CIS 419/519 Applied Machine Learning
Assignment 3

Due: March 26, 2019 11:59pm

Instructions

Read all instructions in this section thoroughly.

Collaboration: Make certain that you understand the collaboration policy described on the course website.

You should feel free to talk to other members of the class in doing the homework. I am more concerned that you learn how to solve the problem than that you demonstrate that you solved it entirely on your own. You may *discuss* the homework to understand the problems and the mathematics behind the various learning algorithms, but you must **write down your solutions yourself**. In particular, **you are not allowed to share problem solutions or your code with any other students.**

We take plagiarism and cheating very seriously, and will be using automatic checking software to detect academic dishonesty, so please don't do it.

You are also prohibited from posting any part of your solution to the internet, even after the course is complete. Similarly, please don't post this PDF file or the homework skeleton code to the internet.

Formatting: This assignment consists of two parts: a problem set and program exercises.

For the problem set, you must write up your solutions electronically and submit it as a single PDF document. We will not accept handwritten or paper copies of the homework. Your problem set solutions must use proper mathematical formatting.

All solutions must be typeset using LaTeX; handwritten solutions of any part of the assignment (including figures or drawings) are not allowed.

Your solutions to the programming exercises must be implemented in python, following the precise instructions included in Part 2. Portions of the programming exercise will be graded automatically, so it is imperative that your code and output follows the specified API. A few parts of the programming exercise asks you to create plots or describe results; these should be included in the same PDF document that you create for the problem set.

Homework Template and Files to Get You Started: All files needed for this assignment are available in an archive posted online at
`https://www.seas.upenn.edu/~cis519/spring2019/homework/hw3/hw3_skeleton.zip`.

Citing Your Sources: Any sources of help that you consult while completing this assignment (other students, textbooks, websites, etc.) ***MUST*** be noted in the your PDF file. This includes anyone you briefly discussed the homework with. If you received help from the following sources, you do not need to cite it: course instructor, course teaching assistants, course lecture notes, course textbooks or other readings.

Submitting Your Solution: You will submit this assignment through **both** Gradescope and Canvas; submission will open approximately one week before the due date.

Both the PDF with your written responses and your python code must be submitted to **both** Canvas and Gradescope. Why do we require that you submit to both sites? Gradescope has better automatic unit testing functionality, and Canvas allows us to run plagiarism detection software. Unfortunately this means that we need you to submit the assignment in two places so we can get the best of both worlds.

All of the files should be uploaded directly; there should not be any directory structure.

Assignment Version 20190313

CIS 519 ONLY Problems: Some parts of the assignment will be marked as “[CIS 519 ONLY]”. Only students enrolled in CIS 519 are required to complete these problems. However, we do encourage students in CIS 419 to read through these problems, although you are not required to complete them.

All homeworks will receive a percentage grade, but CIS 519 homeworks will be graded out of a different total number of points than CIS 419 homeworks. Students in CIS 419 choosing to complete CIS 519 ONLY exercises will not receive any credit for answers to these questions (i.e., they will not count as extra credit nor will they compensate for points lost on other problems).

Philosophy Behind This Homework We are now into the portion of the course where we’re really focusing on applications. The problem set will give you some further practice with probability, including some types of option we haven’t explored, such as the “reject” option. The first programming exercise is a challenge, permitting you to test your ability to learn a classifier and apply it to unseen dats. Like in the real world, your success (i.e., your grade) will be partially based on how well the ML model does on new data! The second programming exercise will give you some experience working with text data – the main application we’re studying in lecture.a

PART I: PROBLEM SET

Your solutions to the problems will be submitted as a single PDF document. Be certain that your problems are well-numbered and that it is clear what your answers are.

1 Probability decision boundary (10pts)

Consider a case where we have learned a conditional probability distribution $P(y \mid \mathbf{x})$. Suppose there are only two classes, and let $p_0 = P(y = 0 \mid \mathbf{x})$ and let $p_1 = P(y = 1 \mid \mathbf{x})$. A loss matrix gives the cost that is incurred for each element of the confusion matrix. (E.g., true positives might cost nothing, but a false positive might cost us \$10.) Consider the following loss matrix:

	$y = 0$ (true)	$y = 1$ (true)
$\hat{y} = 0$ (predicted)	0	10
$\hat{y} = 1$ (predicted)	5	0

(a) Show that the decision $\hat{y}$ that minimizes the expected loss is equivalent to setting a probability threshold θ and predicting $\hat{y} = 0$ if $p_1 < \theta$ and $\hat{y} = 1$ if $p_1 \geq \theta$.

(b) What is the threshold for this loss matrix?

