ISYE 6740, Fall 2025, Prof. Yao Xie
Homework 4

100 points

Student Name Here

Provided Data and Submission Requirements:

- All results, explanations, and math must be included in your PDF submission. Hand-written work is not accepted per the syllabus. Screenshots of work is also not accepted.
- If applicable, questions marked with **GS** must be submitted to Gradescope. Failure to pass gradescope tests will result in penalties to the points for that question.

This assignment does not have any gradescope requirements
Assignment Data:

- Q4: Comparing Classifiers: Divorce Classification / Prediction [marriage.csv]

1. Optimization (25 points).

Consider a simplified logistic regression problem. Given m training samples (x^i, y^i), $i = 1, \ldots, m$. The data $x^i \in \mathbb{R}$, and $y^i \in \{0, 1\}$. To fit a logistic regression model for classification, we solve the following optimization problem, where $\theta \in \mathbb{R}$ is a parameter we aim to find:

$$\max_\theta \ell(\theta), \tag{1}$$

where the log-likelhood function

$$\ell(\theta) = \sum_{i=1}^m \left\{ -\log(1 + \exp\{-\theta^T x^i\}) + (y^i - 1)\theta^T x^i \right\}.$$

1. (5 points) Show step-by-step mathematical derivation for the gradient of the cost function $\ell(\theta)$ in (1).

   1.1 Answer:

$$\nabla_\theta \ell(\theta) = \sum_{i=1}^m \nabla_\theta \ell_i(\theta)$$

Take the gradient of individual term. Answer given in the lecture slides

$$\frac{\partial \ell^i}{\partial \theta} = \frac{e^{-\theta^T x^i} x^i}{1 + e^{-\theta^T x^i}} + (y^i - 1)x^i$$

We can see from the lecture slides that the term $\frac{e^{-\theta^T x^i} x^i}{1+e^{-\theta^T x^i}}$ is the term for $p(y=0|x,\theta)$, so we can also take this as $(1-p(y=1|x,\theta))$. In the notes, $p(y=1|x,\theta)=\sigma(\theta^T x)$. So we can substitute the sigmoid function into the gradient:

$$\frac{\partial \ell^i}{\partial \theta} = (1-\sigma(\theta^T x^i))x^i + (y^i - 1)x^i$$
$$\frac{\partial \ell^i}{\partial \theta} = (y^i - \sigma(\theta^T x^i))x^i$$

So, the gradient is this equation applied to all i:

$$\nabla_\theta \ell(\theta) = \sum_{i=1}^{m} (y^i - \sigma(\theta^T x^i))x^i$$

2. (5 points) Write a pseudo-code for performing **gradient descent** to find the optimizer θ^*. This is essentially what the training procedure does.

   1.2 Answer:

```
Initialize Theta and set max iterations
Loop until max iterations or convergence:
    old_gradient = gradient # save previously calculated gradient

    # calculate gradient of all samples:
    gradient = sum(x[i](y[i]
        - sigmoid(theta.T * x[i])) for i in range(m))

    # update theta:
    theta = theta + learning_rate * gradient

    # check convergence:
    if gradient - old_gradient < tolerance:
        break
return theta
```

3. (5 points) Write the pseudo-code for performing the **stochastic gradient descent** algorithm to solve the training of logistic regression problem (1). Please explain the difference between gradient descent and stochastic gradient descent for training logistic regression.

1.3 Answer: This is largely the same as the previous answer, but it randomly selects individual samples to perform the gradient calculation and theta update instead of calculating on all samples. This makes it more efficient for large datasets, but also more noisy.

```
initialize theta and set max iterations
loop until max iterations or convergence:
    old_gradient = gradient # save previously calculated gradient

    subset_x, subset_y = random_sample(data, step_size)

    # calculate gradient of subset samples:
    gradient = sum(subset_x[i](subset_y[i]
            - sigmoid(theta.T * subset_x[i])) for i in range(step_size))

    # update theta:
    theta = theta + learning_rate * gradient

    # check convergence by seeing if likelihood is close to zero:
    if ||gradient|| < tolerance:
        break
return theta
```

4. (10 points) We will **prove that the training problem in basic logistic regression problem is concave.** Derive the Hessian matrix of $\ell(\theta)$ and based on this, show the training problem (1) is concave. Explain why the problem can be solved efficiently and gradient descent will achieve a unique global optimizer, as we discussed in class.

1.4 Answer: The Hessian is the second derivative of the function. We can arrive there by taking the derivative of the gradient w.r.t θ again.

$$\nabla^2_\theta \ell(\theta) = \sum_{i=1}^{m} (\frac{\partial}{\partial \theta}(y^i - \sigma(\theta^T x^i))x^i)$$

Derivative of sigmoid [1]:

$$\frac{\partial}{\partial x}\sigma(x) = \sigma(x)(1 - \sigma(x))$$

And derive the internal sigmoid term for chain rule:

$$\frac{\partial \theta^T x^i}{\partial}\theta = x^i$$

Apply sigmoid derivation and chain rule:

$$\nabla^2_\theta \ell(\theta) = \sum_{i=1}^{m}(-\sigma(\theta^T x^i)(1-\sigma(\theta^T x^i))x^i(x^i)^T)$$

