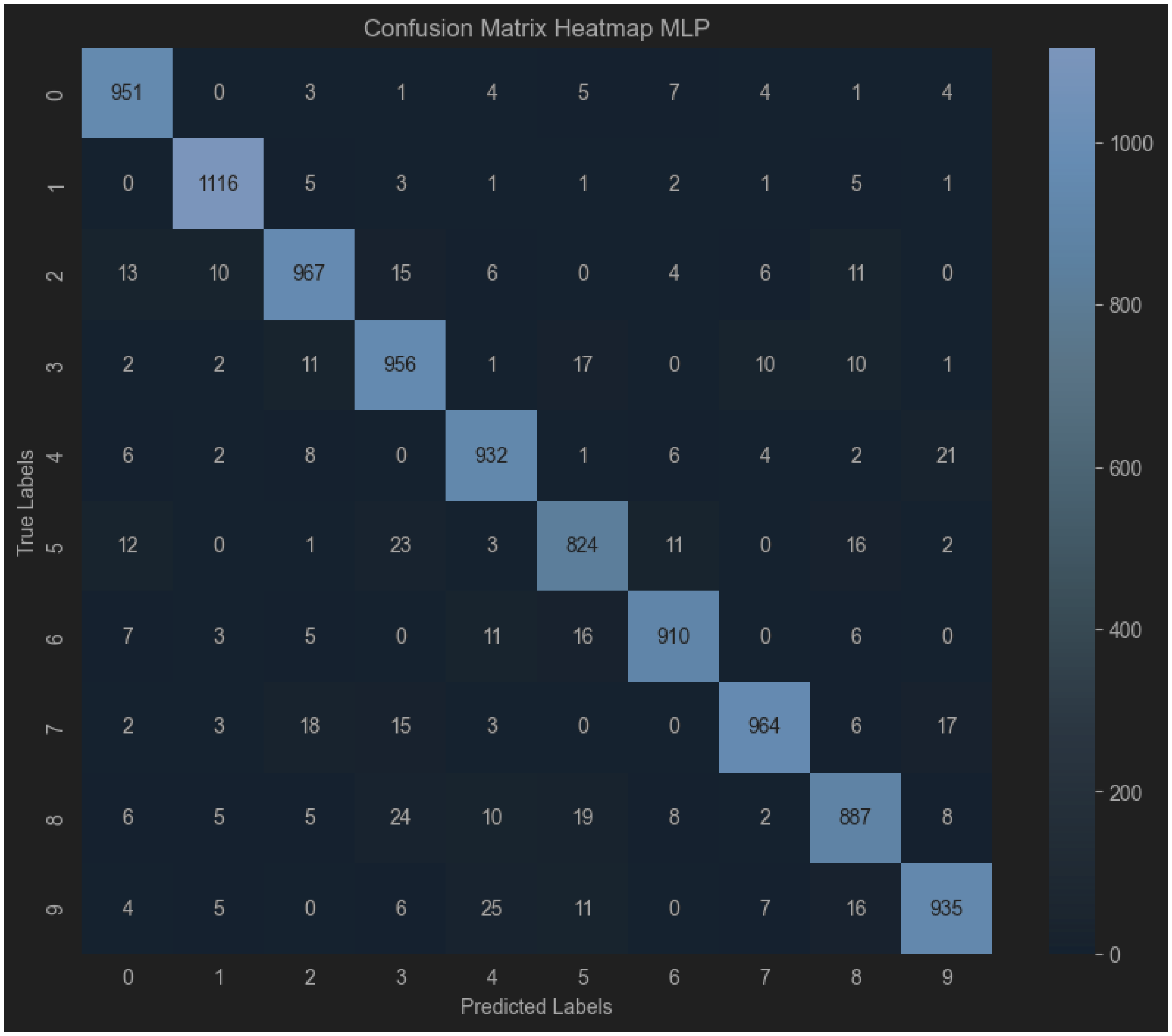
	0	1	2	3	4	5	6	7	8	9
0	969	0	1	1	0	4	3	1	1	0
1	0	1124	2	1	1	1	2	0	3	1
2	7	4	979	13	4	3	6	7	6	3
3	0	0	7	972	0	7	0	9	9	6
4	1	0	5	0	942	0	10	0	2	22
5	9	1	3	25	1	831	7	2	8	5
6	10	3	0	1	3	6	933	0	2	0
7	0	14	14	2	3	0	0	980	1	14
8	3	0	6	22	7	11	2	3	914	6
9	5	7	3	11	19	3	1	14	5	941

Figure 3: SVM RBF Kernel Confusion Matrix



	0	1	2	3	4	5	6	7	8	9
0	958	0	1	3	1	9	4	3	1	0
1	0	1111	4	2	0	2	3	2	11	0
2	6	9	931	16	10	3	12	9	32	4
3	4	1	17	920	1	25	2	10	22	8
4	1	3	8	3	921	0	6	4	6	30
5	9	2	3	35	8	779	15	6	31	4
6	8	3	8	2	7	16	911	2	1	0
7	1	7	24	6	6	1	0	949	2	32
8	10	11	6	20	8	28	14	9	856	12
9	9	7	1	9	21	7	0	21	8	926

Figure 4: Logistic Regression Confusion Matrix



	0	1	2	3	4	5	6	7	8	9
0	951	0	3	1	4	5	7	4	1	4
1	0	1116	5	3	1	1	2	1	5	1
2	13	10	967	15	6	0	4	6	11	0
3	2	2	11	956	1	17	0	10	10	1
4	6	2	8	0	932	1	6	4	2	21
5	12	0	1	23	3	824	11	0	16	2
6	7	3	5	0	11	16	910	0	6	0
7	2	3	18	15	3	0	0	964	6	17
8	6	5	5	24	10	19	8	2	887	8
9	4	5	0	6	25	11	0	7	16	935

Figure 5: MultiLayer Perceptron Confusion Matrix

2. With respect to the F1-Score, it seems that our SVM with Radial Basis Function kernel performed among the best across all 9 digits while our Multi-layer Perceptron performed exceptionally well for the digit 9 with respect to the other classification algorithms though had generally inconsistent performance across classes perhaps due to overfitting certain classes or underfitting others. The exceptional performance of our SVM RBF could be attributed to its ability to capture complex non-linear relationships in the data. Conversely SVM and Logistic Regression (LGM) being linear classifiers may not perform as well on the non-linear data. KNN seems to have more issues where digits have closely resembling neighbors such as 3 and 5 but performs exceptionally well when there is stronger distance separation between classes such as with 2 and 0.

2 SVM

1. The goal when implementing a Support Vector Machine is to find a hyperplane (for ex: in two dimensional space, a line) that separates two classes with the largest possible margin. We can represent the decision boundary as wT * x + b = 0. wT is the Transpose of the weight vector and b represents the bias term.

   The functional margin for a data point is given by:

   $$\gamma_i = y_i(w^T x_i + b) \tag{1}$$

   The geometric margin (the orthogonal distance between a point and the hyperplane) is

   $$\text{Geometric Margin} = \frac{\gamma_i}{\|w\|} \tag{2}$$

   To simplify, one can set the functional margin of the support vectors to 1

   $$y_i(w^T x_i + b) = 1 \tag{3}$$

   Therefore

   $$\min_{w,b} \frac{1}{2}\|w\|^2 \quad \text{subject to} \quad y_i(w^T x_i + b) \geq 1, \forall i \tag{4}$$

   The 1/2 (or constant 2 in the corresponding maximization problem) can be dropped as the scale and proportion the constant induces won't change the resulting wand b values. Similarly, we were able to set the margin c to 1 to fix the scaling and reduce ambiguity in our optimization problem. We can do this for similar reasons, as the weight vector and bias w, b respectively can be scaled without changing the direction of the hyperplane.