2 Double counting the evidence (15pts)

Consider a problem in which the binary class label $Y \in \{T, F\}$ and each training example $\mathbf{x}$ has 2 binary attributes $X_1, X_2 \in \{T, F\}$.

Let the class prior be $p(Y = T) = 0.5$ and $p(X_1 = T \mid Y = T) = 0.8$ and $p(X_2 = T \mid Y = T) = 0.5$. Likewise, $p(X_1 = F \mid Y = F) = 0.7$ and $p(X_2 = F \mid Y = F) = 0.9$. Attribute X_1 provides slightly stronger evidence about the class label than X_2.

Assume X_1 and X_2 are truly independent given Y. Write down the Naive Bayes decision rule.

(a) What is the expected error rate of Naive Bayes if it uses only attribute X_1? What if it uses only X_2?

The expected error rate is the probability that each class generates an observation where the decision rule is incorrect. If Y is the true class label, let $\hat{Y}(X_1, X_2)$ be the predicted class label. Then the expected error rate is $p(X_1, X_2, Y \mid Y \neq \hat{Y}(X_1, X_2))$.

(b) Show that if Naive Bayes uses both attributes, X_1 and X_2, the error rate is 0.235, which is better than if using only a single attribute (X_1 or X_2).

(c) Now suppose that we create new attribute X_3 that is an exact copy of X_2. So for every training example, attributes X_2 and X_3 have the same value. What is the expected error of naive Bayes now?

(d) Briefly explain what is happening with Naive Bayes (2 sentences max).

(e) Does logistic regression suffer from the same problem? Briefly explain why (2 sentences max).

3 Reject option (CIS 519 ONLY – 10pts)

In many applications, the classifier is allowed to "reject" a test example rather than classifying it into one of the classes. Consider, for example, a case in which the cost of misclassification is \$10 but the cost of having a human manually make the decision is only \$3. We can formulate this as the following loss matrix:

	$y = 0$ (true)	$y = 1$ (true)
$\hat{y} = 0$ (predicted)	0	10
$\hat{y} = 1$ (predicted)	10	0
reject	3	3

(a) Suppose $p(y = 1|\mathbf{x}) = 0.2$. Which decision minimizes the expected loss?

(b) Now suppose $p(y = 1|\mathbf{x}) = 0.4$. Now which decision minimizes the expected loss?

(c) Show that in cases such as this there will be two thresholds, θ_0 and θ_1, such that the optimal decision is to predict 0 if $p_1 < \theta_0$, reject if $\theta_0 \le p_1 \le \theta_1$, and predict 1 if $p_1 > \theta_1$.

(d) What are the values of these thresholds for the following loss matrix?

	$y = 0$ (true)	$y = 1$ (true)
$\hat{y} = 0$ (predicted)	0	10
$\hat{y} = 1$ (predicted)	5	0
reject	3	3

PART II: PROGRAMMING EXERCISES

1 Challenge: Generalizing to Unseen Data (30 pts)

One of the most difficult aspects of machine learning is that your classifier must generalize well to unseen data. In this exercise, we are supplying you with labeled training data and *unlabeled* test data. Specifically, you will *not* have access to the labels for the test data, which we will use to grade your assignment.

Fit the best model that you can to the given data and then use that model to predict labels for the test data. It is these predicted labels that you will submit, and we will grade your submission based on your test accuracy (relative to the best performance you should be able to obtain).

Here is some background on the data set:

Five days after meteors struck the DC metropolitan area, local hospitals have been flooded with people suffering from a virus that physicans have never seen before. Although they're suspecting the virus might have been carried on or dispersed by the meteor, they're not sure. What they do know is that left untreated, the disease is fatal within a few days, and it's very easy to miss until the symptoms become extreme. The treatment is painful, but extremely effective if the disease is caught early. Consequently, early screening is critical to saving lives, and every misclassification means that someone either dies or suffers pain during the unnecessary treatment. There aren't enough resources to treat everyone, so they need to predict who has the disease for treatment.

They've gathered some data from local populations using established questionnaires and lab tests, and physicians have spent extensive time to determine whether or not each patient is infected. Your task is to learn a model that can generalize to new patients. The description of the data can be found in `README.txt`. The labeled training data stored in CSV format in `challengeTrainingLabeled.csv`. Each patient is labeled as either being infected '1' or not being infected '0'. We will not provide any further information on the data set or features.

You are welcome to use whatever machine learning methods you like to solve this problem, including any we've covered, and any others that we have not. You may train the model however you like. You are allowed to use any pandas/numpy/sklearn functions, etc. for cleaning the data and predicting whether or not a person is infected. You do not need to implement the machine learning method yourself, but may if you wish. If you can think of a way that the unlabeled data in `challengeTestUnlabeled.csv` would be useful during the training process, you are welcome to let your classifier have access to it during training.

