Hieu Khuong - 915399644 - CIS4526

Programming Assignment 2

Part 1:

Find a public, free, supervised (i.e., it must have features and labels), machine learning dataset from somewhere *other than* The UCI Machine Learning Repository or Kaggle.com. Below is my dataset information:

1. The name of the data set: Yelp Business Dataset
2. Where the data can be obtained: https://yelp-dataset.s3.amazonaws.com/YDC13/yelp_dataset.tar.gz?Signature=vmTlPKoV7tx4lVWDyMcxkrbHIIg%3D&Expires=1572236632&AWSAccessKeyId=AKIAJ3CYHOIAD6T2PGKA

   (you will need to provide some basic information through https://www.yelp.com/dataset/download in order to download the dataset)

3. A brief (i.e. 1-2 sentences) description of the data set including what the features are and what is being predicted:

   The dataset file is composed of a single object type, one JSON-object per-line. The features are the features of the businesses, such as business_id, name, address, rating, categories, attributes (AcceptTakeout, parking, wifiAvailable, etc). The feature being predicted is the rating of the business, which ranges from 1 star to 5 stars.

   This rating can be predicted using the other features, for example: a restaurant with attributes like wifi and parking would usually have better rating than one without those utilities.

4. The number of examples in the data set: 163773 data points of business in the dataset.
5. The number of features for each example. If this isn't concrete, describe it as best as possible:

   Each datapoint has 21 features.

   These features include:

   Cities, states, stars rating, review_count, alcohol, bike_parking, accepts_credit_cards, caters, drivethru, goodforkids, hastv, noise_level, outdoor_seating, restuarants_price_range, delivery, goodforgroups, pricerange, reservations, table_service, takeout, wifi

Part 2:

Question 1: For each dataset, your program should output multiple training error rates, one for each feature. For each of the features, calculate the training error rate if you use only that feature to classify the data.

For the titanic_train.csv dataset:

- Error rate of feature: First_class : 0.3249299719887955 = 32.49%
- Error rate of feature: Sex : 0.21988795518207283 = 21.98%
- Error rate of feature: Age : 0.4061624649859944 = 40.61%
- Error rate of feature: SibSp : 0.4061624649859944 = 40.61%
- Error rate of feature: ParCh : 0.3851540616246498 = 38.51%
- Error rate of feature: Embarked : 0.3837535014005602 = 38.37%

For the breast cancer.csv dataset:

- Error rate of feature: age : 0.2972027972027972 = 29.72%
- Error rate of feature: menopause : 0.2972027972027972 = 29.72%
- Error rate of feature: tumor-size : 0.2972027972027972 = 29.72%
- Error rate of feature: inv-nodes : 0.2727272727272727 = 27.27%
- Error rate of feature: node-caps : 0.27622377622377625 = 27.62%
- Error rate of feature: deg-malig : 0.2797202797202797 = 27.97%
- Error rate of feature: breast : 0.2972027972027972 = 29.72%
- Error rate of feature: breast-quad : 0.29370629370629375 = 29.37%
- Error rate of feature: irradiat : 0.2972027972027972 = 29.72%

Question 2: For each dataset, use some library (e.g., sklearn) to build a full decision tree and report the training error rate. (One might use an existing implementation of the decision tree model for this question.)

For these 2 dataset, I used the decision tree implementation from sklearn, and I iterate the number of max_depth from 1 to 20 to see the training error improvement, and the criterion is entropy:

For the titanic_train.csv dataset:

- Max depth 1 : Error rate 0.21988795518207283
- Max depth 2 : Error rate 0.21988795518207283
- Max depth 3 : Error rate 0.21988795518207283
- Max depth 4 : Error rate 0.2184873949579832
- Max depth 5 : Error rate 0.20868347338935578
- Max depth 6 : Error rate 0.20308123249299714
- Max depth 7 : Error rate 0.20308123249299714
- Max depth 8 : Error rate 0.20308123249299714

- Max depth 9 : Error rate 0.20308123249299714
- Max depth 10 : Error rate 0.20308123249299714
- Max depth 11 : Error rate 0.20308123249299714
- Max depth 12 : Error rate 0.20308123249299714
- Max depth 13 : Error rate 0.20308123249299714
- Max depth 14 : Error rate 0.20308123249299714
- Max depth 15 : Error rate 0.20308123249299714
- Max depth 16 : Error rate 0.20308123249299714
- Max depth 17 : Error rate 0.20308123249299714
- Max depth 18 : Error rate 0.20308123249299714
- Max depth 19 : Error rate 0.20308123249299714
- Max depth 20 : Error rate 0.20308123249299714

So we can see that the decision tree on breast_cancer dataset reached best accuracy at max_depth = 6, with error rate being 20.3%.

For the breast cancer.csv dataset:

- Max depth 1 : Error rate 0.2797202797202797
- Max depth 2 : Error rate 0.24125874125874125
- Max depth 3 : Error rate 0.22377622377622375
- Max depth 4 : Error rate 0.2097902097902098
- Max depth 5 : Error rate 0.18181818181818177
- Max depth 6 : Error rate 0.16083916083916083
- Max depth 7 : Error rate 0.1398601398601399
- Max depth 8 : Error rate 0.12937062937062938
- Max depth 9 : Error rate 0.1048951048951049
- Max depth 10 : Error rate 0.0874125874125874
- Max depth 11 : Error rate 0.07342657342657344
- Max depth 12 : Error rate 0.06643356643356646
- Max depth 13 : Error rate 0.04895104895104896
- Max depth 14 : Error rate 0.038461538461538436
- Max depth 15 : Error rate 0.03146853146853146
- Max depth 16 : Error rate 0.020979020979020935
- Max depth 17 : Error rate 0.020979020979020935
- Max depth 18 : Error rate 0.020979020979020935
- Max depth 19 : Error rate 0.020979020979020935
- Max depth 20 : Error rate 0.020979020979020935

So we can see that the decision tree reached best accuracy at max_depth = 16, with error rate being 2.09%.

Question 3: Can we directly use the kNN or the Perceptron model to train a classifier on these two datasets? A brief and reasonable explanation would be good enough. (Hint: compared to

the dataset in PA 1, is there any trouble on how to compute the distances or how to compute the inner-products? Optionally, one might come up with some ideas so that these issues are resolved.)

I think we can use kNN or Perceptron to train a classifier for these dataset. For example if we want to do kNN we would need to compute the distances, then the problem we are having is that these datasets' features are not numbers, so we need to convert those features into numbers using some kind of label encoder. Then these features need to be normalized and regularized to prevent bias. The same applies with perceptron training: we need to encode the features to make vectors consisting of numbers, which would later be used in matrix/vector multiplication procedure in perceptron.