Homework 3

Martino Boggs
ID: 907 810 3539

Instructions: Use this latex file as a template to develop your homework. Submit your homework on time as a single pdf file to Canvas. Late submissions may not be accepted. Please wrap your code and upload to a public GitHub repo, then attach the link below the instructions so that we can access it. You can choose any programming language (i.e. python, R, or MATLAB). Please check Piazza for updates about the homework.

See code on GitHub: https://github.com/tinoboggs/CS-760/blob/main/hw2/hw2.qmd

1 Questions (50 pts)

1. (9 pts) Explain whether each scenario is a classification or regression problem. And, provide the number of data points (n) and the number of features (p).

   (a) (3 pts) We collect a set of data on the top 500 firms in the US. For each firm we record profit, number of employees, industry and the CEO salary. We are interested in predicting CEO salary with given factors.

   This scenario is a regression problem since the response (CEO salary) is a numeric variable. There are $n = 500$ data points and $p = 3$ features.

   (b) (3 pts) We are considering launching a new product and wish to know whether it will be a success or a failure. We collect data on 20 similar products that were previously launched. For each product we have recorded whether it was a success or failure, price charged for the product, marketing budget, competition price, and ten other variables.

   This scenario is a classification problem since the response (success or failure of product) is a categorical variable. There are $n = 20$ data points and $p = 13$ features.

   (c) (3 pts) We are interesting in predicting the % change in the US dollar in relation to the weekly changes in the world stock markets. Hence we collect weekly data for all of 2012. For each week we record the % change in the dollar, the % change in the US market, the % change in the British market, and the % change in the German market.

   This scenario is a regression problem since the response (% change in U.S. dollars) is a numeric variable. There are $n = 52$ data points and $p = 3$ features.

2. (6 pts) The table below provides a training data set containing six observations, three predictors, and one qualitative response variable.

X_1	X_2	X_3	Y
0	3	0	Red
2	0	0	Red
0	1	3	Red
0	1	2	Green
-1	0	1	Green
1	1	1	Red

   Suppose we wish to use this data set to make a prediction for Y when $X_1 = X_2 = X_3 = 0$ using K-nearest neighbors.

(a) (2 pts) Compute the Euclidean distance between each observation and the test point, $X_1 = X_2 = X_3 = 0$.

The Euclidean distance between an observation and the test point, $X_1 = X_2 = X_3 = 0$, is given by

$$D = \sqrt{X_1^2 + X_2^2 + X_3^2}$$

Therefore,

X_1	X_2	X_3	Y	D
0	3	0	Red	3
2	0	0	Red	2
0	1	3	Red	$\sqrt{10}$
0	1	2	Green	$\sqrt{5}$
-1	0	1	Green	$\sqrt{2}$
1	1	1	Red	$\sqrt{3}$

(b) (2 pts) What is our prediction with $K = 1$? Why?

For $K = 1$, our prediction for the test point is $Y =$ Green. Note that the smallest distance D between each observation and the test point is $D = \sqrt{2}$; therefore, the single nearest neighbor to the test point is the fifth observation with label $Y =$ Green.

(c) (2 pts) What is our prediction with $K = 3$? Why?

For $K = 3$, our prediction for the test point is $Y =$ Red. The three smallest distances D are $\sqrt{2}$, $\sqrt{3}$, and 2, corresponding to the fifth, sixth, and second observations, respectively. Note that the most common label among these three nearest neighbors is $Y =$ Red.

3. (12 pts) When the number of features p is large, there tends to be a deterioration in the performance of KNN and other local approaches that perform prediction using only observations that are near the test ob- servation for which a prediction must be made. This phenomenon is known as the curse of dimensionality, and it ties into the fact that non-parametric approaches often perform poorly when p is large.