To accomplish this challenge, you will need to do extensive exploratory analysis of the data. Start by plotting the data, analyzing each column, etc. Add and remove features where you find it necessary. Throughout the entire process document your steps.

In your writeup, describe the steps you took. What worked / didn't work? How did you determine which model to choose? What steps did you take to clean the data? Report the performance of your model as best you can estimate on the data.

Once your model is working, use the trained classifier to predict a binary label $y \in \{0, 1\}$ for each unlabeled instance in `challengeTestUnlabeled.csv`. Your implementation should output a CSV file with only two columns: the patient_id and the predicted label, such as

```
patient_id, label
77636, 1
82621, 0
79864, 1
82635, 0
78876, 0
82939, 1
...
```

with one prediction per line. (Note that the labels above are not necessarily the true labels for the corresponding patient_ids.) Be very careful not to shuffle the instances in `challengeTestUnlabeled.csv`; the first predicted label should correspond to the first unlabeled instance in the testing data. The number of predictions should match the number of unlabeled test instances.

You should submit two items:
1.) A python program called `challenge.py` that automatically cleans and processes the given data set, trains your ML model, reports its performance, and outputs the predictions for the test data in the CSV format described above.
2.) A set of predictions on the given test data saved in a file called `predictions.csv`. The file will contain just one line for each instance in the test data set with two columns, plus one additional header line with the column titles. Again, be careful not to shuffle the test instances; the order of the predictions should match the order of the test instances.

Your grade will be based upon both your writeup and the performance (which we will compute) of your model on unlabeled test data. Note that we will only be computing the performance of your model during grading; you will not receive any feedback on the performance of your test predictions until you get your grade back for this assignment. It is up to you to estimate how well your model performs based on the given training data, and use this to make the best test predictions you can.

Left unchecked, the virus will decimate the population. Good luck; the fate of humanity rests in your hands.

2 Text Classification

In this section, you will be dealing with textual data. Specifically, you will be performing text classification on documents to determine the authors. Such a classifier can be useful to assert authorship, or to deanonymize written documents.

Your first task is to download `articles.zip`. In each of the `articles/train` and `articles/test` folders, there are 23 folders, each corresponding to an author; each folder contains articles written by that author.

2.1 Multinomial Naive Bayes

For feeding an input to the multinomial Naive Bayes model, we need categorical input features. Start by creating a vocabulary and map each unique word to an identifier. For this, you would need to tokenize the documents and map each unique token to unique integer. It would help to look at the NLTK tokenize package. Once you have created these mappings, represent each document as a feature vector of counts of the words in the vocabulary. You may use the sklearn CountVectorizer function for this.

Now that you have the documents represented as feature vectors, fit a Naive Bayes model on the training data and report the test accuracy, precision, and recall. You should use the Naive Bayes implementation provided with sklearn. Choose 10 authors, report the top 10 most-frequently-used words for each of those authors in a well-formatted table, sorted by author.

Experiment with the *stopwords*, *max_features* and the *ngram_range* arguments in the `MultinomialNB` function. Focus on tuning these parameters to optimize performance on the training data set – remember, you should never use the test data set for tuning! Then, once you have the best tuned classifier, evaluate its performance on the test data. Report the top 10 most-frequently-used words/ngrams for the same 10 authors, and comment on the results of the experiments.

2.2 TF-IDF Features

Instead of using word counts, in this section, you will represent the document in terms of the TF-IDF features. First, compute the TF-IDF vectors using the training documents and obtain feature representations of all documents.

Train and evaluate the following classifiers using the TF-IDF features as inputs:

- SVM with the following kernels:
  - Linear Kernel
  - RBF Kernel
  - Cosine Similarity
- Gaussian Naive Bayes
- Multinomial Logistic Regression

Experiment with the *stopwords*, *max_features* and the *ngram_range* arguments in the `sklearn.TfIdfVectorizer` function. Report, compare and comment on the results you obtain for each classifier. How does this relate to the results you obtained before using the word counts?

2.3 Latent Dirichlet Allocation [CIS 519 Only]

In this section, you will fit an unsupervised model, latent Dirichlet allocation (LDA), on the documents. First off, read the original LDA paper: `http://www.jmlr.org/papers/volume3/blei03a/blei03a.pdf`. Don't worry about the details of the math, but make absolutely certain that you understand the high-level idea. We won't be implementing LDA; we will only be using it.

Pick the same 10 authors as in the Naive Bayes question. For each of the authors, get the average probability of each LDA topic used by the author in their test articles. Experiment with three different numbers of topics and report the results. Do you see authors being biased towards using some topics more frequently than others? Comment on the findings. Moreover, for each author, get the three most-used topics and report the 10 most-probable words for each of these topics.