Written Questions:

1. 4.1
2. 4.2
3. 4.3
4. Congrats! You're the mayor of Northfield! As mayor — you are interested in mitigating damage from potential flooding of the Cannon river. It is of the utmost importance that you begin preparations for flood control early (e.g., ordering sandbags, etc.) if you believe that there is going to be a flood this year. To help you make this prediction, you do the same thing you do every year: reach-out to two teams of hydrologists from Carleton (team A, and team B) to get their opinions. For simplicity, assume that the Cannon flooding in a year is a binary outcome, and that these same teams have made predictions for the past 20 years. From your records, you know that over the past 20 years, the Cannon has flooded 4 times. This time, team A predicts that the Cannon will not flood, and points-out that they have achieved 80% accuracy in predictions over the past 20 years. On the other hand, Team B predicts that the Cannon *will* flood, but, according to your records, they have only achieved 60% prediction accuracy over the past 20 years. Which team's prediction should you base your decision off of? If you can't decide — what additional information (if any) would you like to know from the teams? Ideally, your answer should include discussions of accuracy, precision, recall, constant prediction, etc.
5. Criminal trials in the United States can be viewed as classifiers: juries attempt to make predictions based on evidence about the guilt or innocence of a defendant. Even well-intentioned juries make mistakes; the Innocence Project, for example, is a non-profit organization that "is committed to exonerating wrongly convicted people through the use of DNA testing;"[1] they have helped to overturn over 300 wrongful convictions. Blackstone's Ratio,[2] a legal philosophy credited to Juror William Blackstone, states:

   > It is better that ten guilty persons escape than that one innocent suffer.

   Discuss how Blackstone's ratio relates to precision and recall. In your discussion, you should answer the question of whether Blackstone's Ratio advocates for high-precision juries or high-recall juries.
6. Logistic regression estimates binary probabilities, given a feature vector x. The probability that $y = 1$ is modeled by computing the sigmoid of a dot product of w and x plus a bias, i.e.: $P(y = 1|x) = \sigma(w^T x + b)$ and $P(y = 0|x) = 1 - \sigma(w^T x + b)$. For a true datapoint (x_i, y_i), we can compute the likelihood of the logistic regression parameters w, b given y_i (the true label) and x_i (the vector of features) as

$$\begin{aligned}\texttt{Likelihood}(w, b|x_i, y_i) &= P(y = y_i|W, b, x_i) \\ &= \sigma(w^T x_i + b)^{y_i} \cdot (1 - \sigma(w^T x_i + b))^{1-y_i}\end{aligned}$$

   Note that we have used y_i, the true label, as a "switch" variable. So — if y_i is 1, the likelihood of the parameters is estimated by $\sigma(w^T x_i + b)$. If the true label is 1, then the higher $w^T x_i + b$ is, the more likely W and b are. If y_i is 0, the likelihood of the parameters is estimated by $1 - \sigma(w^T x_i + b)$, i.e., for datapoints with true label 0, the smaller $w^T x_i + b$ is, the more likely the parameters w, b are.

   Binary crossentropy is a loss function that attempts to maximize the log-likelihood of all of the datapoints (because the datapoints are assumed to be independent draws, we can compute their joint probability as a product, i.e. $P((x_1, y_1), (x_2, y_2), (x_3, y_3) \ldots |w, b)$ is assumed to be equal to $P(x_1, y_1|w, b) \cdot P(x_2, y_2|w, b) \cdot P(x_3, y_3|w, b) \ldots$. Maximizing log likelihood is equivalent to minimizing negative log likelihood. Letting $z_i = w^T x_i + b$, the *loss function* (which we will minimize — lower loss is good!) for logistic regression

[1] `https://www.innocenceproject.org/`
[2] https://en.wikipedia.org/wiki/Blackstone%27s_ratio

with binary crossentropy loss is:

$$\begin{aligned}
-\log(\texttt{Likelihood}(w,b|\{x_i,y_i\})) &= -\log(\prod_i P(y=y_i|w,b,x_i)) \\
&= -\sum_i \log(P(y=y_i|w,b,x_i)) \\
&= -\sum_i \log(\sigma(z_i)^{y_i}\cdot(1-\sigma(z_i+b))^{1-y_i}) \\
&= -\sum_i \big[y_i\cdot\log\sigma(z_i)+(1-y_i)\log(\sigma(-z_i))\big] \\
\mathcal{L}(\{x_i,y_i\},w,b) &= \sum_i \big[-y_i\cdot\log\sigma(z_i)-(1-y_i)\log(\sigma(-z_i))\big]
\end{aligned}$$

Let the loss for datapoint i be defined as the term in the sum corresponding to datapoint i, i.e., $\mathcal{L}_i = \big[-y_i\cdot\log\sigma(z_i)-(1-y_i)\log(\sigma(-z_i))\big]$ where $z_i = w^T x_i + b$.

(a) When $y_i = 1$, what is the algebraic expression for the loss $\mathcal{L}_i$ incurred by a given setting of w, b (in terms of w, b, x_i)?