(a) (2pts) Suppose that we have a set of observations, each with measurements on $p = 1$ feature, X. We assume that X is uniformly (evenly) distributed on [0, 1]. Associated with each observation is a response value. Suppose that we wish to predict a test observation's response using only observations that are within 10% of the range of X closest to that test observation. For instance, in order to predict the response for a test observation with $X = 0.6$, we will use observations in the range [0.55, 0.65]. On average, what fraction of the available observations will we use to make the prediction?

Suppose X is uniformly distribution on $[0, 1]$ and let $P|X$ denote the fraction of observations used for prediction; that is,

$$P|X = x := \begin{cases} x + 0.05, & 0 \le x < 0.05 \\ 0.1, & 0.05 \le x < 0.95 \\ 0.05 + (1 - x), & 0.95 \le x \le 1 \end{cases}$$

Then

$$\begin{aligned} \mathbb{E}(P|X = x) &= \int_0^{0.05} x + 0.05\, dx + \int_{0.05}^{0.95} 0.1\, dx + \int_{0.95}^{1} 0.05 + (1 - x)\, dx \\ &= 0.5x^2 + .05x \Big|_0^{0.05} + 0.1x \Big|_{0.05}^{0.95} + 1.05x - 0.5x^2 \Big|_{0.95}^{1} \\ &= 0.00375 + 0.09 + 0.00375 \\ &= 0.0975 \end{aligned}$$

Therefore, on average, we use 9.75% of the available observations to make the prediction.

(b) (2pts) Now suppose that we have a set of observations, each with measurements on $p = 2$ features, $X1$ and $X2$. We assume that predict a test observation's response using only observations that $(X1, X2)$ are uniformly distributed on $[0, 1] \times [0, 1]$. We wish to are within 10% of the range of $X1$ and within 10% of the range of $X2$ closest to that test observation. For instance, in order to predict the response for a test observation with $X1 = 0.6$ and $X2 = 0.35$, we will use observations in the range [0.55, 0.65] for $X1$ and in the range [0.3, 0.4] for $X2$. On average, what fraction of the available observations will we use to make the prediction?

Suppose (X_1, X_2) is uniformly distributed on $[0,1] \times [0,1]$. Note that X_1 and X_2 are independent since

$$\begin{aligned} F_{X_1,X_2}(x_1, x_2) &= \mathbb{P}(X_1 \le x_1, X_2 \le x_2) = x_1 x_2 \\ &= \mathbb{P}(X_1 \le x_1)\mathbb{P}(X_2 \le x_2) = F_{X_1}(x_1) F_{X_2}(x_2) \end{aligned}$$

for $0 \le x_1, x_2 \le 1$. Then we have

$$\begin{aligned} \mathbb{E}[P|(X_1, X_2) = (x_1, x_2)] &= \int_0^1 \int_0^1 P|(x_1, x_2)\, \mathbb{P}(X_1 = x_1, X_2 = x_2)\, dy\, dx \\ &= \int_0^1 \int_0^1 P|x_1\, P|x_2\, \mathbb{P}(X_1 = x_1)\, \mathbb{P}(X_2 = x_2)\, dy\, dx \\ &= \mathbb{E}(P|X_1 = x_1)\, \mathbb{E}(P|X_2 = x_2) \\ &= (0.0975)^2 \\ &= 0.0095 \end{aligned}$$

That is to make the prediction, we use only 0.95% of the available observations on average.

(c) (2pts) Now suppose that we have a set of observations on $p = 100$ features. Again the observations are uniformly distributed on each feature, and again each feature ranges in value from 0 to 1. We wish to predict a test observation's response using observations within the 10% of each feature's range that is closest to that test observation. What fraction of the available observations will we use to make the prediction?

Following from our solution to part (b), it follows that $X_1, \dots, X_{100}$ are independent of each other and we compute

$$\begin{aligned} \mathbb{E}[P|(X_1, \dots, X_{100}) = (x_1, \dots, x_{100})] &= \mathbb{E}(P|X_1 = x_1) \times \cdots \times \mathbb{E}(P|X_{100} = x_{100}) \\ &= (0.0975)^{100} \\ &= 7.95 \times 10^{-102} \end{aligned}$$

That is, on average, we are using practically none of the available observations to make the prediction.