To prove concave, we can multiply the Hessian by a vector and analyze if the result is negative: $z^T H z \leq 0$ [1].

$$\begin{aligned} z^T H z &= z^T(-\sum_{i=1}^{m}(\sigma(\theta^T x^i)(1-\sigma(\theta^T x^i))x^i(x^i)^T))z \\ &= -\sum_{i=1}^{m}(\sigma(\theta^T x^i)(1-\sigma(\theta^T x^i)))(z^T x^i(x^i)^T z) \\ &= -\sum_{i=1}^{m}(\sigma(\theta^T x^i)(1-\sigma(\theta^T x^i)))(z^T x^i)^2 \end{aligned}$$

The following terms in the Hessian cannot be negative:

$\sigma(\theta^T x^i)$	This is a probability, so between [0,1]
$1-\sigma(\theta^T x^i)$	Also [0,1] due to above
$(z^T x^i)^2$	Squared term must be ≥ 0

Since all the terms are non-negative, the negative sign in front of the summation means $z^T H z \leq 0$. This means the Hessian is negative semi-definite, which means the function is concave.

2. *Fair* Gaussian mixture model (25 points).

This question builds on EM for GMM model estimation, as well as optimization lectures; the purpose is to practice constrained optimization. Recall the EM step in the estimation of Gaussian mixture models. Now suppose that in addition to all the original setup, we require a *fairness* constraint: the weights of the Gaussian components cannot be smaller than a pre-specified constant c. We will derive a modified EM algorithm for the Gaussian model, with two components $K = 2$, when the weights for each component are required to be greater than c: $\pi_1 \geq c$, and $\pi_2 \geq c$ (with $c < 0.5$).

Show the modified EM algorithm, as well as the complete mathematical derivations.

Hint: The only thing you will need to change is that in the M-step, we have to solve a constrained optimization problem with the original constraint $\pi_1 + \pi_2 = 1$, as well as $\pi_1 > c$ and $\pi_2 > c$. You will need to introduce additional Lagrangian multipliers for the new constraints and discuss the *cases when the constraints are active (or not active) using the KKT condition (in particular, the complementarity condition).*

Answer:

Expectation: The expectation step is not modified by the extra constraints.

$$\tau_k^i = \frac{\pi_k \mathcal{N}(x^i|\mu_k, \Sigma_k)}{\sum_{k'=1...K} \pi_{k'} \mathcal{N}(x^i|\mu_k, \Sigma_k)}$$

Maximization: The maximization step incorporates the constraint terms. Without the extra terms, the objective function is:

$$f(\theta) = \sum_{i=1}^{m} \sum_{k=1}^{K} \tau_k^i (\log(\pi_k) - \frac{1}{2}(x^i - \mu_k)^T \Sigma_k^{-1} (x^i - \mu_k) - \frac{1}{2}\log(|\Sigma_k|) - \frac{n}{2}\log(2\pi))$$

With the initial constraint Lagrangian term:

$$f(\theta) = \sum_{i=1}^{m} \sum_{k=1}^{K} \tau_k^i (\log(\pi_k) - \frac{1}{2}(x^i - \mu_k)^T \Sigma_k^{-1} (x^i - \mu_k) - \frac{1}{2}\log(|\Sigma_k|) - \frac{n}{2}\log(2\pi)) + \lambda(1 - \sum_{i=1}^{K} \pi_k)$$

So, we need to add additional Lagrangian terms to enforce the constraint. I will make these λ_1 and λ_2 while designating the initial constraint λ_0. Additionally, I will expand the π summations since $K = 2$.

$$f(\theta) = \sum_{i=1}^{m} \sum_{k=1}^{K} \tau_k^i (\log(\pi_k) - \frac{1}{2}(x^i - \mu_k)^T \Sigma_k^{-1} (x^i - \mu_k) - \frac{1}{2}\log(|\Sigma_k|) - \frac{n}{2}\log(2\pi))$$
$$+ \lambda_0(1 - \pi_1 - \pi_2) + \lambda_1(c - \pi_1) + \lambda_2(c - \pi_2)$$

As far as the maximization goes, we perform the usual steps: take the partial derivatives and set equal to zero. We can see that the extra constraints didn't impact the μ_k or Σ_k terms, so their values are unchanged from what we already know:

$$\mu_k = \frac{\sum_{i=1}^{k} \tau_k^i x^i}{\sum_{i=1}^{k} \tau_k^i} \quad \Sigma_k = \frac{\sum_{i=1}^{k} \tau_k^i (x^i - \mu_k)(x^i - \mu_k)^T}{\sum_{i=1}^{k} \tau_k^i}$$

The constraints impact the maximization of π_k. As seen in the normal EM derivation, the terms unrelated to π_k drop out. We will take derivatives with respect to π_1 and π_2.