(b) Say w, b were set such that $w^T x_i + b$ was 3 for this example. What is $P(y_i = 1|x)$? What is the raw value of the loss? Is this a good setting of the parameters for this example, or a bad setting?

(c) Say instead that w, b were set such that $w^T x_i + b$ was -5 for this example. What is $P(y_i = 1|x)$? What is the raw value of the loss? Is this a good setting of the parameters for this example, or a bad setting?

(d) Say instead that $w^T x_i + b$ was still -5, but y_i was actually 0. What is $P(y_i = 0|x_i)$? What is the algebraic expression for the loss incurred on this example for a given setting of w, b? What is the raw value of the loss?

(e) To minimize the negative log likelihood with respect to the vector w and the bias value b for a datapoint (x_i, y_i), we may use a gradient-based method like gradient descent. Thus — we need to compute the gradient of the loss function with respect to each parameter. We will drop the subscript i in this question for notational ease...

i. What is $\frac{d\mathcal{L}}{dz} = \frac{d}{dz}\big[-y\cdot\log\sigma(z)-(1-y)\log(\sigma(-z))\big]$? Potentially helpful facts: $\frac{d}{dz}\sigma(z) = \sigma(z)\cdot(1-\sigma(z))$ and $\frac{d}{dz}\log(z) = \frac{1}{z}$

ii. What is $\frac{dz}{db} = \frac{d}{db}(w^T x + b)$?

iii. What is $\frac{dz}{dw_i} = \frac{d}{dw_i} w^T x + b = \frac{d}{dw_i}\sum_j w_j x_j + b$?

iv. Wrapping up — what is the gradient of the loss $\mathcal{L}$ with respect to the parameter b? (Note that the chain rule states: $\frac{d\mathcal{L}}{db} = \frac{d\mathcal{L}}{dz}\frac{dz}{db}$).

v. Wrapping up — what is the gradient of the loss $\mathcal{L}$ with respect to the parameter w_i? (Note that the chain rule states: $\frac{d\mathcal{L}}{dw_i} = \frac{d\mathcal{L}}{dz}\frac{dz}{dw_i}$)

Now that you've derived the gradient of the loss with respect to each parameter, you could write your own gradient descent loop, i.e., with learning rate α, $w_i := w_i - \alpha * \frac{d\mathcal{L}}{dw_i}$ for each w_i, and $b := b - \alpha * \frac{d\mathcal{L}}{db}$. You aren't required to do so, but you did do the hardest parts!

7. Please complete `intro.py` which contains some numpy introduction exercises. Insert your answers in the code (see the instructions at the top of the file).

Programming Assignment:

In this assignment, you will be implementing four document classifiers, and applying those classifiers to a dataset of movie reviews. Your goal will be to predict whether a given movie review is positive or negative based on the text of the review. You will then evaluate the performance of your classifiers using four metrics.

1. Implement `classify_doc_hand_design`. This function takes in an input document, and checks to see if any of the words in the argument `valid_words` appear in the input. If a word appears in the input, e.g., `good`, then the word's corresponding score (e.g., +1) is added to a running total "score" for the document. Once all of the words have been checked, if the score is greater than zero, this function returns 1, otherwise, it returns 0.

2. Implement a naive bayes classifier. My solution splits this process into two functions — one that precomputes probabilities for each word, i.e., it returns two dictionaries: one that maps a word to its smoothed probability in the positive class, and one that maps a word to its smoothed probability in the negative class. Additionally, it returns the number of positive and negative training documents there are. Next, I wrote `classify_doc_naive_bayes`, which takes in the output of the probability computing function, and computes the summed log probability of each class (according to the naive bayes assumption).

3. Read over the function `get_logistic_regression`, which gets an sklearn model that performs logistic regression. Note that we could have trained our own logistic regression model using gradient descent, but this is much easier, and has some fancy features! Make sure you understand this code — check out the sklearn documentation for `LogisticRegressionCV`, `StandardScalar`, and `Pipeline`. There are some questions in the comments – please write python code that prints out the answer to them (or — in one case, write a short answer in the comments).

4. Implement `classify_doc_logistic_regression`, which takes in a document, the vocabulary mapping words to indices, and the logistic regression model from `get_logistic_regression_model`; the output is the prediction of the model in the input document (using sklearn's `model.predict(x)` will be helpful here).

5. Implement the evaluation functions `get_accuracy`, `get_precision`, `get_recall`, and `get_f1`. These functions take in a numpy array of true labels and a numpy of predicted labels, and compute the corresponding evaluation metric. While it's not required, you should try to use vectorization as much as possible, here. None of my implementations have an explicit python for loop, and none of them exceed 4 lines of code. You don't need to be that efficient, but I thought I would share, for reference!

6. What results accuracy, precision, recall, and f1 score did your models achieve? Report your results in a results table like this (included are my numbers, for reference... but don't just copy my results! I will be checking your code to make sure it runs!).

	accuracy	precision	recall	f1
Constant Prediction	74.90	N/A	N/A	N/A
Hand-designed classifier	64.60	30.57	32.27	31.40
Naive Bayes	83.30	67.36	64.94	66.13
Logistic Regression	85.70	78.42	59.36	67.57