2. To derive this, we first take the Lagrangian function for the primal SVM optimization problem. We then take the partial derivatives of L with respect to w and b to address the KKT conditions for a saddle point (point where Langrange has pair of primal and dual optimal solutions).

   $$L(w, b, \alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{n} \alpha_i [y_i(w^T x_i + b) - 1] \tag{5}$$

   $$\frac{\partial L}{\partial w} = w - \sum_{i=1}^{n} \alpha_i y_i x_i = 0 \tag{6}$$

   $$\frac{\partial L}{\partial b} = -\sum_{i=1}^{n} \alpha_i y_i = 0 \tag{7}$$

   $$\sum_{i=1}^{n} \alpha_i y_i = 0 \tag{8}$$

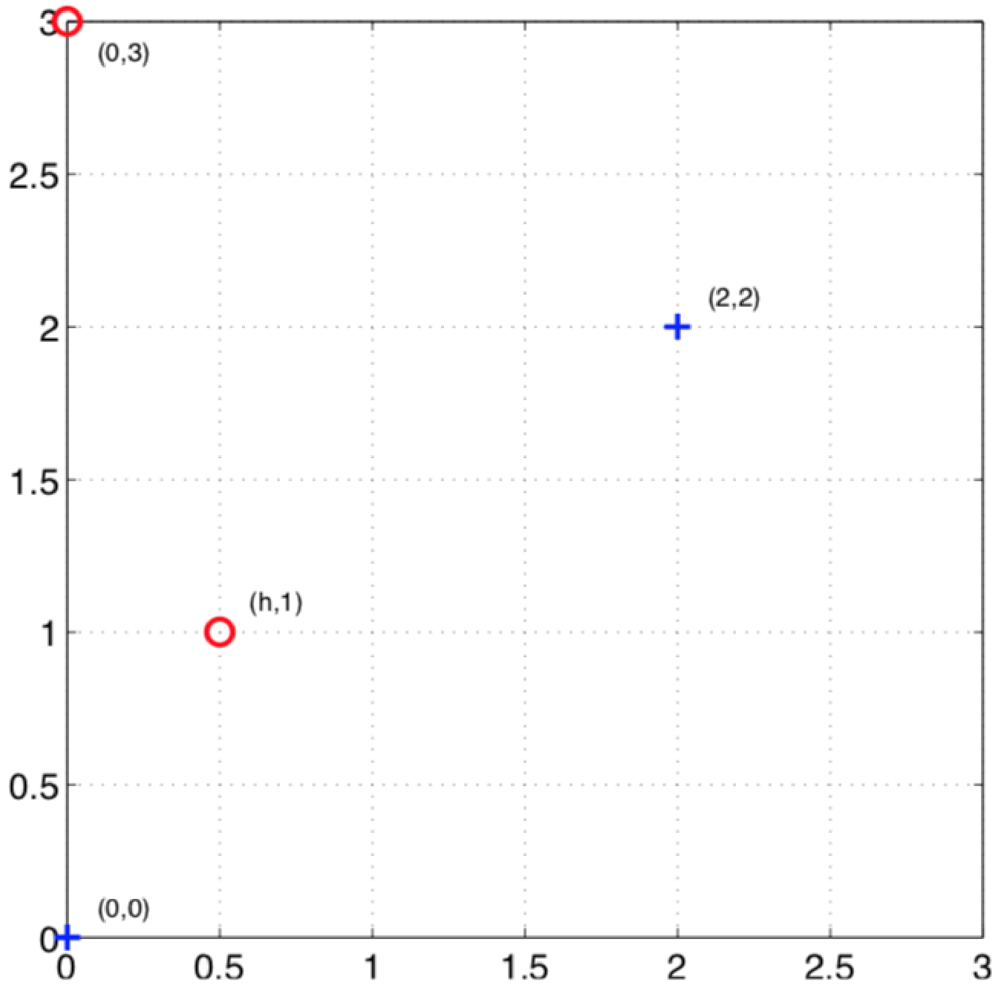
   Only the data points with a non-zero alpha (support vectors) contribute to defining the decision boundary. The other data points have an alpha of 0 and do not influence the weight vector w.

3. Using the derivations above and remembering the KKT conditions, we know that for non-support vectors, alpha must be 0 and for support vectors, alpha must be greater than 0. Support vectors are where

$$y_i(w^T x_i + b) = 1 \tag{9}$$

Only data points which lie exactly on the margin will have alpha greater than 0 and thus contribute to the weight vector w. Other points, further from the margin, have alpha equal to 0 and do not affect the weight vectors w.

4. We'll need to check if there's a linear decision boundary which separates the negative and positive samples in the shown image. The positive and negative points should lie on opposite sides of the hyperplane (line as this is two-dimensional).



a) For separability, we must find the respective maximum value of h such that h would be on the same side as the other negative class and not cross over into the boundary of the positive classes. To do this, we can first create a hyperplane boundary assuming points x1 and x2 are at the margin and therefore defining the hyperplane.

The equation of a line is y = ax + b where the scalar a stands in place of our weight vector w in multi-dimensional spaces. To find the respective weight a, let's take the slope of the line between points x1 and x2 as defined by (y2-y1)/(x2-x1) which results in a slope of 1. Plugging in our slope into our equation as well as point x1 results in a b of 0. The equation of our linearly separable line is therefore y = x. Assuming both x1 and x2 are on the decision boundary whereas x3 and x4 are not means all space to the right and under the line connecting them in the positive class and all points above and to the left are the negative class. This can be represented as the positive class have y ¡ x and positive class having y ¿ x To satisfy the condition that x3 must be to the left and higher than our decision boundary, h ¡ y of x3, therefore (0 ¡ h ¡ 1).

b) The maximum margin decision boundary orientation does indeed change as h changes when the points are separable. From our previous problem (0 ¡ h ¡ 1). Our initial decision boundary where we assumed x1 and x2 on the boundary itself found itself as y = x where the bias term b = 0. For the maximum decision boundary, we use a set of inequalities to help separate the negative and positive classes optimally.

$$w^T x_1 + b \geq 1 \quad \text{for positive points} \tag{10}$$

$$w^T x_3 + b \leq -1 \quad \text{for negative points} \tag{11}$$

$$\text{Using x1 to get the bias term: } w_1 \cdot 0 + w_2 \cdot 0 + b = 1 \quad \Rightarrow \quad b = 1 \tag{12}$$

$$w_1 h + w_2 \cdot 1 + b = -1 \quad \Rightarrow \quad w_1 h + w_2 + 1 = -1 \tag{13}$$

$$w_1 h + w_2 = -2 \tag{14}$$

$$w = (w_1, -2 - w_1 h) \tag{15}$$

$$\text{Slope of the boundary} = \frac{-w_1}{w_2} = \frac{-w_1}{-2 - w_1 h} = \frac{w_1}{2 + w_1 h} \tag{16}$$

The orientation of the decision boundary is determined by the direction of the weight vector w. As h increases towards 1, the slope of decision boundary w becomes less steep. As h decreases, the slope of the decision boundary becomes more steep.