(d) (3pts) Using your answers to parts (a)–(c), argue that a drawback of KNN when p is large is that there are very few training observations "near" any given test observation.

For sufficiently large p, we have shown in parts (a)–(c) that the fraction of observations near (that is, within 10% of each feature's range) a test observation goes to zero.

(e) (3pts) Now suppose that we wish to make a prediction for a test observation by creating a p-dimensional hypercube centered around the test observation that contains, on average, 10% of the training observations. For p =1, 2, and 100, what is the length of each side of the hypercube? Comment on your answer.

For $p = 1$, we create a line segment of length $l = 0.1$ centered around the test observation. For $p = 2$, the hypercube is a square centered around the test observation with sides of length $l = (0.1)^{1/2}$ since the area of this square would contain 10% of the training observations. Lastly, for $p = 100$, we have a 100-dimensional hypercube centered around the test observation with sides of length $l = (0.1)^{1/100}$ since we need the hypervolume to contain 10% of the training observations.

4. (6 pts) Suppose you trained a classifier for a spam detection system. The prediction result on the test set is summarized in the following table.

		Predicted class	
		Spam	not Spam
Actual class	Spam	8	2
	not Spam	16	974

Calculate

(a) (2 pts) Accuracy $\text{Accuracy} = \frac{\text{TP + TN}}{\text{TP + TN + FP + FN}} = \frac{8+974}{8+974+16+2} = 0.982$

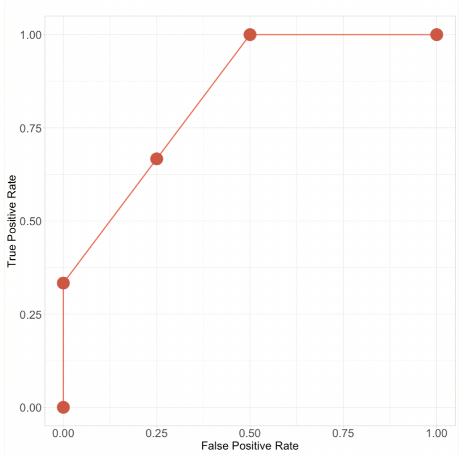
(b) (2 pts) Precision $\text{Precision} = \frac{\text{TP}}{\text{TP + FP}} = \frac{8}{8+16} = 0.333$

(c) (2 pts) Recall $\text{Recall} = \frac{\text{TP}}{\text{TP + FN}} = \frac{8}{8+2} = 0.800$

5. (9pts) Again, suppose you trained a classifier for a spam filter. The prediction result on the test set is summarized in the following table. Here, "+" represents spam, and "-" means not spam.

Confidence positive	Correct class
0.95	+
0.85	+
0.8	-
0.7	+
0.55	+
0.45	-
0.4	+
0.3	+
0.2	-
0.1	-

(a) (6pts) Draw a ROC curve based on the above table.



(b) (3pts) (Real-world open question) Suppose you want to choose a threshold parameter so that emails with confidence positives above the threshold can be classified as spam. Which value will you choose? Justify your answer based on the ROC curve.

We might choose a threshold of 0.5 because it corresponds to the point (0.25, 0.66) on the ROC curve which is closest to the ideal point (0, 1). If we never checked our spam folder and did not want to miss emails that were misclassified as spam, it might be better to use a threshold of 0.8.

