General guidelines:

- These homeworks are tentatively due on Monday of week 3, 5, 7, and 9. If the corresponding lecture material (module 1 for homework 1 etc.) has not been completed by the deadline, the deadline will be moved by one week. But the final deadlines are always **as posted on gradescope.**
- Welcome to start early, but homeworks should be considered in "draft form" until the submission page is active on gradescope
- **Each homework is worth 8 marks, and one HW is dropped, so that all HWs are worth 24 marks**
- Homeworks must be completed individually

Homework stubs and files:
https://drive.google.com/drive/folders/1f5xFMJYdjzh0cU-XLK3nL07flhPlANwK?usp=sharing
Solutions will be posted to the same folder.

The "runner" files exhibit similar behavior to what the autograder is doing, and should run without error once you've implemented your solution.

You should submit a single file containing your code (e.g. "homework1.py") to the autograder

Regression (week 1):

First, using the book review data (see the "runner" code for the exact dataset names), let's see whether ratings can be predicted as a function of review length, or by using temporal features associated with a review.

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

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

   Rather than using the review length directly, scale the feature to be between 0 and 1 by dividing by the maximum review length in the dataset. Return the value of θ and the Mean Squared Error of your predictor (on the entire dataset).
2. Extend your model to include (in addition to the scaled length) features based on the time of the review. The runner contains code to compute the weekday. Using a one-hot encoding for the weekday and month, write down feature vectors for the first two examples. Be careful not to include any redundant dimensions: e.g. your feature vector, including the offset term and the length feature, should contain no more than 19 dimensions.
3. Train models that
   a. use the weekday and month values directly as features, i.e.,
      $\text{star rating} \approx \theta_0 + \theta_1 \times [\text{review len in chars}] + \theta_2 \times [\text{t.weekday()}] + \theta_3 \times [\text{t.month}]$
   b. use the one-hot encoding from Question 2.

Return the MSE of each.

4. Repeat the above question, but this time split the data into a training and test set. You should split the data into 50%/50% train/test fractions **following the split used by the code stub (or runner).** After training on the training set, compute the MSE of the two models (the one-hot encoding from Question 2 and the direct encoding from Question 3) on the test set.

Classification (week 2):

Next, 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 (done in the runner), 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]
```

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

   $$p(\text{rating is positive}) \approx \sigma(\theta 0 + \theta 1 \times [\text{length}])$$

   Using the class weight='balanced' option, compute the number of True Positives, True Negatives, False Positives, False Negatives, and the Balanced Error Rate of the classifier.
6. Compute the precision@K of your classifier for $K \in \{1, 100, 1000, 10000\}$.
7. Improve your predictor (specifically, reduce the balanced error rate) by incorporating additional features from the data (e.g. beer styles, ratings, features from text, etc.). The BER should be ~3% higher than the solution from Q5.