$$\begin{aligned}
0 = \frac{\partial L}{\partial \pi_1} &= \frac{\partial}{\partial \pi_1}\lambda_0(1-\pi_1-\pi_2) + \lambda_1(c-\pi_1) + \lambda_2(c-\pi_2) + \sum_{i=1}^{m}(\tau_1^i \log \pi_1) + (\tau_2^i \log \pi_2) \\
&= -\lambda_0 - \lambda_1 + \frac{\sum_{i=1}^m \tau_1^i}{\pi_1} \\
\lambda_0 + \lambda_1 &= \frac{\sum_{i=1}^m \tau_1^i}{\pi_1} \\
0 = \frac{\partial L}{\partial \pi_2} &= \frac{\partial}{\partial \pi_2}\lambda_0(1-\pi_1-\pi_2) + \lambda_1(c-\pi_1) + \lambda_2(c-\pi_2) + \sum_{i=1}^{m}(\tau_1^i \log \pi_1) + (\tau_2^i \log \pi_2) \\
&= -\lambda_0 - \lambda_2 + \frac{\sum_{i=1}^m \tau_2^i}{\pi_2} \\
\lambda_0 + \lambda_2 &= \frac{\sum_{i=1}^m \tau_2^i}{\pi_2}
\end{aligned}$$

These equations are also the stationary conditions of KKT. We need to show Primal feasibility, dual feasibility, and complementary slackness [3]. For a solution to be optimal, it must satisfy all 4 condition categories.

Primal feasibility:

$$\begin{aligned}
\pi_1 + \pi_2 &= 1 \\
\pi_1 &\geq c \\
\pi_2 &\geq c
\end{aligned}$$

Dual feasibility:

$$\begin{aligned}
\lambda_1 &\geq 0 \\
\lambda_2 &\geq 0
\end{aligned}$$

Complementary slackness:

$$\begin{aligned}
\lambda_1(c-\pi_1) &= 0 \\
\lambda_2(c-\pi_2) &= 0
\end{aligned}$$

Since there are two inequality constraints, there are four cases to consider.

Neither fairness constraint is active: If neither constraint is active, then $\lambda_1, lambda_2 = 0$, meaning they don't affect the equations. In this case, we can update the π_k equations to:

$$\begin{aligned}
\pi_1 &= \frac{\sum_{i=1}^m \tau_1^i}{\lambda_0} \\
\pi_2 &= \frac{\sum_{i=1}^m \tau_2^i}{\lambda_0}
\end{aligned}$$

Substitute this into the original constraint:

$$1 = \frac{\sum_{i=1}^m \tau_1^i}{\lambda_0} + \frac{\sum_{i=1}^m \tau_2^i}{\lambda_0}$$
$$\lambda_0 = \sum_{i=1}^m \tau_1^i + \sum_{i=1}^m \tau_2^i = m$$

Substituting back into the π_k equations:

$$\pi_1 = \frac{\sum_{i=1}^m \tau_1^i}{m}$$
$$\pi_2 = \frac{\sum_{i=1}^m \tau_2^i}{m}$$

These π equations are the same as the normal EM algorithm, which is to be expected since the extra fairness constraints are not active. These are valid if the equations are both $\geq c$, but the next cases explore what happens if that is not true.

Either constraint is active: If $\frac{\sum_{i=1}^m \tau_1^i}{m} \geq c$ does not hold or $\frac{\sum_{i=1}^m \tau_2^i}{m} \geq c$ does not hold (but not both at the same time), this means that the constraint for one of the *pi* terms is active. In the case of the active constraint:

$$\lambda(c - \pi) = 0 \implies \pi = c$$

And given $\pi_1 + \pi_2 = 1$, the other *pi* term is $1 - c$.

Both constraints are active:

This cannot be true. From the same logic in the case above, this would require $\pi_1, pi_2 = c$. This cannot be true because that would require $c = 0.5$ to satisfy the original constraint $pi_1 + pi_2 = 1$, which is specifically disallowed in the problem formulation.

How this impacts the algorithm: EM acts as usual until calculating the π_k terms. At that point, if the $\pi_k \geq c$, then the normal maximization continues. If that is not the case, then the constraint $\pi_{active} = c$ and $\pi_{inactive} = 1 - c$.

3. Bayes Classifier for spam filtering (30 points)

In this problem, we will use the Bayes Classifier algorithm to fit a spam filter by hand. This will enhance your understanding to the Bayes classifier and build intuition. This question does not involve any programming but only derivation and hand calculation. Tools can be used (Python, Excel, Etc.) but all calculations and derivations must still be provided in your report.

Spam filters are used in all email services to classify received emails as "Spam" or "Not Spam". A simple approach involves maintaining a vocabulary of words that commonly occur in "Spam" emails and classifying an email as "Spam" if the number of words from the dictionary that are present in the email is over a certain threshold. We are given the vocabulary consists of 15 words

$V = \{$free, cheese, pizza, get, fast, now, order, available, slice, offer, not, good, pepperoni, tastes, crust$\}$.

We will use V_i to represent the ith word in V. Use the training set provided in the table below.

Spam	**Non Spam**
free cheese pizza get fast now	good pepperoni cheese pizza now available
order available pizza slice fast	pepperoni pizza tastes good
pizza offer not good	pizza crust tastes not good
	cheese slice now pizza

Table 1: Spam and Non-Spam Messages

Recall that the Naive Bayes classifier assumes the probability of an input depends on its input feature. The feature for each sample is defined as $x^{(i)} = [x_1^{(i)}, x_2^{(i)}, \dots, x_d^{(i)}]^T$, $i = 1, \dots, m$ and the class of the ith sample is $y^{(i)}$. In our case the length of the input vector is $d = 15$, which is equal to the number of words in the vocabulary V. Each entry $x_j^{(i)}$ is equal to the number of times word V_j occurs in the i-th message.