3 Neural networks and backpropagation

1. The cost function is shown below

$$\ell(w, \alpha, \beta) = \sum_{i=1}^{m} (y_i - \sigma(w^T z_i))^2 \tag{17}$$

The sigmoid function is given by:

$$\sigma(x) = \frac{1}{1+e^{-x}} \tag{18}$$

z is a 2-dimensional vector where zi1 is sigma of alphaT*xi and zi2 is sigma of betaT*xi.

$$u_i = w^T z_i \tag{19}$$

$$\ell(w,\alpha,\beta) = \sum_{i=1}^{m} (y_i - \sigma(u_i))^2 \tag{20}$$

$$\frac{\partial \ell(w,\alpha,\beta)}{\partial w} = \sum_{i=1}^{m} \frac{\partial}{\partial w} \left((y_i - \sigma(u_i))^2\right) \tag{21}$$

$$= \sum_{i=1}^{m} 2(y_i - \sigma(u_i)) \cdot \frac{\partial}{\partial w}(-\sigma(u_i)) \tag{22}$$

$$\frac{\partial u_i}{\partial w} = z_i \tag{23}$$

$$\frac{\partial \sigma(u_i)}{\partial u_i} = \sigma(u_i)(1-\sigma(u_i)) \tag{24}$$

$$\frac{\partial \ell(w,\alpha,\beta)}{\partial w} = -\sum_{i=1}^{m} 2(y_i - \sigma(u_i))\sigma(u_i)(1-\sigma(u_i))z_i \tag{25}$$

By taking the partial derivative and implementing the chain rule as well as substitution, we can find the gradient ofL with respect to w.

2. The gradients with respect to alpha and beta follow a similar process. Differentiate with respect to the parameter requested, then apply the necessary chain rule and substitutions from the known sigmoid function derivative.

   For the gradient of L with respect to alpha:

$$\frac{\partial \ell(w,\alpha,\beta)}{\partial \alpha} = \sum_{i=1}^{m} 2(y_i - \sigma(u_i)) \cdot \frac{\partial}{\partial \alpha}\sigma(u_i) \tag{26}$$

$$\frac{\partial \sigma(u_i)}{\partial \alpha} = \sigma(u_i)(1-\sigma(u_i)) \cdot w_1 \cdot \frac{\partial z_{i1}}{\partial \alpha} \tag{27}$$

$$\frac{\partial z_{i1}}{\partial \alpha} = \sigma(\alpha^T x_i)(1-\sigma(\alpha^T x_i))x_i \tag{28}$$

$$\frac{\partial \ell(w,\alpha,\beta)}{\partial \alpha} = -\sum_{i=1}^{m} 2(y_i - \sigma(u_i))\sigma(u_i)(1-\sigma(u_i))w_1\sigma(\alpha^T x_i)(1-\sigma(\alpha^T x_i))x_i \tag{29}$$

   For the gradient of L with respect to Beta:

$$\frac{\partial \ell(w,\alpha,\beta)}{\partial \beta} = \sum_{i=1}^{m} 2(y_i - \sigma(u_i)) \cdot \frac{\partial}{\partial \beta}\sigma(u_i) \tag{30}$$

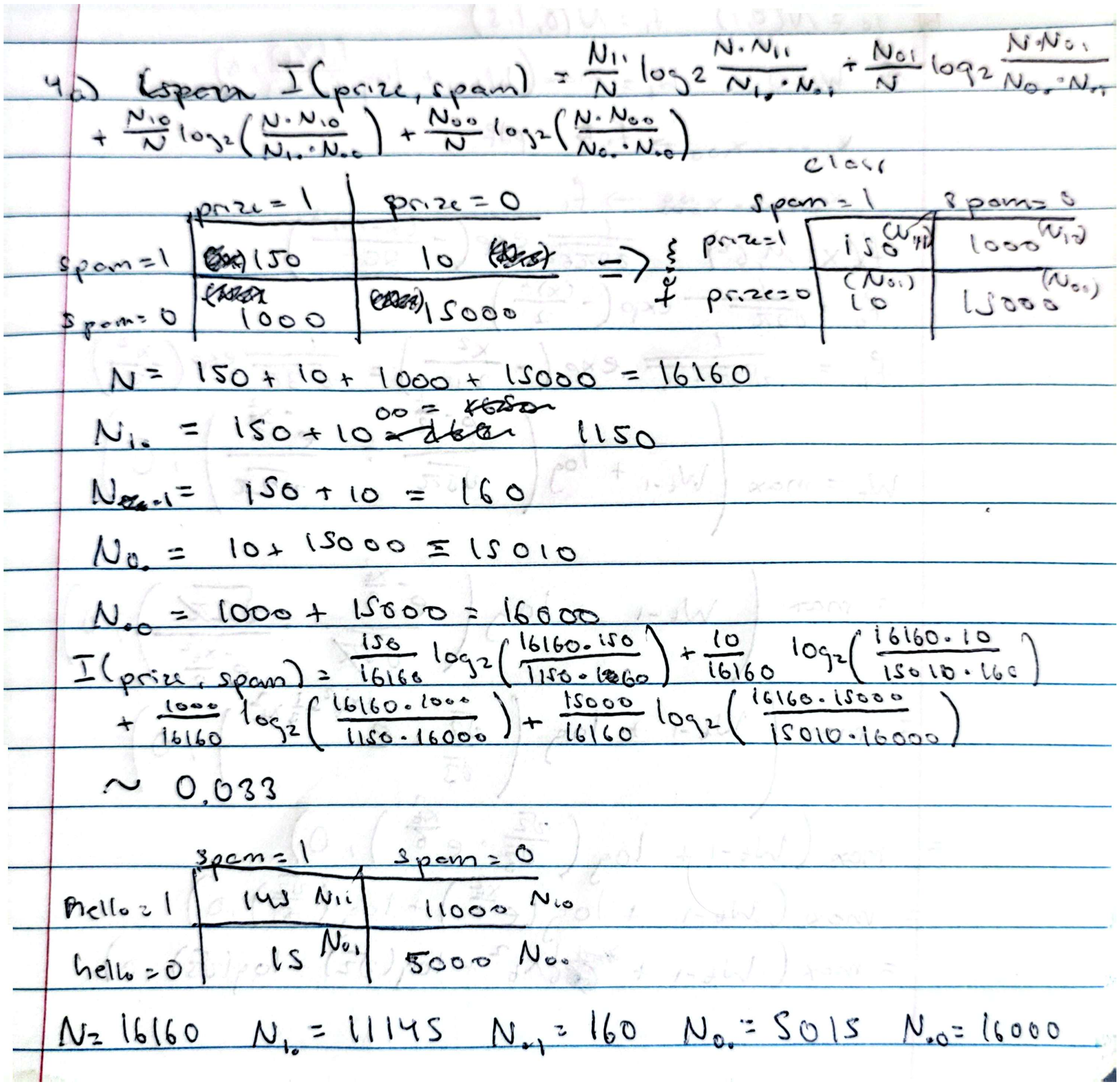
$$\frac{\partial \sigma(u_i)}{\partial \beta} = \sigma(u_i)(1-\sigma(u_i)) \cdot w_2 \cdot \frac{\partial z_{i2}}{\partial \beta} \tag{31}$$

$$\frac{\partial z_{i2}}{\partial \beta} = \sigma(\beta^T x_i)(1-\sigma(\beta^T x_i))x_i \tag{32}$$

$$\frac{\partial \ell(w,\alpha,\beta)}{\partial \beta} = -\sum_{i=1}^{m} 2(y_i - \sigma(u_i))\sigma(u_i)(1-\sigma(u_i))w_2\sigma(\beta^T x_i)(1-\sigma(\beta^T x_i))x_i \tag{33}$$

