Homework 1 (Sept. 8)

Deadline: Monday, Sept. 22, at 11:59pm.

Submission: You need to submit one file through Quercus with your answers to Questions 1 and 2. It should be a PDF file titled `hw1_writeup.pdf`. You can produce the file however you like (e.g. LaTeX, Microsoft Word, scanner), as long as it is readable.

Marking: Your mark will be out of 7.5. This mark will be your mark on *either* question 1 *or* question 2. We will decide which question to mark after the deadline, and we will mark the same question for everyone in the class. To aid your learning, we will be releasing the solutions to both questions, and covering the solutions in office hours.

Neatness: We reserve the right to deduct a point for neatness, if we have a hard time reading your solutions or understanding the structure of your code.

Late Submission: 10% of the total possible marks will be deducted for each day late, up to a maximum of 3 days. After that, no submissions will be accepted.

Computing: To install Python and required libraries, see the instructions on the course web page.

Other Policies: See the syllabus[1] for detailed collaboration, generative AI, and academic integrity policies.

1. [**7.5pts**] **Classification with Nearest Neighbours.** In this question, you will use the `scikit-learn`'s KNN classifer to classify real vs. fake news headlines. The aim of this question is for you to read the `scikit-learn` API and get comfortable with training/validation splits.

   We will use a dataset of 1298 "fake news" headlines (which mostly include headlines of articles classified as biased, etc.) and 1968 "real" news headlines, where the "fake news" headlines are from https://www.kaggle.com/mrisdal/fake-news/data and "real news" headlines are from https://www.kaggle.com/therohk/million-headlines. The data were cleaned by removing words from titles not part of the headlines, removing special characters and restricting real news headlines after October 2016 using the word "trump". The cleaned data and starter code are available as `clean_real.txt` and `clean_fake.txt` in `hw1_starter.zip` on the course webpage. It is expected that you use these cleaned data sources for this assignment.

   We are providing starter code in `hw1_starter.zip` for this assignment. This folder contains the data, `hw1.py`, and `hw1.ipynb`. To run the code you simply need to to run `hw1.py` using your favourite Python interpreter, or you can upload `hw1.ipynb` to JupyterHub or Google Colab to run as a notebook. To make the code correct, you will need to fill in the body of two functions. If you do this correctly, the code should run and output something that looks like the following:

   ```
   Selected K: 10
   Test Acc: 0.5
   ```

[1] https://www.cs.toronto.edu/~cmaddis/courses/sta314_f25/sta314_f25_syllabus.pdf

The numbers you get may be different from the ones above. Depending on the Python installation that you use, the numbers you get may be different from the numbers your colleagues get. Before you implement anything, or if you implement it incorrectly, the code may raise an Exception. This is expected behaviour.

You will build a KNN classifier to classify real vs. fake news headlines. Instead of coding the KNN yourself, you will do what we normally do in practice — use an existing implementation. You should use the `KNeighborsClassifier` included in `sklearn`. Note that figuring out how to use this implementation, its corresponding attributes and methods is a part of the assignment.

(a) **[3pts]** Complete the function `process_data`. It should do the following.

- First, split the entire dataset randomly into 70% training, 15% validation, and 15% test examples using `train_test_split` function (`https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html`). You can use the stratify option to keep the label proportions the same in the split.
- Then, preprocess the data using a `CountVectorizer` (`https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html#sklearn.feature_extraction.text.CountVectorizer`). You will need to understand what this preprocessing does, so you should play around with it. Also, the `CountVectorizer` should be fit only on the training set.

In your writeup, include the body of the function `process_data` that you wrote.

(b) **[3pts]** Complete the function `select_knn_model` that selects a k-NN classifer using a training set and a validation set to classify between real vs. fake news. This function should do the following.

- Iterate over k values between 1 to 20.
- For each k value, fit a `KNeighborsClassifier` to the training set, leaving other arguments at their default values.
- Measure the validation accuracy.
- Return the best choice of k and the corresponding model that has been fit to the training set with that value of k.

In your writeup, include the body of the function `select_knn_model` that you wrote, as well as the output of the `hw1.py` script.

(c) **[1.5pts]** Repeat part (b), passing argument `metric='cosine'` to the `KNeighborsClassifier`. You should observe an improvement in accuracy. How does `metric='cosine'` compute the distance between data points, and why might this perform better than the Euclidean metric (default) here? *Hint: consider the dataset* `['cat', 'bulldozer', 'cat cat cat']`.

2. **[7.5pts] High Dimensional Random Vectors Are...** In this question, you will study the properties of random vectors in high dimensions. For this question, let $X, Y \in \{-1, 1\}^d$ be two random vectors whose coordinates are all identically and independently distributed as -1 or 1 with equal probability. That is,

$$P(X = b, Y = a) = \frac{1}{4^d}, \qquad (0.1)$$

for all $a, b \in \{-1, 1\}^d$. Similar results to those we'll prove in this question hold for other distributions over random vectors.

(a) **[2pts]** Let X_i, Y_i be the ith coordinates of X and Y, respectively. Compute the following.

$$\mathbb{E}\left[\left(\sum_{i=1}^{d} X_i Y_i\right)^2\right] \tag{0.2}$$

(b) **[2pts] Probably Approximately Orthogonal.** Let θ be the angle between X and Y. Recall the formula

$$\cos\theta = \frac{\sum_{i=1}^{d} X_i Y_i}{\sqrt{\sum_{i=1}^{d} X_i^2}\sqrt{\sum_{i=1}^{d} Y_i^2}}. \tag{0.3}$$

Show that for all $\epsilon > 0$

$$P(|\cos\theta| < \epsilon) \geq 1 - \frac{1}{d\epsilon^2}. \tag{0.4}$$

That is, if I take two high dimensional random vectors, it is very likely that the angle between them is approximately 90°. *Hint: Use Markov's inequality. If Z is a non-negative random variable, then Markov's inequality states that for all $a > 0$,*

$$P(Z < a) \geq 1 - \frac{\mathbb{E}[Z]}{a}. \tag{0.5}$$

You need to identify an appropriate random quantity Z, and apply Markov's inequality.

(c) **[2pts]** Compute the following.

$$\mathbb{E}\left[\sum_{i=1}^{d}(X_i - Y_i)^2\right] \tag{0.6}$$

$$\mathrm{Var}\left[\sum_{i=1}^{d}(X_i - Y_i)^2\right] \tag{0.7}$$

(d) **[1.5pts] Probably Approximately The Same Distance Away.** The two vectors $(1, \ldots, 1)$ and $(-1, \ldots, -1)$ are the furthest apart that any two vectors in $\{-1, 1\}^d$ can be. The squared distance between these two is $4d$. Thus,

$$\frac{1}{4d}\sum_{i=1}^{d}(X_i - Y_i)^2 \tag{0.8}$$

is the ratio of the squared distance between X, Y and the maximum possible squared distance. Show that for all $\epsilon > 0$

$$P\left(\left|\frac{1}{4d}\sum_{i=1}^{d}(X_i - Y_i)^2 - \frac{1}{2}\right| < \epsilon\right) \geq 1 - \frac{1}{4d\epsilon^2} \tag{0.9}$$

That is, if I take two high dimensional random binary vectors, it is very likely that they are $2d$ squared distance apart, or half as far apart as the furthest possible distance. *Hint: This question also uses Markov's inequality, see question 2(b).*