Assignment 1

Due on November 14, 2022 (23:59:59)

Instructions. The goal of this problem set is to make you understand and familiarize with K-Nearest Neighbor Algorithm. There are two parts in this assignment. The first part involves a KNN-classification experiment and and the second part involves KNN-regression experiment.

1 PART 1: Classification of News Articles

In this part of the assignment, you will implement a nearest neighbor algorithm to classify determine which category a news article belongs to among five categories (Sport, Business, Politics, Entertainment, Tech) (see Table 1).You will also extend your implementation as weighted KNN algorithm.

ArticleId	Text	Category
651	wales silent on grand slam talk rhys williams says wales are still not thinking ...	sport
2034	car giant hit by mercedes slump a slump in profitability at luxury car maker mercedes has prompted ...	business
1582	howard truanted to play snooker conservative leader michael howard has admitted he used to play ...	politics
1407	rapper snoop dogg sued for rape us rapper snoop dogg has been sued for ...	entertainment
1532	security scares spark browser fix microsoft is working on a new version of its internet explorer web browser. ...	tech

Figure 1: Some examples from different categories in the dataset

1.1 Classification Dataset: News Articles Dataset [1]

- You can download the dataset from given link.
- Dataset consists of 1491 samples with 5 discrete (”Category” attribute) ground-truth class types.
- English News Dataset is a dataset provided to determine the category of a news article. It includes the following features:
  - **ArticleId:** Id number of the article
  - **Text:** Text of the article, could be incomplete
  - **Category:** Ground-truth category label of the news article (Sport, Business, Politics, Entertainment, Tech)

1.2 Approach

1.2.1 Understanding the data

You will be predicting the category of a news article from words that appear in the text. Is that feasible? Give 3 examples of specific keywords that may be useful, together with statistics on how often they appear in each of five categories' articles.

1.2.2 Implementing k Nearest Neighbor

You will represent your data with features and use them to train a classifier via the k Nearest Neighbor algorithm. You have to implement your own k Nearest Neighbor algorithm.

- Features: You will use Bag of Words (BoW) model which learns a vocabulary from all of the documents, then models each document by counting the number of times each word appears. You will use BoW with two options:
  - Unigram: The occurrences of words in a document(frequency of the word).
  - Bigram: The occurrences of two adjacent words in a document.

You may encounter words during testing that you haven't during training. This may be for a particular class or overall. Your code should deal with that. Hint: You can ignore unseen words on test documents or you can give a non-zero default value (0.001).

You have to use a dictionary for BoW representation. You can implement your own method to obtain BoW model or you can use Count Vectorizer function (`https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html`).

1.2.3 Analys the words

1. **Analyzing effect of the words on prediction**
   - List the 10 words whose presence most strongly predicts that the article belongs to specific category for each five categories.
   - List the 10 words whose absence most strongly predicts that the article belongs to specific category for each five categories.

   You can narrow down your dictionary by choosing specific words for articles belongs to all five categories. In other words, your classification results can be improved by selecting a subset of extremely effective words for the dictionary. TF-IDF (`https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfTransformer.html`) and Information Theory are good places to start looking. Reimplement the 1.2.2 and see the effect of using specific words on the task.

Compare the influence of presence vs absence of words on predicting whether the article is specific category of five different categories.

2. **Stopwords**
   You may find common words like “a”, “to”, and others in your list in 1.2.3(1). These are called stopwords. A list of stopwords is available in sklearn here. You can import this as follows:
   `from sklearn.feature_extraction.text import ENGLISH_STOP_WORDS`
   Now, list the 10 non-stopwords that most strongly predict that the article belongs to specific category for each five categories.

   Analyzing effect of the stopwords: Why might it make sense to remove stop words when interpreting the model? Why might it make sense to keep stop words?

1.3 Classification Performance Metric

You will compute "Accuracy", "Precision" and "Recall" of your model to measure the success of your classification method:

$$\textbf{Accuracy} = \frac{TP + TN}{TP + FP + FN + TN} \tag{1}$$

$$\textbf{Precision} = \frac{TP}{TP + FP} \tag{2}$$

$$\textbf{Recall} = \frac{TP}{TP + FN} \tag{3}$$

You will report accuracy, precision and recall of each test with respect to 5-fold cross validation and average accuracy, precision and recall of these tests.

1.4 Error Analysis for Classification

- Find a few misclassified samples and comment on why you think they were hard to classify.
- Compare performance of different feature choices and investigate the effect of important system parameters (number of training samples used, k in k-NN, etc.). Wherever relevant, feel free to discuss computation time in addition to the classification rate.

Steps to Follow for Classification

1. Read your classification data and implement BoW for preparing the features.

2. For the test samples
    - predict their classes using k-NN.
    - predict their classes using weighted k-NN.
3. Compute and report your accuracy, precision and recall of your different k-NN and weighted k-NN models with different k parameters (Your must experiment with this k parameters: **(1,3,5,7,9)**) on 5-fold cross validation.
4. Report your findings in "Error Analysis for Classification" section.