4 Feature selection and change-point detection

1. Mutual Information is calculated as I(X,Y) = H(Y) - H(Y—X). H(Y) is the entropy of Y and H(Y—X) is the conditional entropy of Y given X. The mutual information for this particular problem can be calculated in the same method as the example from the course lecture, as shown below. From comparing the caclulated mutual information for both I(Hello, Spam) and I(Prize, spam), "Prize" seems to be the more informative keyword for deciding whether or not an email is spam:



4a) ~~spam~~ $I(\text{prize}, \text{spam}) = \frac{N_{11}}{N} \log_2 \frac{N \cdot N_{11}}{N_{1\cdot} \cdot N_{\cdot 1}} + \frac{N_{01}}{N} \log_2 \frac{N \cdot N_{01}}{N_{0\cdot} \cdot N_{\cdot 1}}$

$+ \frac{N_{10}}{N} \log_2\left(\frac{N \cdot N_{10}}{N_{1\cdot} \cdot N_{\cdot 0}}\right) + \frac{N_{00}}{N} \log_2\left(\frac{N \cdot N_{00}}{N_{0\cdot} \cdot N_{\cdot 0}}\right)$

	prize = 1	prize = 0
spam = 1	150	10
spam = 0	1000	15000

$\Longrightarrow$

		class	
		spam = 1	spam = 0
term	prize = 1	150 (N_{11})	1000 (N_{10})
	prize = 0	(N_{01}) 10	15000 (N_{00})

$N = 150 + 10 + 1000 + 15000 = 16160$

$N_{1\cdot} = 150 + 1000 = \cancel{16250}\ \cancel{1160}\ 1150$

$N_{\cdot 1} = 150 + 10 = 160$

$N_{0\cdot} = 10 + 15000 = 15010$

$N_{\cdot 0} = 1000 + 15000 = 16000$

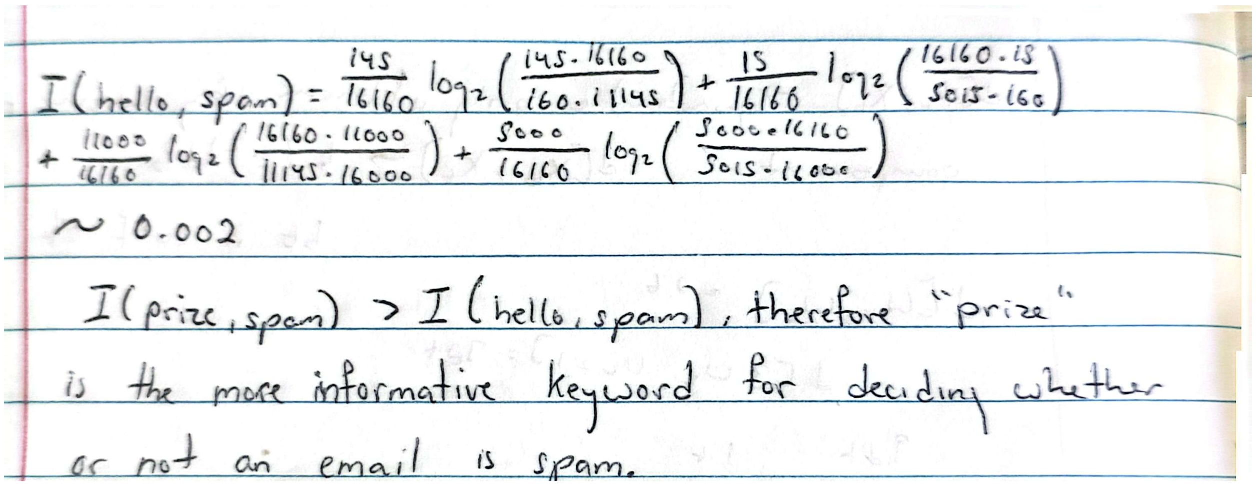
$I(\text{prize}, \text{spam}) = \frac{150}{16160} \log_2\left(\frac{16160 \cdot 150}{1150 \cdot 160}\right) + \frac{10}{16160} \log_2\left(\frac{16160 \cdot 10}{15010 \cdot 160}\right)$

$+ \frac{1000}{16160} \log_2\left(\frac{16160 \cdot 1000}{1150 \cdot 16000}\right) + \frac{15000}{16160} \log_2\left(\frac{16160 \cdot 15000}{15010 \cdot 16000}\right)$

~ 0.033

	spam = 1	spam = 0
hello = 1	145 N_{11}	11000 N_{10}
hello = 0	15 N_{01}	5000 N_{00}

$N = 16160 \quad N_{1\cdot} = 11145 \quad N_{\cdot 1} = 160 \quad N_{0\cdot} = 5015 \quad N_{\cdot 0} = 16000$



$$I(\text{hello}, \text{spam}) = \frac{145}{16160}\log_2\left(\frac{145 \cdot 16160}{160 \cdot 11145}\right) + \frac{15}{16166}\log_2\left(\frac{16160 \cdot 15}{5015 \cdot 160}\right)$$
$$+ \frac{11000}{16160}\log_2\left(\frac{16160 \cdot 11000}{11145 \cdot 16000}\right) + \frac{5000}{16160}\log_2\left(\frac{5000 \cdot 16160}{5015 \cdot 16000}\right)$$
$$\sim 0.002$$

$I(\text{prize}, \text{spam}) > I(\text{hello}, \text{spam})$, therefore "prize" is the more informative keyword for deciding whether or not an email is spam.

2. The below image shows the generated random variate X values where the first 100 follow a random N(0,1) distribution and the next 50 follow N(0, sqrt(1.5)). You can see the greater troughs and peaks for points after 100.

