Homework 3: Classification and Language Models (Deadline: November 28, 2022, 11:59pm)

Problem 1: Rule-based Classifiers

Consider the following classification rule extracted from the medical history of patients:

BloodPressure > 150 → *HeartDisease=Severe*

Suppose the coverage of the rule is 5% and the accuracy is 60% on the training data. Coverage refers to the proportion of patients in the training set who satisfy the rule condition (i.e., left-hand side of the rule) while accuracy is the fraction of such patients (who satisfy the rule condition) who also have heart disease.

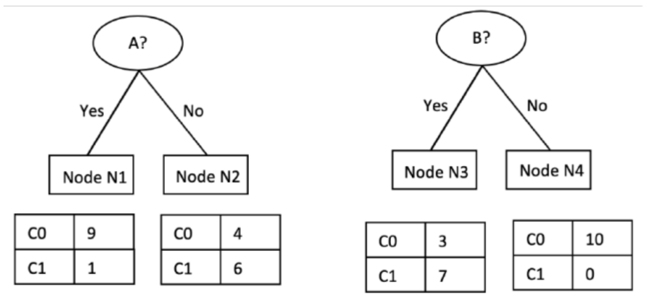
1. Explain whether coverage and accuracy (1) will stay the same, (2) will increase or stay the same, (3) will decrease or stay the same, or (4) can change in either direction (increase/decrease), if we add the conjunct *CholesterolLevel > 245*, to the left-hand side of the rule.

2. Explain whether coverage and accuracy (1) will stay the same, (2) will increase or stay the same, (3) will decrease or stay the same, or (4) can change in either direction (increase/decrease) if the rule condition is *BloodPressure > 200* instead of *150*.

3. Instead of a 3-class problem (HeartDisease is *Severe, Mild, and None*), suppose we reduce this to a 2-class problem (HeartDisease is *Yes or No*), where all the training examples assigned to the *Severe* and *Mild* classes are in the *Yes* category while the remaining is in the *No* category. Explain whether coverage and accuracy (1) will stay the same, (2) will increase or stay the same, (3) will decrease or stay the same, or (4) can change in either direction (increase/decrease) when the rule becomes *BloodPressure > 150* → *HeartDisease=Yes*.

Problem 2: Decision Tree

Below we show two possible ways of splitting the current node into two nodes.



1. What is the Gini Impurity of the current node before splitting?
2. What is the information gain after splitting using feature A? (Using Entropy)
3. What is the information gain after splitting using feature B? (Using Entropy)
4. Which is the better way to split (A or B)?

Problem 3: Classification problem

The learning rule of a classification algorithm (i.e., Perceptron) is shown as follows:

Note: We didn't introduce perceptron algorithm formally in our lectures. However, the information given below should be sufficient to answer this question. No additional knowledge of perceptron algorithm is required to answer this question.

Given (1) Dataset $X = \{x_1, \cdots, x_i, \cdots, x_m\}$ where $x_i = [A_1, \cdots, A_j, \cdots, A_n]$ (n is the number of attributes/features) and (2) the learning rate η.
[**STEP 1**] If we use the bias w_0 and then add $A_0 = 1$ to x_i, i.e., $x_i = [1, A_1, \cdots, A_j, \cdots, A_n]$.
[**STEP 2**] Initialize weights as 0 or small numbers $\{w_0, w_1, \cdots, w_n\}$ (if there is no bias, then remove $\{w_0\}$).
[**STEP 3**] Update weights according to the learning rule of Perceptron as follows:

```
Repeat until converge:
  For $i = 1$ to $m$ ($m$ is the number of data):
    Calculate the output value
      $$g(W^T x_i) = \begin{cases} 1, \text{ if } W^T x_i \geq 0.5; \\ 0, \text{ otherwise.} \end{cases}$$
    For $j = 0$ to $n$ (If using bias; otherwise $j = 1$. $n$ is the number of features.):
      Update $w_j^{new} = w_j + \eta(y_i - g(W^T x_i))A_j^{(i)}$
```

Suppose that we are going to work on AND Gate Problem using perceptron. The AND gate returns if and only if both inputs are true as shown in the table below

Data	Feature 1	Feature 2	Ground Truth (y)
x_1	0	0	0
x_2	0	1	0
x_3	1	0	0
x_4	1	1	1

Let the learning rate (η) be 0.5. Please list the weights after the first round (i.e., when T = 1) for the following two scenarios:

1. Using the bias w_0 and initializing $w_0 = 0.1, w_1 = 0.3, w_2 = 0.5$
2. Without using the bias and initializing $w_1 = 0.9, w_2 = 0.9$

Problem 4: Vector Space Models

In a text data corpus, there are three documents

- DocA = 'Document classification or document categorization is a problem in computer science.'
- DocB = 'Document classification task is to assign a document to one or more classes or categories.'
- DocC = 'Documents may be classification according to their subjects or according to other attributes.'

