General guidelines:

- These homeworks are tentatively due at the end of weeks 2, 4, 6, 8, 10. But the final deadlines are **as posted on gradescope**
- Welcome to start early, but homeworks **should be considered to be in "draft form"** until the corresponding stubs are posted in the link below
- **Each homework is worth 10% of the final grade, so that all HWs are worth 50%**
- Homeworks must be completed individually
- Homework stubs are available here

json datasets can generally be read as follows:

```
z = gzip.open("path.json.gz")
dataset = []
for l in z:
    d = eval(l)
    dataset.append(d)
```

for other datasets, follow instructions from the respective dataset pages and stubs.

Homework 1: Regression and classification

Regression: download the steam dataset:
https://cseweb.ucsd.edu/classes/fa24/cse258-b/files/steam.json.gz

Experiments in this section will only use the "time played" (`d["hours"]`) and the length of the review text (`len(d["text"])`).

1. Implement regression on time played (hours) using the length of the review and a bias term. Use an 80%/20% training/test split (split using the first 80% and last 20%, i.e., do not shuffle the data). Use sklearn's `linear_model.LinearRegression`. *Report the MSE on the test set.*

Implement the following interventions (independently of each other):

2. Delete outliers from the training set. Keep the bottom 90% of time played values. *Report the MSE on the test set, as well as the MAE on the test set.*
3. Transform the target variable by taking log_2(y + 1). After performing regression on the test set, invert the transform (so that your model still predicts a number of hours), and *report the MSE on the test set.*

4. Build a regressor that optimizes the MAE rather than the MSE (see e.g. https://stackoverflow.com/questions/71534025/how-mae-loss-is-optimized-with-sgd-optimizer-in-sklearn). *Report the MSE and the MAE on the test set.*

Classification: download the German Credit dataset:
https://archive.ics.uci.edu/dataset/144/statlog+german+credit+data
(use the "numeric" version of the data)

5. Implement a logistic regressor (sklearn's `linear_model.LogisticRegression`) to predict the target variable (credit = good). Implement a 80/20 training/test split. Use a regularization value of C = 1.[1] *Report the accuracy and TPR on the test set.*

For the following, treat the 20th feature (i.e., feat[19] in the numeric data, which is related to housing) as the "sensitive attribute" i.e., z=1 if the feature value is 1.

6. *Report the TPR for Z=0 and Z=1.*
7. By changing the classifier thresholds, generate an ROC curve for your classifier. *Report the TPR when the FPR is (as close as possible to) 0.8.*
8. Use an instance weighting approach (this option is already available in the library implementation) to tune the model such that instances with z=0 have 5x the instance weight of instances with z=1. *Report the TPR for Z=0 and Z=1.*

9. (harder, 2 marks) Implement a support vector machine (using sklearn). Using your solution from question Q5, generate ROC curves for both the logistic regression model and the support vector machine. Find the point[2] on this curve where the two models intersect, and *report the corresponding TPR/FPR values.*

Solutions (answers_hw1.txt) should take the form:

```
{
    "Q1": float,          # MSE
    "Q2": [float, float], # MSE, MAE
    "Q3": float,          # MSE
    "Q4": [float, float], # MSE, MAE
    "Q5", [float, float], # Accuracy, TPR
    "Q6": [float, float], # TPR_0, TPR_1
    "Q7": float,          # TPR
    "Q8": [float, float], # TPR_0, TPR_1
    "Q9": [float, float]  # TPR, FPR
}
```

[1] A few people reported they needed to increase LogisticRegression's "max_iter" to 1000 before getting the right solutions to some of these problems
[2] If there are multiple such points, you may return any (non-trivial) point

Homework 2: Intro to bias and fairness

Download the German Credit dataset:
https://archive.ics.uci.edu/dataset/144/statlog+german+credit+data
(use the “numeric” version of the data)

Implement a (logistic regression) classification pipeline using an 80/20 test split. Use a regularization value of C = 1.