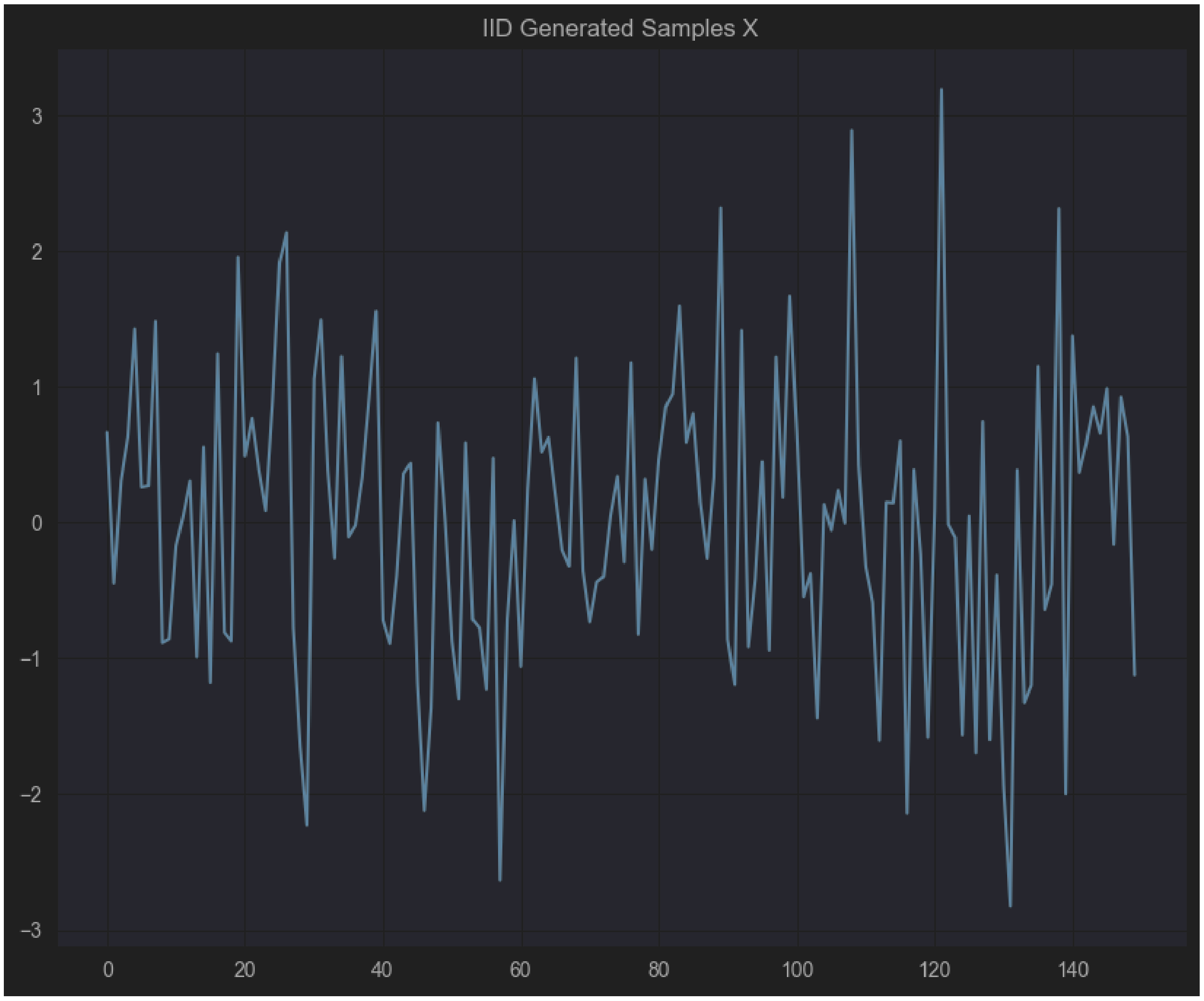


Figure 6: Generated X values

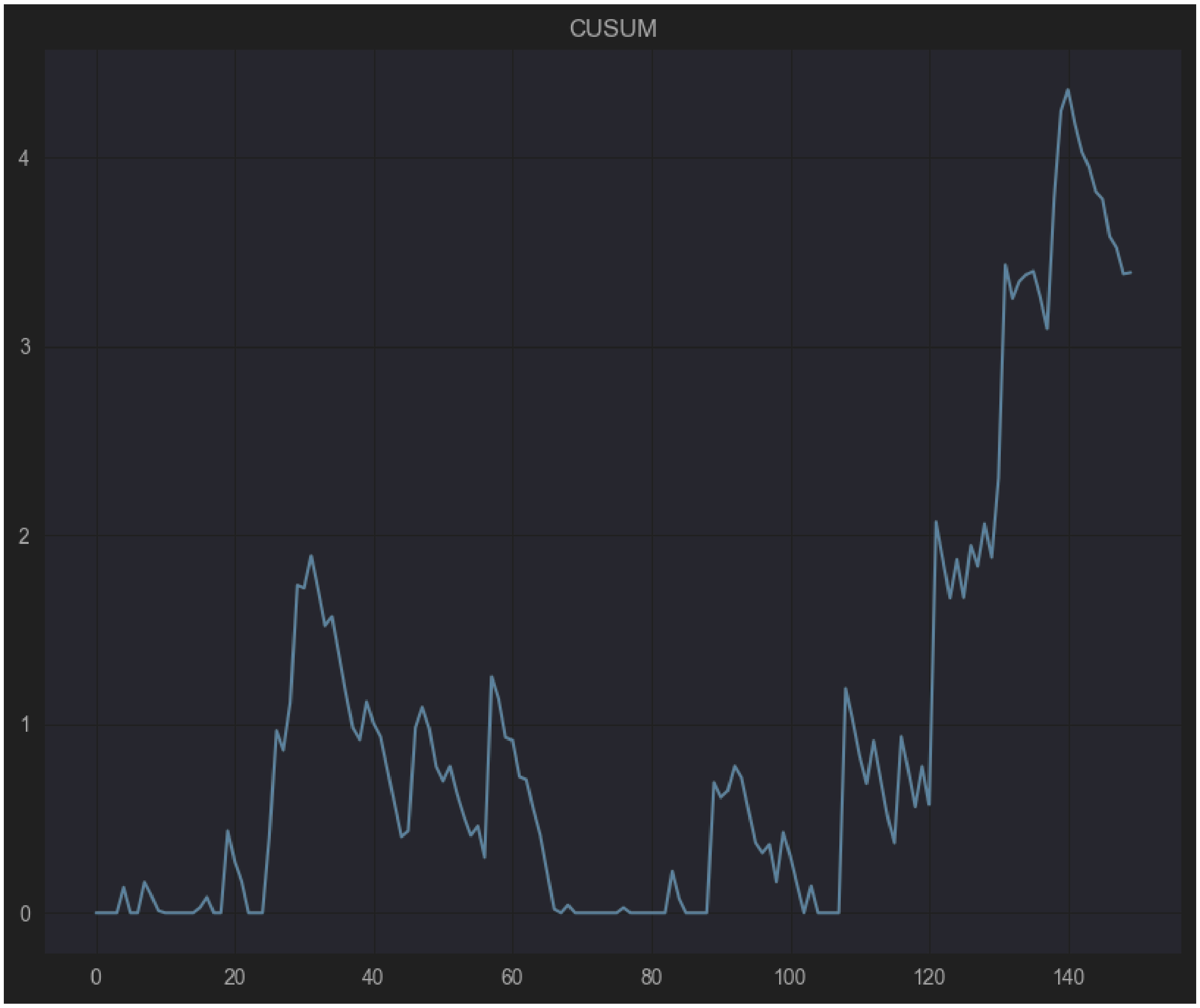


Figure 7: CUSUM

Based on the plot, depending on a specific threshold you set, you can approximate when a change is detected. From simply eyeballing the plot, it appears a change may be detected around t = 125 or so.

b) $f_0 = N(0,1) \qquad f_1 = N(0,1.5)$

$W_0 = 0 \qquad W_t = \max\left(W_{t-1} + \log\frac{f_1(x_t)}{f_0(x_t)}, 0\right)$