6. (8 pts) In this problem, we will walk through a single step of the gradient descent algorithm for logistic regression. As a reminder,

$$f(x;\theta) = \sigma(\theta^\top x)$$

$$\text{Cross entropy loss } L(\hat{y}, y) = -[y \log \hat{y} + (1-y)\log(1-\hat{y})]$$

$$\text{The single update step } \theta^{t+1} = \theta^t - \eta \nabla_\theta L(f(x;\theta), y)$$

(a) (4 pts) Compute the first gradient $\nabla_\theta L(f(x;\theta), y)$.
Note that $\sigma'(z) = \sigma(z)(1-\sigma(z))$ implies that

$$\begin{aligned}\nabla_\theta[f(x;\theta)] &= \sigma(\theta^\top x)\,[1-\sigma(\theta^\top x)]\,\nabla_\theta[\theta^\top x]\\ &= \sigma(\theta^\top x)\,[1-\sigma(\theta^\top x)]\,x\end{aligned}$$

Then

$$\begin{aligned}\nabla_\theta L(f(x;\theta), y) &= -y\,\frac{1}{f(x;\theta)}\,\nabla_\theta[f(x;\theta)] - (1-y)\,\frac{1}{1-f(x;\theta)}\,\nabla_\theta[1-f(x;\theta)]\\ &= -y\,\frac{\sigma(\theta^\top x)\,[1-\sigma(\theta^\top x)]\,x}{\sigma(\theta^\top x)} + (1-y)\,\frac{\sigma(\theta^\top x)\,[1-\sigma(\theta^\top x)]\,x}{1-\sigma(\theta^\top x)}\\ &= \big(-y\,[1-\sigma(\theta^\top x)] + (1-y)\,\sigma(\theta^\top x)\big)\,x\\ &= \big(\sigma(\theta^\top x) - y\big)\,x.\end{aligned}$$

(b) (4 pts) Now assume a two dimensional input. After including a bias parameter for the first dimension, we will have $\theta \in \mathbb{R}^3$.

$$\text{Initial parameters : } \theta^0 = [0,0,0]$$

$$\text{Learning rate } \eta = 0.1$$

$$\text{data example : } x = [1,3,2], y = 1$$

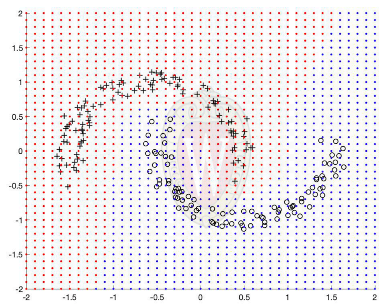
Compute the updated parameter vector θ^1 from the single update step.
From part (a), we have

$$\begin{aligned}\theta^1 &= \theta^0 - \eta\nabla_\theta L(f(x;\theta^0), y)\\ &= \theta^0 - \eta\,\big(\sigma(\theta^{0\top} x) - y\big)\,x\\ &= \begin{bmatrix}0\\0\\0\end{bmatrix} - 0.1\left(\sigma\left(\begin{bmatrix}0\\0\\0\end{bmatrix}^\top \begin{bmatrix}1\\3\\2\end{bmatrix}\right) - 1\right)\begin{bmatrix}1\\3\\2\end{bmatrix}\\ &= \begin{bmatrix}0\\0\\0\end{bmatrix} - 0.1\,(\sigma(0)-1)\begin{bmatrix}1\\3\\2\end{bmatrix}\\ &= \begin{bmatrix}0\\0\\0\end{bmatrix} - 0.1\left(\frac{1}{2}-1\right)\begin{bmatrix}1\\3\\2\end{bmatrix}\\ &= \begin{bmatrix}0\\0\\0\end{bmatrix} + 0.05\begin{bmatrix}1\\3\\2\end{bmatrix}\\ &= \begin{bmatrix}0.05\\0.15\\0.10\end{bmatrix}\end{aligned}$$