Treat “foreign worker” as the “sensitive attribute” i.e., z=1 for foreign workers and z=0 for others.

1. *Report the prevalence in the test set.*
2. *Report the per-group prevalence for z=0 and z=1.*
3. *What is the demographic parity (expressed as a ratio between z=0 and z=1) for your classifier on the test set?*
4. *Report TPR_0, TPR_1, FPR_0, and FPR_1 (see "equal opportunity" slides).*
5. *Compute PPV_0, PPV_1, NPV_0, and NPV_1 (see "are fairness goals compatible" slides).*
6. Implement a "fairness through unawareness" classifier, i.e., don't use Z in your feature vector. Find the classifier coefficient which undergoes the largest (absolute value) change compared to the classifier with the feature included, *and report its new coefficient.*
7. *Report the demographic parity of the classifier after implementing the above intervention.*
8. *Report the Generalized False Positive Rate and Generalized False Negative Rate of your original (i.e., not the one with z excluded).*
9. (harder, 2 marks) Changing the classifier threshold (much as you would to generate an ROC curve) will change the False Positive and False Negative rates for both groups (i.e., FP_0, FP_1, FN_0, FN_1). Find the (non-trivial) threshold that comes closest to achieving Treatment Equality, and report the corresponding values of FP_0, FP_1, FN_0, and FN_1.

Solutions (answers_hw2.txt) should take the form:

```
{
    “Q1”: float,           # prevalence
    “Q2”: [float, float],  # prevalence_0, prevalence_1
    “Q3”: float,           # parity
    “Q4”: [float, float, float, float], # TPR_0, TPR_1, FPR_0, FPR_1
    “Q5”, [float, float, float, float], # PPV_0, PPV_1, NPV_0, NPV_1
    “Q6”: [string, float], # feature name, coefficient
    “Q7”: float,           # parity
    “Q8”: [float, float],  # GFPR, GFNR
    “Q9”: [float, float, float, float]  # FP_0, FP_1, FN_0, FN_1
}
```

Homework 3: Fairness and bias interventions

Classification: Download the “adult” dataset:
https://archive.ics.uci.edu/dataset/2/adult

Implement a (logistic regression) classification pipeline using an 80/20 test split. Use a regularization value of C = 1.

Treat “sex” as the “sensitive attribute” i.e., z=1 for females and z=0 for others.

1. *Report the "discrimination in the dataset"* (see "pre-processing" section).
2. *Report the discrimination of the classifier.*
3. (2 marks) Implement a "massaging" approach that improves the discrimination score by at least 3%; *report the new discrimination score.* (Note: the interventions in questions 3-5 should be separate from each other)
4. (2 marks) Implement a "reweighting" approach that improves the discrimination score by at least 3%; *report the new discrimination score.*
5. (2 marks) Implement a "post processing" (affirmative action) policy. Lowering per-group thresholds will increase the (per-group) TPR and FPR. For whichever group has the lower per-group TPR, lower the threshold until the TPR for both groups is (as close as possible to) equal. *Report the rates (TPR_0, TPR_1, FPR_0, and FPR_1) for both groups.*
6. Modify the solution from Q5 to exclude the sensitive attribute (z) from the classifier’s feature vector. Again implementing the same strategy as in Q5, *report the rates (TPR_0, TPR_1, FPR_0, and FPR_1) for both groups.*
7. Modify the solution from Q5 by training two separate classifiers, one for z=0 and one for z=1. Again implementing the same strategy as in Q5, *report the rates (TPR_0, TPR_1, FPR_0, and FPR_1) for both groups.*

Solutions (answers_hw3.txt) should take the form:

```
{
    “Q1”: float, # discrimination
    “Q2”: float, # discrimination
    “Q3”: float, # discrimination
    “Q4”: float, # discrimination
    “Q5”, [float, float, float, float], # TPR_0, TPR_1, FPR_0, FPR_1
    “Q6”: [float, float, float, float], # TPR_0, TPR_1, FPR_0, FPR_1
    “Q7”: [float, float, float, float]  # TPR_0, TPR_1, FPR_0, FPR_1
}
```