1. (5 points) Calculate class prior $\mathbb{P}(y = 0)$ and $\mathbb{P}(y = 1)$ from the training data, where $y = 0$ corresponds to spam messages, and $y = 1$ corresponds to non-spam messages. Note that these class prior essentially corresponds to the frequency of each class in the training sample. Write down the feature vectors for each spam and non-spam messages.

   3.1 Answer: As mentioned, the priors are just the proportion of the class in the training data:

$$P(y = 0) = \frac{3}{7} \quad P(y = 1) = \frac{4}{7}$$

   The feature vectors are a mapping of prevalence from the given message to the word table.

Spam	**Non Spam**
111111000000000	011001010001100
001010111000000	001000000001110
001000000111000	001000000011011
	011001001000000

2. (15 points) Assuming the keywords follow a multinomial distribution, the likelihood of a sentence with its feature vector x given a class c is given by

$$\mathbb{P}(x|y=c) = \frac{n!}{x_1!\cdots x_d!}\prod_{k=1}^{d}\theta_{c,k}^{x_k}, \quad c=\{0,1\}$$

where $n = x_1 + \cdots x_d$, $0 \le \theta_{c,k} \le 1$ is the probability of word k appearing in class c, which satisfies

$$\sum_{k=1}^{d}\theta_{c,k} = 1, \quad c = \{0,1\}.$$

Given this, the complete log-likelihood function for our training data is given by

$$\ell(\theta_{0,1},\ldots,\theta_{0,d},\theta_{1,1},\ldots,\theta_{1,d}) = \sum_{i=1}^{m}\sum_{k=1}^{d} x_k^{(i)}\log\theta_{y^{(i)},k}$$

(In this example, $m = 7$.) Calculate the maximum likelihood estimates of $\theta_{0,1}$, $\theta_{0,5}$, $\theta_{1,2}$, $\theta_{1,15}$ by maximizing the log-likelihood function above.
(Hint: We are solving a constrained maximization problem and you will need to introduce Lagrangian multipliers and consider the Lagrangian function.)

3.2 Answer: The constraint $\sum_{k=1}^{d}\theta_{c,k} = 1, \quad c = \{0,1\}$ is actually two constraints. Using the bag of words example, this constraint says that "when pulling words from the spam bag, the sum of the probabilities of pulling each word in this class is 1[2]." More simply, within each bag, all the word probabilities in that bag must sum to 1. The two constraints look like:

$$\sum_{k=1}^{d}\theta_{0,k} = 1, \sum_{k=1}^{d}\theta_{1,k} = 1$$

To maximize this function, we use the Lagrangian to incorporate the constraints directly into the function.

$$\ell(\theta_{c,k}) = \sum_{i=1}^{m}\sum_{k=1}^{d} x_k^{(i)}\log\theta_{y^{(i)},k} - \lambda_0(1 - \sum_{k=1}^{d}\theta_{y^i=0,k}) - \lambda_1(1 - \sum_{k=1}^{d}\theta_{y^i=1,k})$$

The Lagrangian multipliers λ_c acts as punishment terms for exceeding the constraints. Since $1-\sum_{k=1}^{d}\theta_{c,k}$ is distributed to lambda, exceeding the constraints $\sum_{k=1}^{d}\theta_{c,k}$ results in a negative term in the function. If the parameters exceed the constraints, the Lagrangian multiplier reduces the objective function, and those parameters are unlikely to be the maximized parameters.

From the theory of the bag of words model, we are going to calculate the likelihood of pulling each string of words from the spam bag and the non-spam bag. That's what the likelihood function is calculating. It is easier to understand the function by breaking the generalized form into each class $c = y^i$ to match the constraints. We can split the $\sum_{i=1}^{m}$ term based on which vectors belong to each class.

$$\ell(\theta_{0,k}) = \sum_{i \in c=0} \sum_{k=1}^{d} x_k^{(i)} \log \theta_{0,k} - \lambda_0 (1 - \sum_{k=1}^{d} \theta_{y^i=0,k})$$
$$\ell(\theta_{1,k}) = \sum_{i \in c=1} \sum_{k=1}^{d} x_k^{(i)} \log \theta_{1,k} - \lambda_1 (1 - \sum_{k=1}^{d} \theta_{y^i=1,k})$$

Generalized:

$$\ell(\theta_{c,k}) = \sum_{i \in c} \sum_{k=1}^{d} x_k^{(i)} \log \theta_{c,k} - \lambda_c (1 - \sum_{k=1}^{d} \theta_{y^i=c,k})$$

This function looks the same as the given function, but it splits the summation over all vectors to summation of vectors within each class. This also allows us to split the lagrangian multipliers to each class, meaning we do not have to take both into account with the partial derivatives. We can take the partial derivative set to 0 to solve for any given $\theta_{c,k}$. Since we are only finding one parameter, we can remove the summation over k.