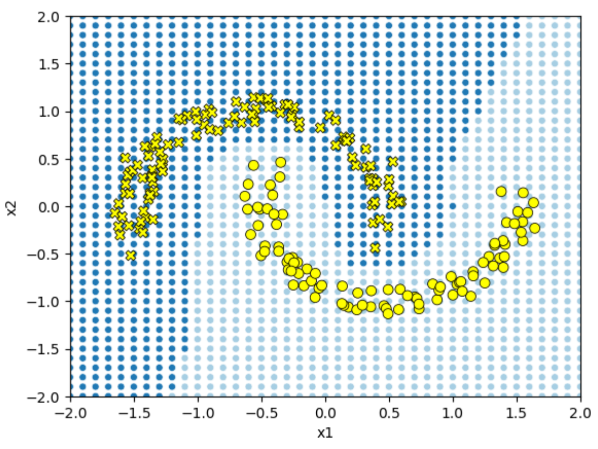
2 Programming (50 pts)

1. (10 pts) Use the whole D2z.txt as training set. Use Euclidean distance (i.e. $A = I$). Visualize the predictions of 1NN on a 2D grid $[-2 : 0.1 : 2]^2$. That is, you should produce test points whose first feature goes over $-2, -1.9, -1.8, \ldots, 1.9, 2$, so does the second feature independent of the first feature. You should overlay the training set in the plot, just make sure we can tell which points are training, which are grid.

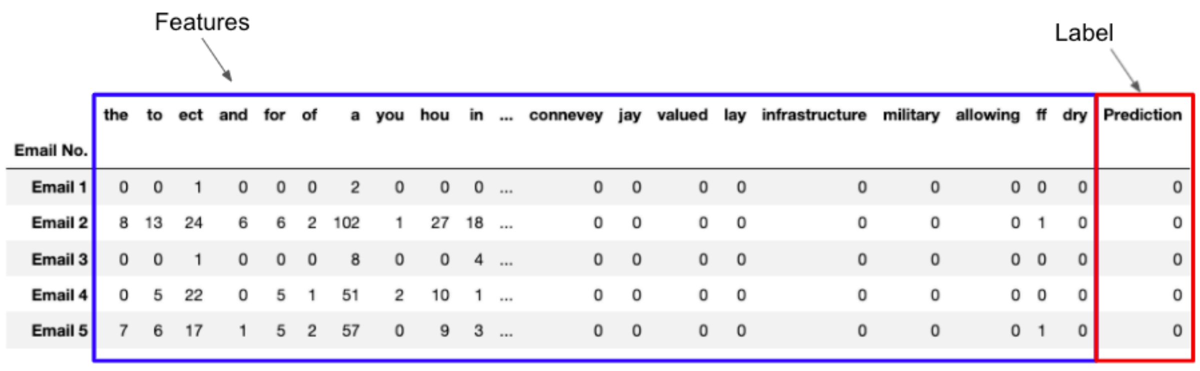
   The expected figure looks like this.



   Plot is given below. See code on GitHub.



Spam filter Now, we will use 'emails.csv' as our dataset. The description is as follows.


Features
Label

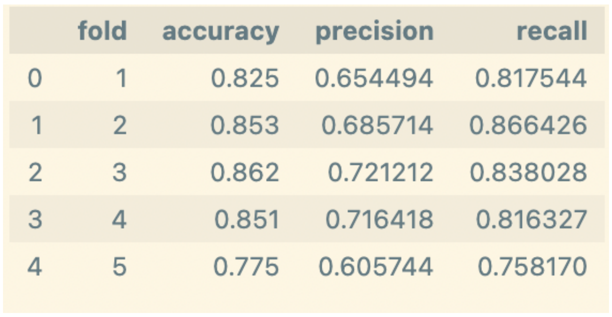
Email No.	the	to	ect	and	for	of	a	you	hou	in	...	connevey	jay	valued	lay	infrastructure	military	allowing	ff	dry	Prediction
Email 1	0	0	1	0	0	0	2	0	0	0	...	0	0	0	0	0	0	0	0	0	0
Email 2	8	13	24	6	6	2	102	1	27	18	...	0	0	0	0	0	0	0	1	0	0
Email 3	0	0	1	0	0	0	8	0	0	4	...	0	0	0	0	0	0	0	0	0	0
Email 4	0	5	22	0	5	1	51	2	10	1	...	0	0	0	0	0	0	0	0	0	0
Email 5	7	6	17	1	5	2	57	0	9	3	...	0	0	0	0	0	0	0	1	0	0