$x_1 \dots x_{100} \rightarrow f_0$ pdf

$x_{101} \dots x_{150} \rightarrow f_1$

$$f(x \mid \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

$$f_0 = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(x)^2}{2}\right)$$

$$f_1 = \frac{1}{\sqrt{2\pi \cdot 1.5}} \exp\left(-\frac{x^2}{2 \cdot 1.5}\right) = \frac{1}{\sqrt{3\pi}} \exp\left(-\frac{x^2}{3}\right)$$

$$W_t = \max\left(W_{t-1} + \log\left(\frac{e^{-\frac{x_t^2}{3}}}{\sqrt{3\pi}} \div \frac{e^{-\frac{x_t^2}{2}}}{\sqrt{2\pi}}\right), 0\right)$$

$$= \max\left(W_{t-1} + \log\left(\frac{e^{-\frac{x_t^2}{3}}}{\sqrt{3\cancel{\pi}}} \cdot \frac{\sqrt{2\cancel{\pi}}}{e^{-\frac{x_t^2}{2}}}\right), 0\right)$$

$$= \max\left(W_{t-1} + \log\left(\frac{\sqrt{2}}{\sqrt{3}} \cdot e^{\frac{1}{2}x_t^2 - \frac{1}{3}x_t^2}\right), 0\right)$$

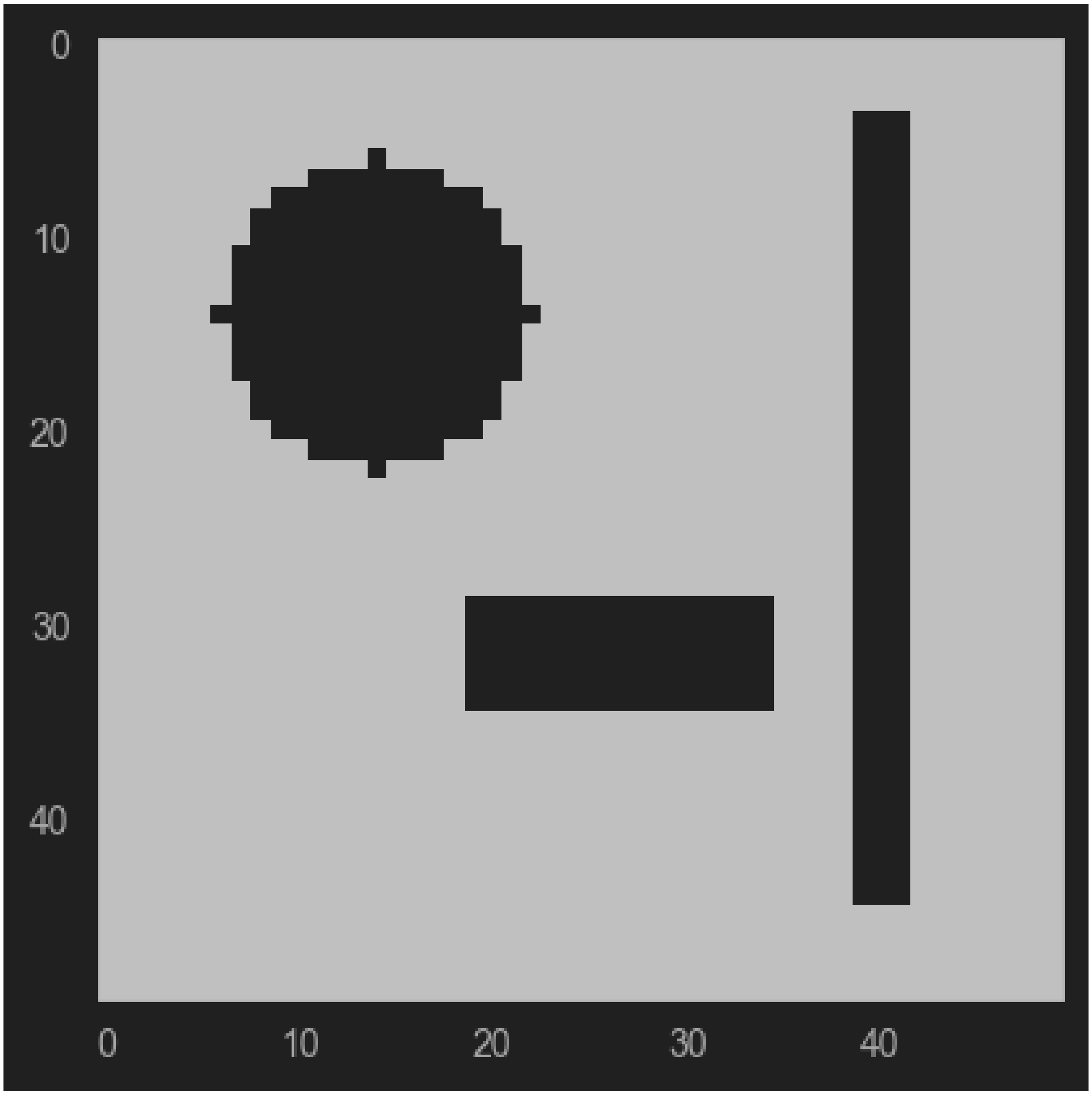
$$= \max\left(W_{t-1} + \log\left(\frac{\sqrt{2}}{\sqrt{3}} \cdot e^{\frac{x_t^2}{6}}\right), 0\right)$$

$$= \max\left(W_{t-1} + \log\left(e^{\frac{x_t^2}{6}}\right) + \log\left(\frac{\sqrt{2}}{\sqrt{3}}\right), 0\right)$$

$$= \max\left(W_{t-1} + \frac{1}{6}x_t^2 + \log(\sqrt{2}) - \log(\sqrt{3}), 0\right)$$