After removing stop words and conducting stemming, we then have three vectors as follows:

- DocA = ['document', 'classify', 'document', 'category', 'problem', 'computer', 'science']
- DocB = ['document', 'classify', 'task', 'assign', 'document', 'class', 'category']
- DocC = ['document', 'classify', 'accord', 'subject', 'accord', 'attribute']

Based on the preprocessed text data, answer the following questions.

(1) Create a word vocabulary dictionary in order according to word appearance from DocA to DocC. For example, {0: 'document'; 1: 'classify'; ..., n: attribute}

(2) Represent each document as a vector with each word represented as 1 for present and 0 for absent from the vocabulary created in (a).

(3) Represent each document as a vector with the number of times each word appears in a document.

(4) Use Term Frequency-Inverse Document Frequency (TF-IDF) values to represent each document (i.e., DocA, DocB, DocC) as a vector.

Problem 5: Language Models (n-grams)

Consider the following training data:

```
<s> I am Sam </s>
<s> Sam I am </s>
<s> Sam I like </s>
<s> Sam I do like </s>
<s> do I like Sam </s>
```

Assume we have a bigram language model based on the above training data

1. What is the most probable next word predicted by the model for the following word sequences?

(1) <s> *Sam . . .*

(2) <s> *Sam I do . . .*

(3) <s> *Sam I am Sam . . .*

(4) <s> *do I like . . .*

2. Which of the following sentences is better, i.e., gets a higher probability with this model?

(5) <s> *Sam I do I like* </s>

(6) <s> *Sam I am* </s>

(7) <s> *I do like Sam I am* </s>

3. What is the perplexity of the following sentence

<S> I do like Sam

4. Now, consider a bigram language model with Laplace smoothing. Compute the following bigram probabilities:

$P(\texttt{do}|\texttt{<s>})$ $P(\texttt{do}|\texttt{Sam})$ $P(\texttt{Sam}|\texttt{<s>})$ $P(\texttt{Sam}|\texttt{do})$
$P(\texttt{I}|\texttt{Sam})$ $P(\texttt{I}|\texttt{do})$ $P(\texttt{like}|\texttt{I})$

5. Calculate the probabilities of the following sequences according to the bigram language model with Laplace smoothing. Which of the two sequences is more probable?

(8) `<s>` *do Sam I like*

(9) `<s>` *Sam do I like*

Problem 6: Programming Assignment (Classification)

Two datasets (hw3_dataset1, hw3_dataset2) can be found on ICON. Here is a short description of the first two datasets:

Each line represents one data sample. The last column of each line is class label, either 0 or 1. The rest columns are feature values, each of them can be a real-value (continuous type) or a string (nominal type).

hw3_dataset1: 569 observations, 31 attributes

hw3_dataset2: 462 observations, 10 attributes

Complete the following tasks

a. Implement four classification algorithms: **Nearest Neighbor, Decision Tree (binary), Naïve Bayes, SVM** (Normalize the data to avoid scaling issue, and/or apply regularization to avoid overfitting if needed.)
   Note: You are free to use existing libraries or packages for the classification algorithms.
b. Implement **Random Forests** based on the implementation of Decision Tree.
c. Implement **Boosting** based on the implementation of Decision Tree.
d. Adopt 10-fold **Cross Validation** to evaluate the performance of all methods on the provided two datasets in terms of **Accuracy, Precision, Recall, and F-1 measure**

Problem 7: Programming Assignment (Statistical Language Models)

For this task, we will use **restaurants.zip** dataset. This dataset contains two folders 'train' and 'test'. The train and test folders contain 38,688 and 19,803 restaurant review documents respectively. The files are in JSON format. As in the previous homework, Python programming language should be used for the programming assignment.

Document: In this dataset, document refers to each individual review document. Thus, each JSON file might contain multiple documents.

Complete the following tasks:

a. <u>Perform the preprocessing steps:</u>

i) Tokenization: Tokenize the review content of each document into tokens

ii) Normalization: Normalize tokens by removing individual punctuation marks (List of Punctuation marks: https://en.wikipedia.org/wiki/English_punctuation), converting tokens into lower cases, and recognizing digit numbers, e.g., integers and floats, and map them to a special symbol "NUM".

iii) Stemming: Stem the tokens back to their root form

iv) Stopword removal: Remove a standard list of stop words. Since the definition of stopwords is domain-specific, please use the stop words in the given "STOPWORDS.TXT" file as a standard list.