- Task: spam detection
- The number of rows: 5000
- The number of features: 3000 (Word frequency in each email)

- The label (y) column name: 'Predictor'
- For a single training/test set split, use Email 1-4000 as the training set, Email 4001-5000 as the test set.
- For 5-fold cross validation, split dataset in the following way.
  - Fold 1, test set: Email 1-1000, training set: the rest (Email 1001-5000)
  - Fold 2, test set: Email 1000-2000, training set: the rest
  - Fold 3, test set: Email 2000-3000, training set: the rest
  - Fold 4, test set: Email 3000-4000, training set: the rest
  - Fold 5, test set: Email 4000-5000, training set: the rest

2. (8 pts) Implement 1NN, Run 5-fold cross validation. Report accuracy, precision, and recall in each fold.

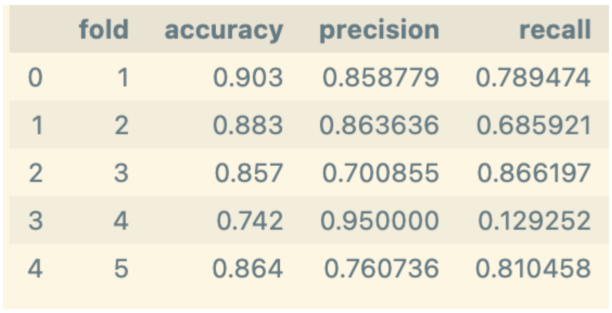
   See code for implementation on GitHub.



	fold	accuracy	precision	recall
0	1	0.825	0.654494	0.817544
1	2	0.853	0.685714	0.866426
2	3	0.862	0.721212	0.838028
3	4	0.851	0.716418	0.816327
4	5	0.775	0.605744	0.758170

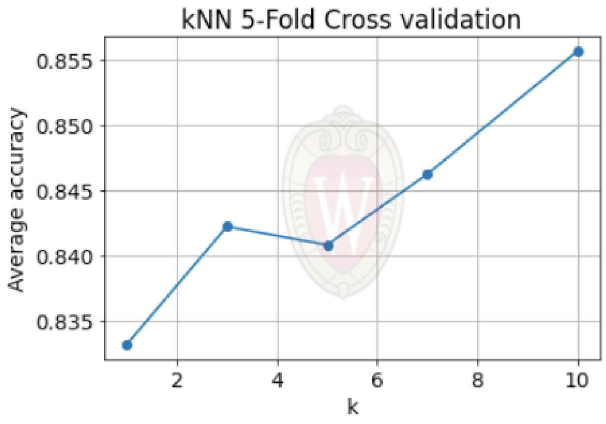
3. (12 pts) Implement logistic regression (from scratch). Use gradient descent (refer to question 6 from part 1) to find the optimal parameters. You may need to tune your learning rate to find a good optimum. Run 5-fold cross validation. Report accuracy, precision, and recall in each fold.

   See code for implementation on GitHub.

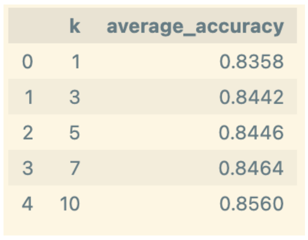


	fold	accuracy	precision	recall
0	1	0.903	0.858779	0.789474
1	2	0.883	0.863636	0.685921
2	3	0.857	0.700855	0.866197
3	4	0.742	0.950000	0.129252
4	5	0.864	0.760736	0.810458