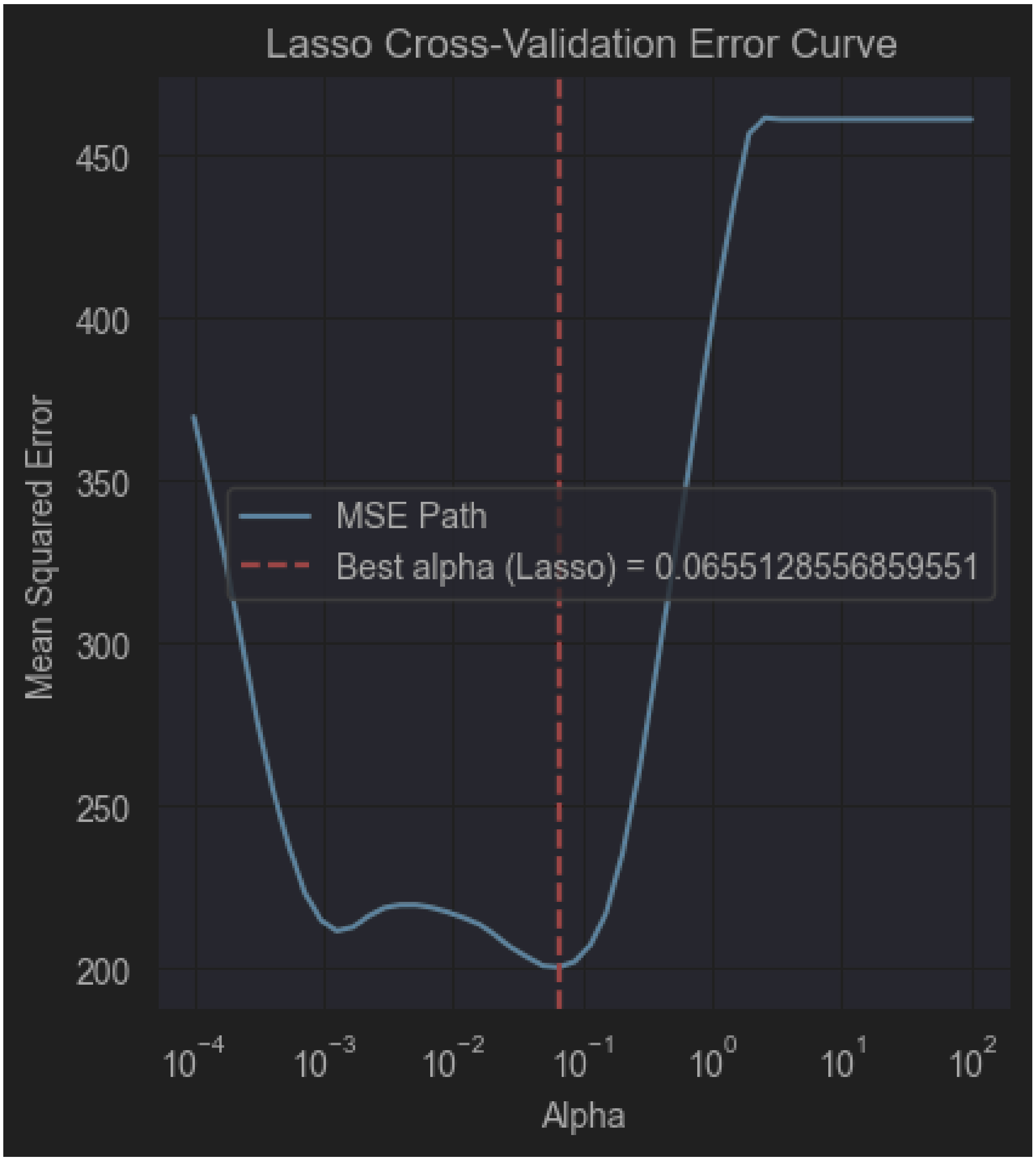
5 Medical imaging reconstruction

The original image can be shown below:



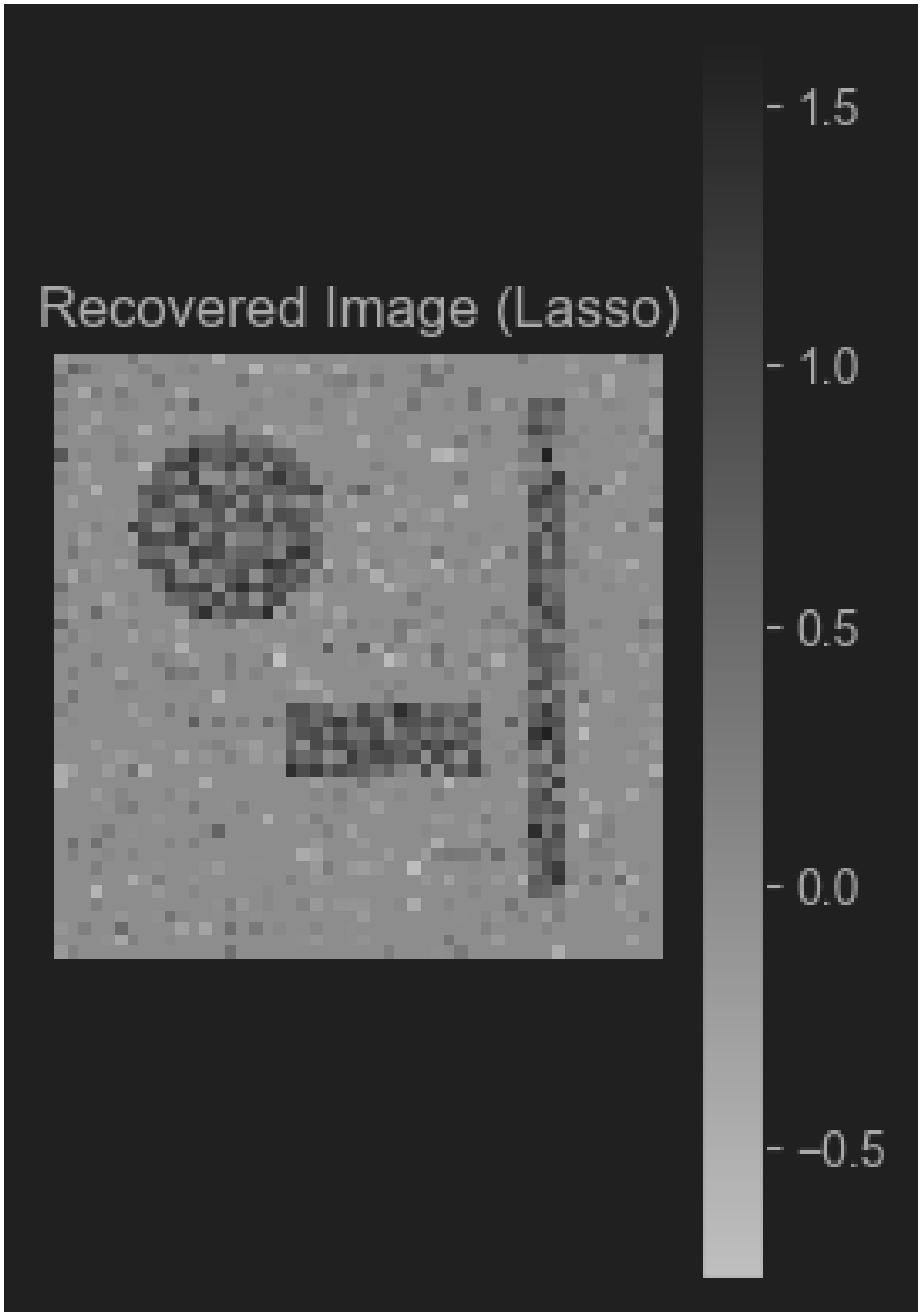
For the following two problems, Lasso and Ridge regression were used with GridSearchCV with cv = 10 to find the optimal alphas across the range of 10e-4 to 10e2 using the Mean squared Error as the metric to optimize.