For tokenization and stemming, please feel free to use Python packages such as NLTK modules (https://www.nltk.org/api/nltk.tokenize.html#module-nltk.tokenize) (https://www.nltk.org/api/nltk.stem.html#module-nltk.stem)

b. <u>Validate Zipf's law</u>

i) For each token, go over all the review documents containing it (in both train and test folder), and accumulate its frequency in those documents, i.e., total term frequency (TTF)

ii) Order the tokens by their TTF in descending order

iii) Create a dot plot by treating each word's rank as x-axis and its TTF as y-axis. Please use log-log scale for this plot.

Deliverable: Report the generated curve for Zip'f law in log-log space.

c. <u>Construct a Controlled Vocabulary</u>

i) Generate all the bigrams (as discussed in our lecture slides) based on the resulting tokens from the preprocessing step (**only in the train folder**) and mix those bigrams with unigrams as our initial controlled vocabulary.

ii) Collect the Document frequency (DF) statistics for all the N-grams (i.e., unigram and bigram) in this initial controlled vocabulary (**only in the train folder**)

iii) Filter N-grams with DF smaller than 50.

Deliverable: Report the size of the resulting controlled vocabulary (i.e., total N-grams in it). Report the top 50 and bottom 50 N-grams according to DF in the resulting controlled vocabulary, and their corresponding Inverse Document Frequency (IDFs). IDFs should be estimated based on the **train folder** only.

d. <u>Compute Similarity between Documents</u>

i) With the above controlled vocabulary, each review document can be represented as a N-gram vector. Specifically, each dimension in this vector space is defined by the mix of unigrams and bigrams defined in the controlled vocabulary, while the weight for each unigram/bigram in a review document is defined by TF-IDF. Please use the sub-linear TF scaling (refer to the lecture slides for formula) to compute the normalized TF of each unigram/bigram in a review document.

ii) Use the above document representation to encode all the review documents in the test folder. Also, we have provided a query file (named QUERY.JSON) that contains five restaurant reviews outside the corpus. Construct the vector space representations for these five reviews (like what you did for the test folder) and identify the most similar reviews to them from the test folder. Please use **cosine similarity** as the similarity metric

Deliverable: For each document in the QUERY.JSON file, please report top 3 most similar review documents (including Author, content, and date) from the test folder and the corresponding cosine similarity.

e. <u>Maximum Likelihood Estimation (MLE) for Language Models with Proper Smoothing</u>

i) Use all the review documents in the **train folder** to estimate a unigram language model.

ii) Use all the review documents in the **train folder** to estimate two bigrams language models, i.e., <u>one using linear interpolation and another one using absolute discounting smoothing based on the unigram language model</u>. In linear smoothing, set the parameter $\boldsymbol{\lambda} = \mathbf{0.9}$ and in absolute discount smoothing, set the parameter $\boldsymbol{\delta} = \mathbf{0.1}$.

Deliverable: Report the top 10 words selected from the corresponding two bigram language models. Are the top 10 words the same from these two bigram language models? Please explain your observation.

f. Text Generation and Language Model Evaluation

i) Fixing the sentence length to 15, generate 10 sentences by sampling words from above estimated unigram, two bigram models. For the bigram models, please sample the first word of a sentence from the unigram models and then sample the next words from the corresponding bigram model.

ii) Compute the perplexity of the estimated language models, on all the review documents in the **test folder**. Please follow the definition to compute perplexity for every review document in the test folder and compute the mean and standard deviation of the resulting perplexities.

Deliverable: Report the 10 sentences generated from your unigram and two bigram language models. Please report the likelihood given the corresponding language model. Please report the mean and standard deviation of the perplexities on the **test folder**. Can you conclude which language model predicts the data in the **test folder** better? And why?

Submission. Make a zipped folder named "[HawkID]-Classification_LanguageModels.zip", where "[HawkID]" refers to your HawkID. In the folder, you should include: (a) **Report: Solution to problems 1- 5 and deliverables requested for the programming assignments. (b) A folder named Code, which contains all codes used in this homework**. Inside the folder, please have a file README which describes how to run your code. The folder name for your code should be "Code.zip". An ICON assignment page has been created for homework3. Please submit your zipped folder there.