$$\frac{\partial \ell(\theta_{c,k})}{\partial \theta_{c,k}} = 0 = \frac{\partial}{\partial \theta_{c,k}} \sum_{i \in c} \log \theta_{c,k} - \lambda_c (1 - \theta_{c,k})$$
$$0 = \frac{\sum_{i \in c} x_k^{(i)}}{\theta_{c,k}} - \lambda_c$$
$$\theta_{c,k} = \frac{\sum_{i \in c} x_k^{(i)}}{\lambda_c}$$

$$\frac{\partial \ell(\theta_{c,k})}{\partial \lambda_c} = 0 = \frac{\partial}{\partial \lambda_c} \sum_{i \in c} \sum_{k=1}^{d} x_k^{(i)} \log \theta_{c,k} - \lambda_c (1 - \sum_{k=1}^{d} \theta_{y^i = c,k})$$
$$0 = \frac{\partial}{\partial \lambda_c} (-\lambda_c + \lambda_c \sum_{k=1}^{d} \theta_{c,k})$$
$$0 = (-1 + \sum_{k=1}^{d} \theta_{c,k})$$
$$1 = (\sum_{k=1}^{d} \theta_{c,k})$$

This answer is the same as the constraint. We can substitute our answer for $\theta_{c,k}$ into the constraint to solve for λ_c.

$$1 = (\sum_{k=1}^{d} \frac{\sum_{i \in c} x_k^{(i)}}{\lambda_c})$$
$$\lambda_c = \sum_{k=1}^{d} \sum_{i \in c} x_k^{(i)}$$

So, the optimal λ_c is the total number of words in class c. So, putting this into the earlier equation for $\theta_{c,k}$, we get:

$$\theta_{c,k} = \frac{\sum_{i \in c} x_k^{(i)}}{\sum_{k=1}^{d} \sum_{i \in c} x_k^{(i)}}$$

From this equation, we can see that the likelihood of pulling word k from class c is the number of times the word appears in class c divided by the number of words in class c, which makes logical sense. We can use this generalized equation to calculate the requested likelihoods.

$$\theta_{0,1} = \frac{1}{15}$$
$$\theta_{0,5} = \frac{2}{15}$$
$$\theta_{1,2} = \frac{2}{19}$$
$$\theta_{1,15} = \frac{1}{19}$$

3. (10 points) Given a test message "cheese pizza available", using the Naive Bayes classifier that you have trained in Part (a)-(b), to calculate the posterior and decide whether it is spam or not spam. Derivations must be shown in your report.

   3.3 Answer: The feature vector for this message is: [011000010000000]. We will use Bayes Theorem to calculate the posterior probability that the message belongs to spam and non-spam. We will compare these probabilities and choose the higher probability for classification. Bayes Theorem is:

$$P(C|x) = \frac{P(x|C)P(C)}{P(x)} = \frac{P(x|C)P(C)}{\sum_C P(x|C)P(C)}$$

   We already calculated the prior probabilities in 3.1. We will first calculate the per-word likelihoods for each class.

$$\theta_{0,2} = \frac{1}{15}$$
$$\theta_{0,3} = \frac{3}{15}$$
$$\theta_{0,8} = \frac{1}{15}$$
$$\theta_{1,2} = \frac{2}{19}$$
$$\theta_{1,3} = \frac{4}{19}$$
$$\theta_{1,8} = \frac{1}{19}$$

   We can use the given formula to calculate the likelihood of the message for each class.

$$\mathbb{P}(x|y=c) = \frac{n!}{x_1! \cdots x_d!} \prod_{k=1}^{d} \theta_{c,k}^{x_k}, \quad c = \{0,1\}$$

   However, we can ignore the $\frac{n!}{x_1! \cdots x_d!}$ term since it is the same for both classes. By that same logic, we can also ignore the $P(X)$ term in Bayes Theorem. So, in the end, we just need to calculate $P(x|C)P(C)$ for each class and compare them.

$$P(x|C=0) \propto \theta_{0,2} \times \theta_{0,3} \times \theta_{0,8} = \frac{1}{15} \times \frac{3}{15} \times \frac{1}{15}$$
$$P(x|C=1) \propto \theta_{1,2} \times \theta_{1,3} \times \theta_{1,8} = \frac{2}{19} \times \frac{4}{19} \times \frac{1}{19}$$

   Now calculate $P(x|C)P(C)$ and choose the class with the higher value.

$$P(C=0|x) \propto (\frac{1}{15} \times \frac{3}{15} \times \frac{1}{15}) \times \frac{3}{7} = .00038$$
$$P(C=1|x) \propto (\frac{2}{19} \times \frac{4}{19} \times \frac{1}{19}) \times \frac{4}{7} = .00067$$

Since $P(C = 1|x) > P(C = 0|x)$, we classify this message as non-spam.

4. Comparing classifiers: Divorce classification/prediction (20 points)

In lectures, we learn different classifiers. This question is compare them on two datasets. Python users, please feel free to use Scikit-learn, which is a commonly-used and powerful Python library with various machine learning tools. But you can also use other similar libraries in other languages of your choice to perform the tasks.