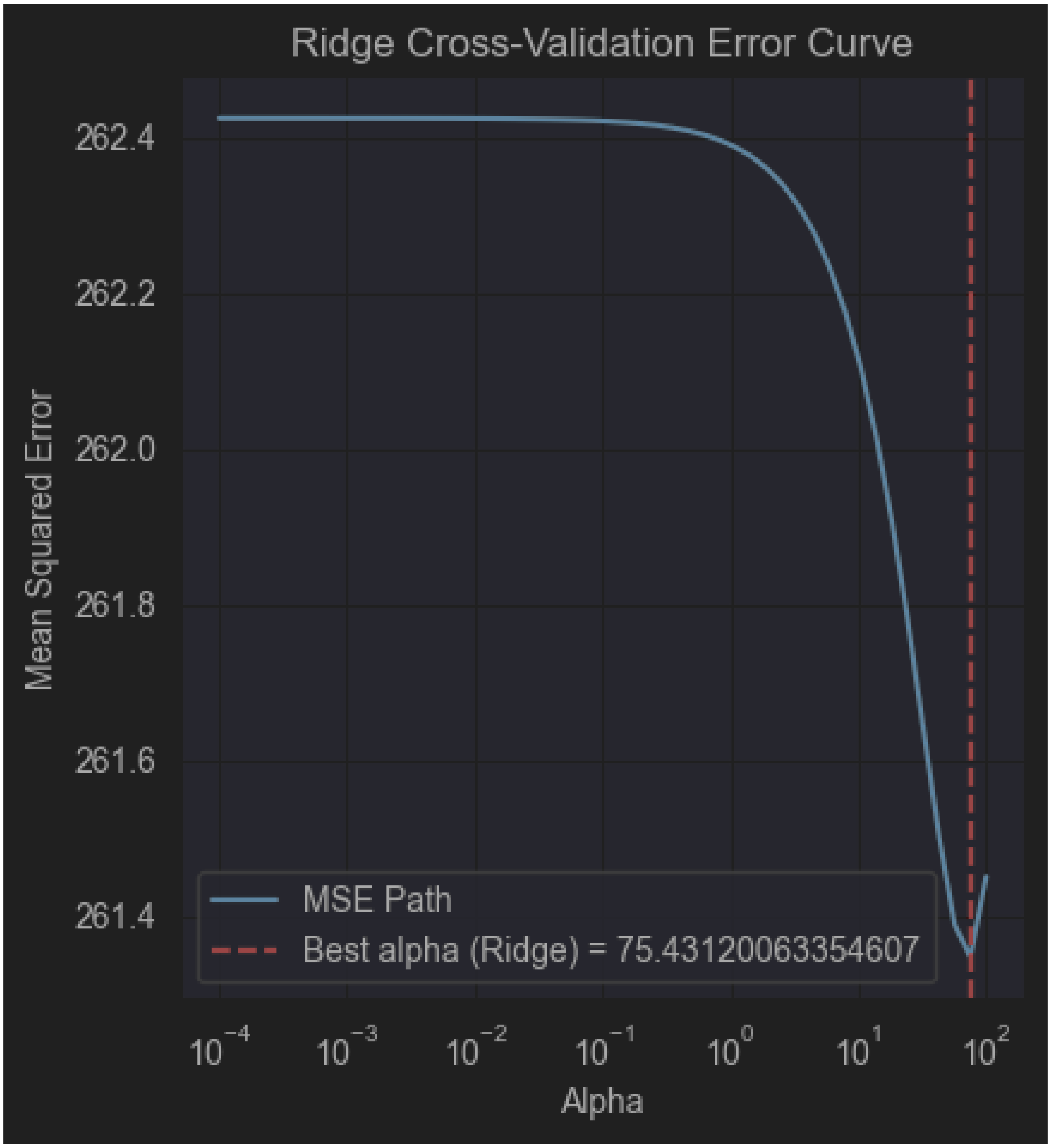
1. The Lasso Regression MSE are plotted below and it appears the optimal alpha is in the 10e-1 order of magnitude at roughly 0.0655.



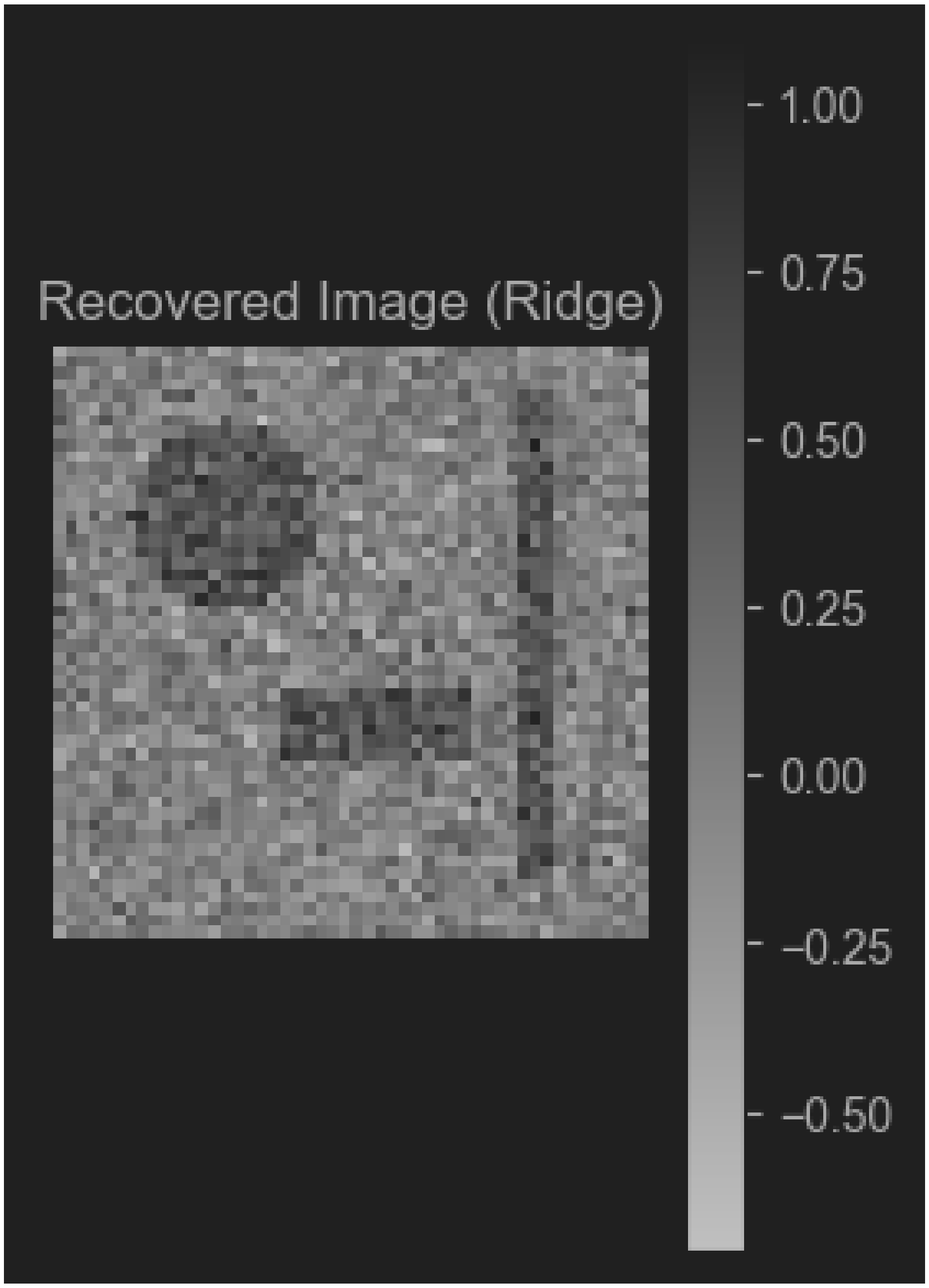
The recovered image below reshapes the coefficients of the best estimator from our Lasso CV runs.

2. The Ridge Regression Mean Squared Errors are plotted below and it appears the best alpha is 75 or between the order of 10e1 and 10e2.


Recovered Image (Lasso)
1.5
1.0
0.5
0.0
−0.5



The recovered image for Ridge regression is plotted below:



By visually comparing the results as well as by looking at the CV error plot, it appears that Lasso regression is a better option. Lasso regression can force some coefficients to 0 and generally works better with sparse data than ridge regression. The MRI was sparse. You can see the recovered image using Ridge regression has far more noise outside of the core objects from the MRI whereas the Lasso recovered image more clearly highlights the objects from the original image.

Sources

1. https://scikit-learn.org/dev/modules/generated/sklearn.neighbors.KNeighborsClassifier.html
2. https://scikit-learn.org/1.5/modules/generated/sklearn.linear_model.LassoCV.html
3. https://scikit-learn.org/1.5/modules/generated/sklearn.linear_model.Ridge.html
4. https://numpy.org/doc/stable/reference/random/generated/numpy.random.normal.html
5. Course Lectures and Demo Code