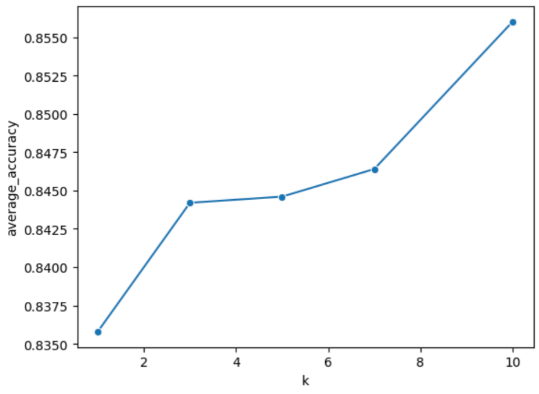
4. (10 pts) Run 5-fold cross validation with kNN varying k (k=1, 3, 5, 7, 10). Plot the average accuracy versus k, and list the average accuracy of each case.
Expected figure looks like this.



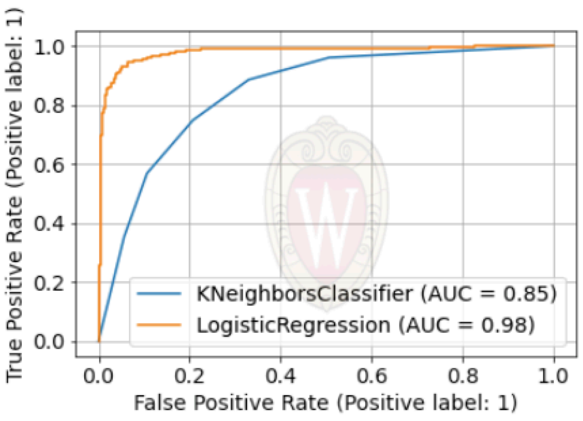
Plot and table given below. See code on GitHub.



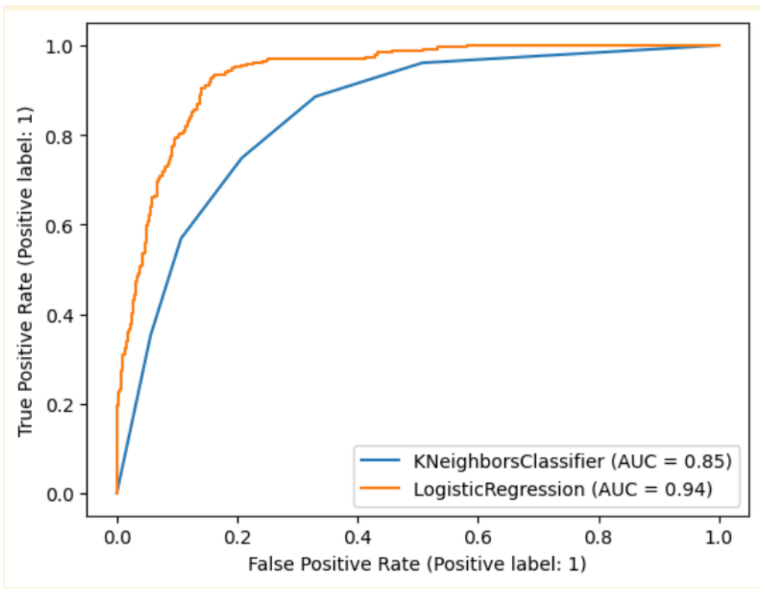
	k	average_accuracy
0	1	0.8358
1	3	0.8442
2	5	0.8446
3	7	0.8464
4	10	0.8560



5. (10 pts) Use a single training/test setting. Train kNN (k=5) and logistic regression on the training set, and draw ROC curves based on the test set.
Expected figure looks like this. Note that the logistic regression results may differ.



Plot given on next page.


True Positive Rate (Positive label: 1)
False Positive Rate (Positive label: 1)
0.0
0.2
0.4
0.6
0.8
1.0
KNeighborsClassifier (AUC = 0.85)
LogisticRegression (AUC = 0.94)