This dataset is about participants who completed the personal information form and a divorce predictors scale. The data is a modified version of the publicly available at https://archive.ics.uci.edu/dataset/539/divorce+predictors+data+set (by injecting noise so you will not get the exactly same results as on UCI website). The dataset **marriage.csv** is contained in the homework folder. There are 170 participants and 54 attributes (or predictor variables) that are all real-valued. The last column of the CSV file is label y (1 means "divorce", 0 means "no divorce"). Each column is for one feature (predictor variable), and each row is a sample (participant). A detailed explanation for each feature (predictor variable) can be found at the website link above. Our goal is to build a classifier using training data, such that given a test sample, we can classify (or essentially predict) whether its label is 0 ("no divorce") or 1 ("divorce").

We are going to compare the following classifiers (**Naive Bayes, Logistic Regression, and KNN**). Use the first 80% data for training and the remaining 20% for testing. If you use scikit-learn you can use train_test_split to split the dataset.

Remark: Please note that, here, for Naive Bayes, this means that we have to estimate the variance for each individual feature from training data. When estimating the variance, if the variance is zero to close to zero (meaning that there is very little variability in the feature), you can set the variance to be a small number, e.g., $\epsilon = 10^{-3}$. We do not want to have include zero or nearly variance in Naive Bayes. This tip holds for both Part One and Part Two of this question.

1. (10 points) Report testing accuracy for each of the three classifiers. Comment on their performance: which performs the best and make a guess why they perform the best in this setting.

   4.1 Answer: The accuracies I got were strangely uniform:

   - Naive Bayes accuracy: 0.9705882352941176
   - LR accuracy: 0.9705882352941176
   - KNN best accuracy (k=2): 0.9705882352941176

   I think the small dataset could have something to do with this uniformity. Judging from the questions, not only do they seem pretty indicative of whether a relationship will fail, but the individuals' state of mind while filling out the survey could be a factor. Most importantly, it could just be very easy to predict divorce from the feature questions (indicating a well-formulated survey, if you ask me). Further, my random state variable could have picked 97% easy-to-classify test samples, which would explain the uniform accuracy.

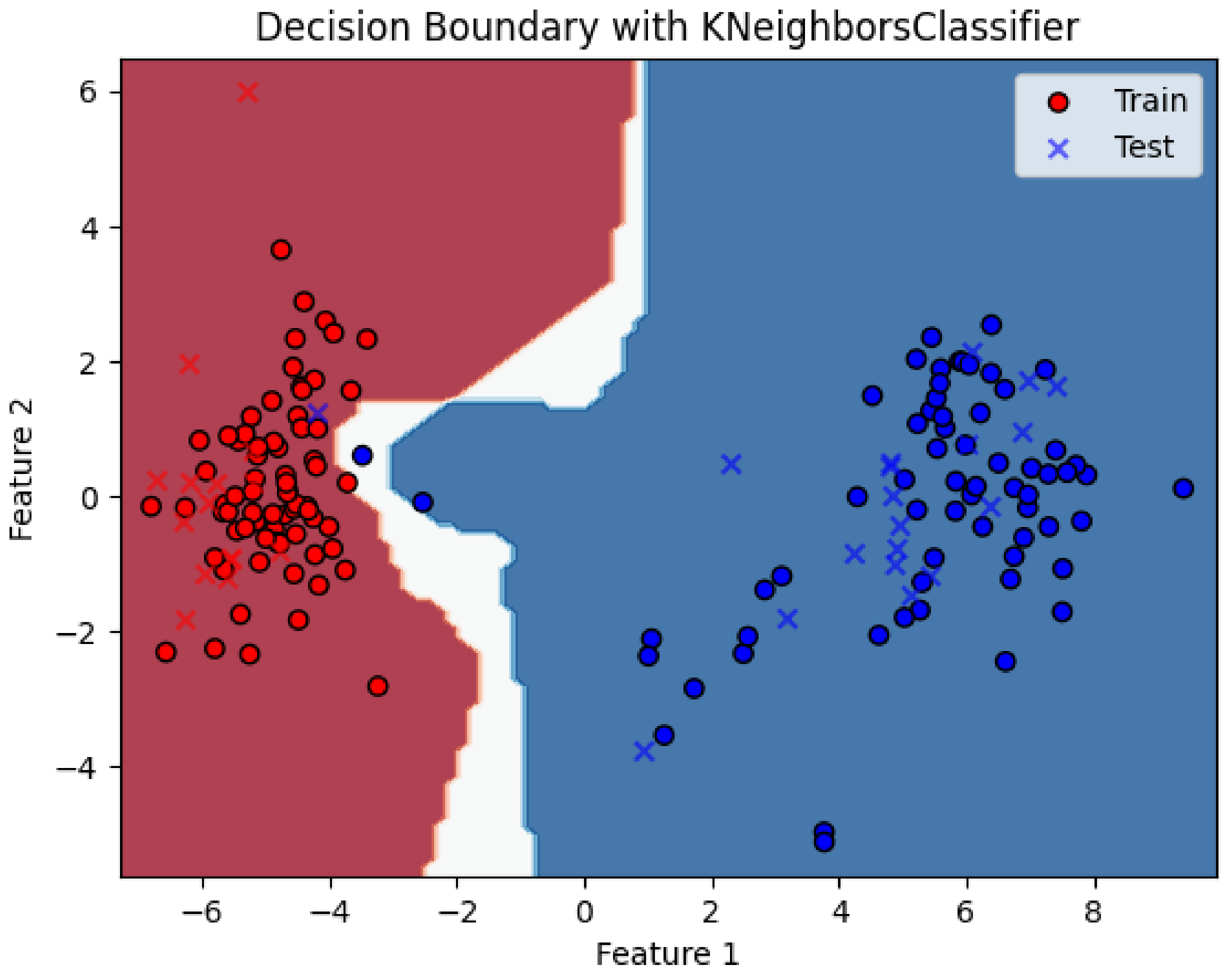
2. (10 points) Now perform PCA to project the data into two-dimensional space. Build the classifiers (**Naive Bayes, Logistic Regression, and KNN**) using the two-dimensional PCA results. Plot the data points and decision boundary of each classifier in the two-dimensional space. Comment on the difference between the decision boundary for the three classifiers. Please clearly represent the data points with different labels using different colors.