Homework 4: Fairness and bias interventions

Regression: Download the “wine quality” dataset:
https://archive.ics.uci.edu/dataset/186/wine+quality

Implement a linear regressor using all continuous attributes (i.e., everything except color) to predict the wine quality. Use an 80/20 train/test split. Use sklearn’s linear_model.LinearRegression

1. *Report the feature with the largest coefficient value and the corresponding coefficient (not including any offset term).*
2. On the first example in the test set, *determine which feature has the largest effect and report its effect* (see "Explaining predictions using weight plots & effect plots").
3. (2 marks) Based on the MSE, compute ablations of the model including every feature (other than the offset). Find the most important feature (i.e., such that the ablated model has the highest MSE) and *report the value of MSE_ablated - MSE_full.*
4. (2 marks) Implement a full backward selection pipeline and *report the sequence of MSE values for each model as a list* (of increasing MSEs).
5. (2 marks) Change your model to use an l1 regularizer. Increasing the regularization strength will cause variables to gradually be removed (coefficient reduced to zero) from the model. Which is the first and the last variable to be eliminated via this process?

Implement a classifier to predict the wine color (red / white), again using an 80/20 train/test split, and including only continuous variables.

6. *Report the odds ratio associated with the first sample in the test set.*
7. Find the 50 nearest neighbors (in the training set) to the first datapoint in the test set, based on the l2 distance. Train a classifier using only those 50 points, and *report the largest value of e^theta_j* (see “odds ratio” slides).

Solutions (answers_hw4.txt) should take the form:

```
{
    “Q1”: [string, float],  # feature name, coefficient
    “Q2”: [string, float],  # feature name, coefficient
    “Q3”: [string, float],  # feature name, difference
    “Q4”: list, # increasing MSEs, same length as feature vector
    “Q5”, [string, string], # feature names
    “Q6”: float,            # odds ratio
    “Q7”: float             # e^theta_j
}
```

Homework 5: Fairness and bias in application domains

Use code from https://cseweb.ucsd.edu/~jmcauley/pml/ (Chapter 4) to build a simple recommender based on beer review data. You can use the "mostSimilarFast" function as your recommender, though will have to modify the data loader a little bit to use the beer dataset. You may use the 50,000 review dataset available here: https://datarepo.eng.ucsd.edu/mcauley_group/pml_data/beer_50000.json

You may also use code from the above link for simple utilities (e.g. Gini coefficient)

1. (2 marks) Measure the gini coefficient of the recommender. Generate recommendations (N=10) for the first 100 items, and *report the Gini coefficient in terms of item popularity* (i.e., number of times each item appears in the dataset).
2. (2 marks) Measure the calibration of the above recommendations, i.e., report the frequency of each category in the training set, compared to the frequency with which each category is recommended. *Report results for the 5 most common categories in the data.*
3. (2 marks) Implement a Maximum Marginal Relevance pipeline using a similarity function that penalizes beers with similar ABVs. Your relevance function should still be the Jaccard similarity; your similarity function should be || ABV_a - ABV_b ||^2; use a value of lambda = 0.5. Generate N=10 recommendations for the first item. *Report the recommendations and their associated scores.*
4. (4 marks) For this question, we'll measure "relevance" by considering average rating ("overall") for each item. Using your model from Q3, generate N=10 recommendations for the first 100 items. Report the average relevance of all recommendations for lambda \in [1, 0.8, 0,6, 0.4]

Solutions (answers_hw5.txt) should take the form:

```
{
    "Q1": float, # gini coefficient
    "Q2": list, # 5 floats from most common to 5th most common
    "Q3": [list, list], # list of 10 items, list of 10 scores
    "Q4": list # 4 floats
}
```