2 Medical Insurance Cost Estimation from Data

In this part of the assignment, you will implement a nearest neighbor algorithm to estimate medical insurance costs for people. You will also extend your implementation as weighted KNN algorithm.

A dataset [2] is provided for your training phase. In other words, you should split your training dataset into two sets; training set which will be used to learn model, and validation set which will be used to measure the success of your model. You will use 5-fold cross validation method which is explained in the class.

2.1 Regression Dataset: Medical Insurance Cost Estimation Dataset [2]

- You can download the dataset from given link.
- Dataset consists of 1338 samples with continues medical cost ("charges") value.
- **Attribute information for each sample in dataset:**
    1. **Age:** Age of primary beneficiary.
    2. **Sex:** Insurance contractor gender, female, male.
    3. **BMI:** Body mass index, providing an understanding of body, weights that are relatively high or low relative to height, objective index of body weight ($Weight(kg)/Height^2(m)$) using the ratio of height to weight, ideally 18.5 to 24.9.
    4. **Children:** Number of children covered by health insurance / Number of dependents.
    5. **Smoker:** Smoking.
    6. **Region:** The beneficiary's residential area in the US, northeast, southeast, southwest, northwest.

7. **Charges:** Individual medical costs billed by health insurance.

2.2 Regression Performance Metric

You will compute "Mean Absolute Error" of your model to measure the success of your regression method:

Mean Absolute Error (MAE) $= \frac{1}{n}\sum_{i=1}^{n}|d_i - \hat{d}_i|$
where,
d_i is the actual (ground-truth) *charges* value of sample
$\hat{d}_i$ is the predicted/estimated *charges* value of sample
n is the number of samples

You will report MAE of each test with respect to 5-fold cross validation and average MAE of these tests.

2.3 Feature Normalization

You will use min-max normalization on the features of your samples to re-scale each feature (feature/attribute column on data) between (0-1) range.

$$n_i = \frac{f_i - min(f)}{max(f) - min(f)} \tag{4}$$

Where, n_i represents the i^{th} normalized element of your specific feature column (f) and f_i represents the i^{th} original element of your specific feature column (f).

2.4 Error Analysis for Regression

- Compare performance of different feature normalization choices and investigate the effect of important system parameters (number of training samples used, k in k-NN, etc.). Wherever relevant, feel free to discuss computation time in addition to regression/estimation rate.

Steps to Follow for Regression

1. Read your regression data and transform it to the Numpy array collection.
2. For the test samples
    - estimate their *charges* values using k-NN. (with feature normalization and without feature normalization)
    - estimate their *charges* values using weighted k-NN. (with feature normalization and without feature normalization)

3. Compute and report your MAE of your different k-NN and weighted k-NN models with different k parameters (Your must experiment with this k parameters: **(1,3,5,7,9)**) on 5-fold cross validation.
4. Report your findings in "Error Analysis for Regression" section.

3 Implementation Details

- **You can't use ready-made libraries for your K-fold cross-validation/Shuffle methods for your data for both of your Part 1 and Part 2 implementations. You must implement these on your own**
- **You can't use ready-made libraries for your k-NN/weighted k-NN methods for your data for both of your Part 1 and Part 2 implementations. You must implement these on your own**
- **You can't use ready-made libraries for computing "Accuracy", "Precision", "Recall" and "MAE" metrics. You must implement these on your own**
- **You can't use ready-made libraries for computing "min-max" feature normalization.**
- You may use Numpy array functions for your intermediate implementation steps for your Part 1 and Part 2 implementations.
- You may use "Pandas" library for reading and writing/creating .csv files: `https://pandas.pydata.org/docs/index.html`

Submit

You are required to submit all your code in a Jupyter notebook, along with a report in ipynb format, which should also be prepared using Jupyter notebook. The code you submit should be thoroughly commented. Your report should be self-contained and include a concise overview of the problem and the details of your implemented solution. Feel free to include pseudocode or figures to highlight or clarify specific aspects of your solution. Finally, prepare a ZIP file named name-surname-a1.zip containing:

- assignment_1.ipynb (including your report and code)
- assignment_1.py (py file version of your ipynb file)
- Do not send the dataset.

Grading

- Code (60): kNN(Classification): 10 points, weighted kNN(Classification): 10 points, BoW: 20 points, kNN(Regression): 10 points, weighted kNN(Regression): 10 points.
- Report (40): Analysis of the results for prediction: 40 points.

  Notes for the report: Preparing good report is important as well as your solutions! You should explain your choices (Unigram, Bigram or both of their use for Bow, or constraints on data) and their effects to the results.

Academic Integrity

All work on assignments must be done individually unless stated otherwise. You are encouraged to discuss with your classmates about the given assignments, but these discussions should be carried out in an abstract way. That is, discussions related to a particular solution to a specific problem (either in actual code or in the pseudocode) will not be tolerated. In short, turning in someone else's work, in whole or in part, as your own will be considered as a violation of academic integrity. Please note that the former condition also holds for the material found on the web as everything on the web has been written by someone else.

References

[1] `https://www.kaggle.com/datasets/qusaybtoush1990/english-news`

[2] `https://www.kaggle.com/datasets/mirichoi0218/insurance`