   4.2 Answer: The PCA graphs show that the data is largely separable. As expected from the 4.1 results, it seems algorithmically possible to separate the two classes into separable groups. There are three data points that are projected close to the wrong group. Otherwise, the data groupings are quite clear. In all three cases with the PCA, the testing accuracy was the same as before.

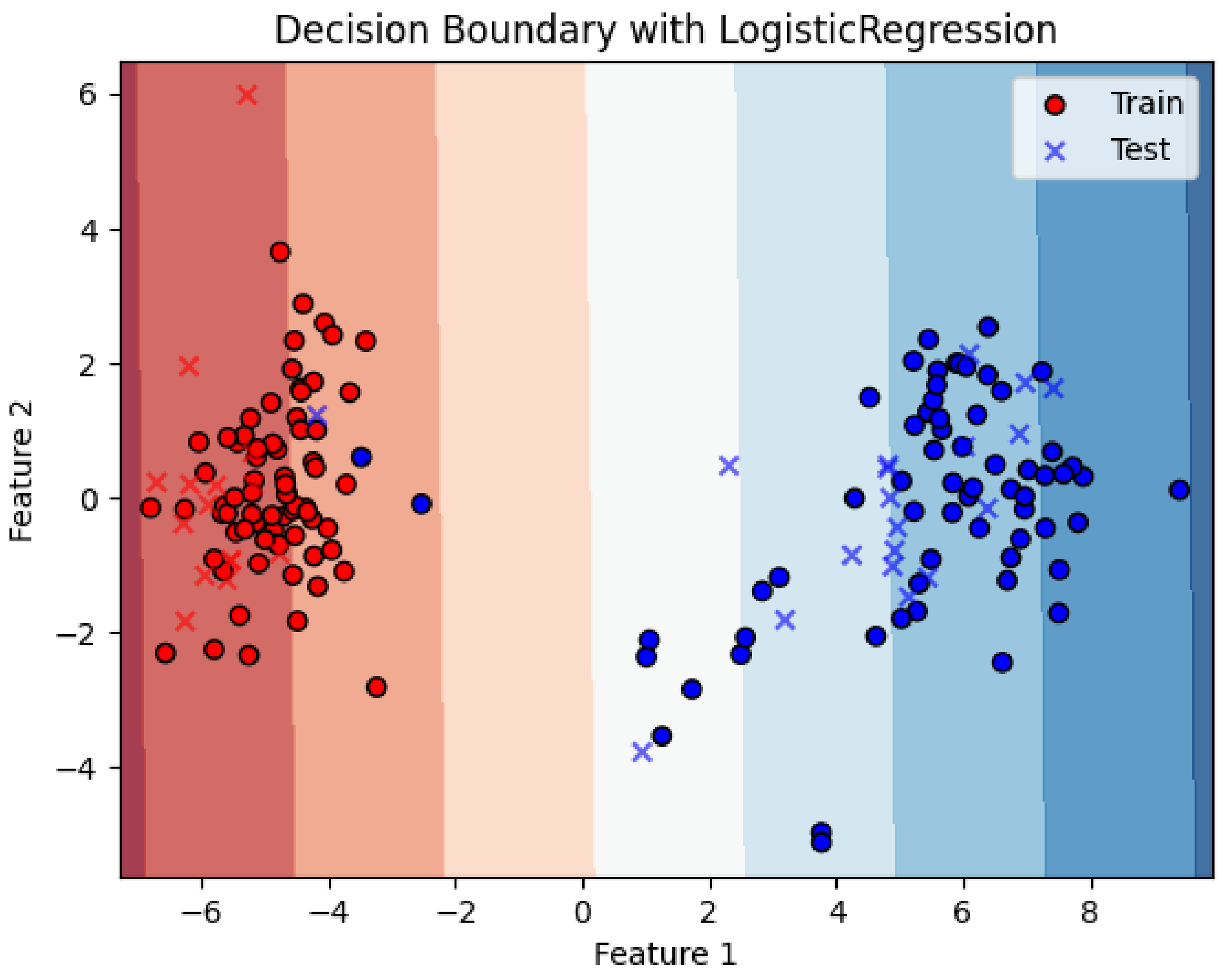
   While all of the classifiers misclassified one test point, the KNN classifier misclassified one less training point. Its nonlinear boundary could be seen as overfitting, but the fit also promisingly directs towards the misclassified test point.

   The Logistic Regression classifier is shown as a gradient, but it is essentially a straight line boundary. While this technically performs the same on the test set, the boundary seems underfit, as it is further from the misclassified points than the other classifiers.

