CSE 158/258, Fall 2021: Homework 1

Instructions

Please submit your solution **by the beginning of the week 3 lecture (Oct 11).** Submissions should be made on **gradescope**. Please complete homework **individually**.

This specification includes both questions from the undergraduate (CSE158) and graduate (CSE258) classes. You are welcome to attempt questions from both classes but will only be graded on those for the class in which you are enrolled.

You will need the following files:

GoodReads Fantasy Reviews : `https://cseweb.ucsd.edu/classes/fa21/cse258-b/data/fantasy_10000.json.gz`

Beer Reviews : `https://cseweb.ucsd.edu/classes/fa21/cse258-b/data/beer_50000.json` The above is a *json* formatted dataset. Data can be read using the *json.loads* function in Python, or by using *eval*.

Code examples : `http://cseweb.ucsd.edu/classes/fa21/cse258-b/code/week1.py` (regression) and `http://cseweb.ucsd.edu/classes/fa21/cse258-b/code/week2.py` (classification)

Executing the code requires a working install of Python 2.7 or Python 3 with the scipy packages installed. **Please include the code of (the important parts of) your solutions.**

Tasks — Regression (week 1):

First, using the *book review* data, let's see whether ratings can be predicted as a function of review length, or by using temporal features associated with a review.

1. **(CSE158 only)** What is the distribution of ratings and review lengths in the dataset? Report the number of 1-, 2-, 3-star (etc.) ratings, and show the relationship with length (e.g. via a scatterplot) (1 mark).

2. Train a simple predictor that estimates rating from review length, i.e.,

$$\text{star rating} \simeq \theta_0 + \theta_1 \times [\text{review length in characters}]$$

Report the values θ_0 and θ_1, and the Mean Squared Error of your predictor (on the entire dataset) (1 mark).

3. Extend your model to include (in addition to the length) features based on the time of the review. You can parse the time data as follows:

```
> import dateutil.parser
> t = dateutil.parser.parse(d['date_added'])
> t.weekday(), t.year # etc.
```

Using a *one-hot encoding* for the weekday and year, write down feature vectors for the first two examples (1 mark).[1]

4. Train models that

   - use the weekday and year values directly as features, i.e.,

$$\text{star rating} \simeq \theta_0 + \theta_1 \times [\text{review length in characters}] + \theta_2 \times [\text{t.weekday()}] + \theta_3 \times [\text{t.year}]$$

   - use the one-hot encoding from Question 3.

Report the MSE of each (1 mark).

[1] Hint: your one-hot encoding of the year should not have 2021 dimensions! Also be careful not to include any redundant dimensions, e.g. your one hot encoding for the week should have only six dimensions.

5. Repeat the above question, but this time split the data into a training and test set. You should split the data randomly into 50%/50% train/test fractions. Report the MSE of each model separately on the training and test sets.

6. **(CSE258 only)** Show that for a trivial predictor, i.e., $y = \theta_0$, the best possible value of θ_0 in terms of the Mean *Absolute* Error is the *median* (i.e., the average value of the label y). Hint: compute the derivative of the model's MSE and solve for θ_0

Tasks — Classification (week 2):

In this question, using the *beer review* data, we'll try to predict ratings (positive or negative) based on characteristics of beer reviews. Load the 50,000 beer review dataset, and construct a label vector by considering whether a review score is four or above, i.e.,

```
y = [d['review/overall'] >= 4 for d in dataset]
```

7. Fit a logistic regressor that estimates the binarized score from review length, i.e.,

$$p(\text{rating is positive}) \simeq \sigma(\theta_0 + \theta_1 \times [\text{length}])$$

Using the `class_weight='balanced'` option, report the True Positive, True Negative, False Positive, False Negative, and Balanced Error Rates of the predictor (1 mark).

8. Plot the precision@K of your classifier for $K = \{1 \dots 10000\}$ (i.e., the x-axis of your plot should be K, and the y-axis of your plot should be the precision@K) (1 mark).[2]

9. Our precision@K plot from Question 8 only measures precision with regard to the *positive* class. For this type of binary classification, we may be equally interested in the classifier's accuracy for both the positive and negative classes. Recompute confidence scores for your classifier so that the 'most confident' predictions include either the most confident positive *or* the most confident negative predictions (i.e., probability closest to 1 *or* probability closest to zero).[3] The precision@K now measures whether the classifier has the correct label (either 'positive' or 'negative') among the K most confident entries. Report this precision@K for $K \in \{1, 100, 10000\}$ and include a plot as in Question 8.

[2] You need not plot every value of K, and just plot at several increments. An efficient implementation should take less than one second to generate the plot.

[3] Could be computed via e.g. $|p(\text{positive}) - 0.5|$.