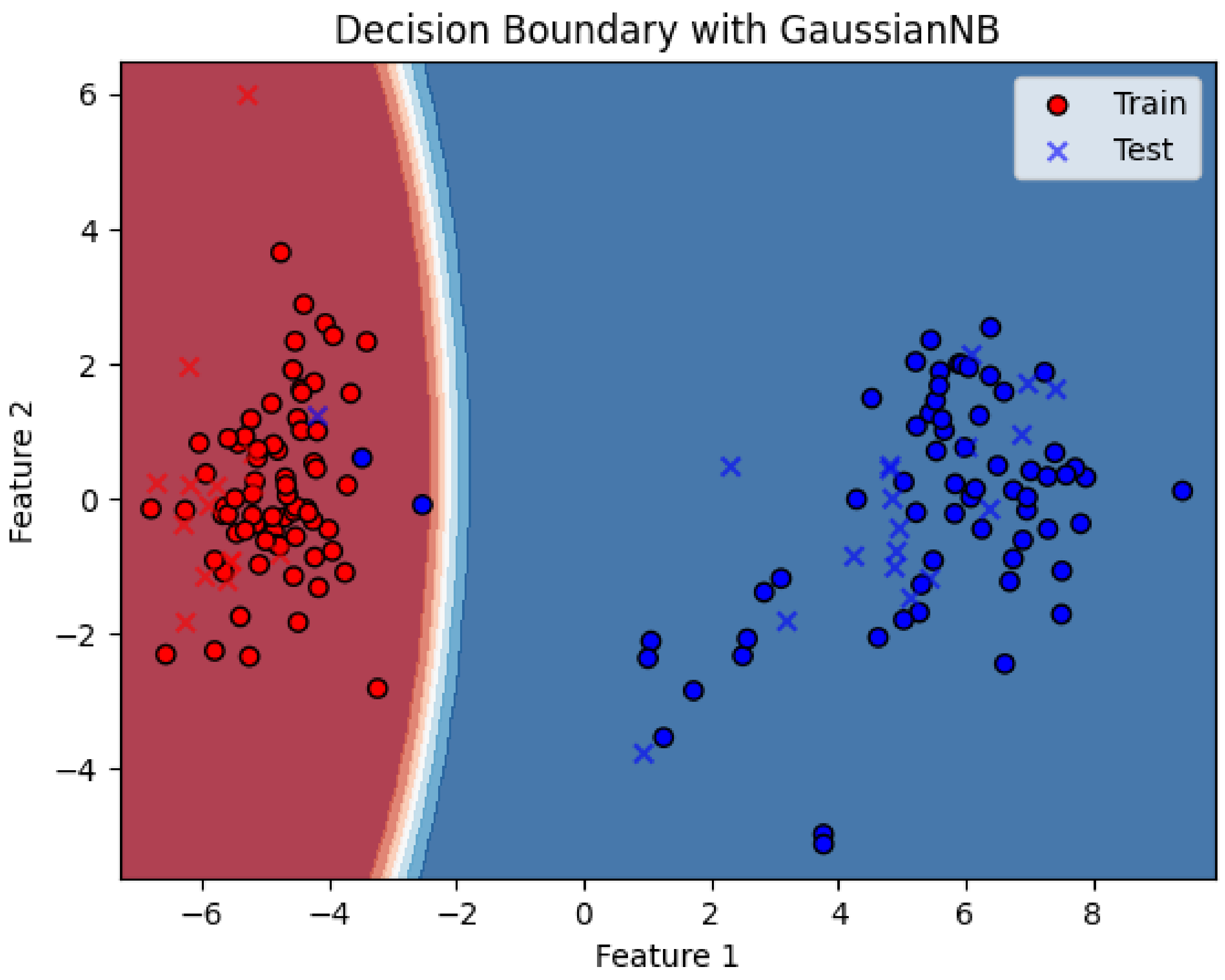
   The Naive Bayes classifier is non-linear and closely fit to the more clustered grouping. This puts the boundary close to one of the misclassified training points. If I were worried about the KNN overfitting, this model would provide a good middle ground.



Decision Boundary with KNeighborsClassifier
Train
Test
Feature 1
Feature 2
6
4
2
0
−2
−4
−6
−4
−2
0
2
4
6
8



Decision Boundary with LogisticRegression
Train
Test
Feature 1
Feature 2
6
4
2
0
−2
−4
−6
−4
−2
0
2
4
6
8



References

[1] Michael Percy (https://math.stackexchange.com/users/229646/michael-percy). *Derivative of sigmoid function* $\sigma(x) = \frac{1}{1+e^{-x}}$. Mathematics Stack Exchange. URL:https://math.stackexchange (version: 2017-09-01). 2017. eprint: https://math.stackexchange.com/q/1225116. URL: https://math.stackexchange.com/q/1225116.

[2] Brent T Speelman. 2025.

[3] Wikipedia contributors. *Karush–Kuhn–Tucker conditions — Wikipedia, The Free Encyclopedia*. [Online; accessed 13-October-2025]. 2025. URL: https://en.wikipedia.org/w/index.php?title=Karush%E2%80%93Kuhn%E2%80%93Tucker_conditions&oldid=1310095500.