Homework 2

1. *Standardizing Boolean features.* Consider a raw input u that is Boolean, *i.e.*, $u^i \in \{0,1\}^m$. We define U as the data matrix

$$U = \begin{bmatrix} (u^1)^T \\ (u^2)^T \\ \vdots \\ (u^n)^T \end{bmatrix}.$$

   (a) Let p_j be the mean of the jth feature over all data points, so p is the m-vector of feature means. Explain how to compute p_j from U. Briefly explain what p_j is, in plain English.

   (b) Now consider the standard deviation of each feature, denoted σ_j, $j = 1, \ldots, m$. Explain how to express σ_j in terms of p_j.

   (c) Let $x = \phi(u)$ be the feature mapping that standardizes u, with $X \in \mathbf{R}^{n \times m}$ the associated data matrix. Show how to find X.

   (d) Give a short Julia function that carries out the standarization. The function takes as input U and produces X.

2. *k-nearest neighbor weights.* Let $g^{\rm snn}$ be the soft nearest neighbor predictor associated with the data set $x^1, \ldots, x^n$, $y^1, \ldots, y^n$. For each $x \in \mathbf{R}^d$, $g^{\rm snn}(x)$ is a *weighted average* of the outcomes, *i.e.*,

$$g^{\rm snn}(x) = \sum_{i=1}^n w_i y^i,$$

   where w_i are set of nonnegative weights that add up to one. The particular weights depend on the distances to the data feature vectors, $\delta_i = \|x - x^i\|_2$, $i = 1, \ldots, n$.

   Now let $g^{k\rm nn}$ be the k-nearest neighbor predictor associated with the same data set. Show that $g^{k\rm nn}(x)$ is also a weighted average of the $y^i, i = 1, \ldots, n$, where the weights are a (different) function of the distances. Describe what these weights are, as a function of the distances $\delta_1, \ldots, \delta_n$. You may assume that the distances are unique, *i.e.*, no two are equal.

   Hint. The weights are not given by a formula, as it is in soft nearest neighbor predictor; they are most easily described using cases (that depend on the distances).

3. *Feature engineering for nearest neighbor predictors.* Some common feature engineering transforms have no effect for some types of predictors. Here we examine one of these.

   It is very common to add a first feature that always has the value 1. This is done with the feature engineering transform $\mathcal{T}(x) = (1, x)$. How does this feature engineering mapping affect a k-nearest neighbor or soft nearest neighbor predictor? Justify your answer.

   Hint. How are $\|x - x^i\|_2$ and $\|\mathcal{T}(x) - \mathcal{T}(x^i)\|_2$ related?

4. *Faithful embedding of hours of the year.* Consider a raw feature that gives the date and hour, *e.g.*, July 13 4AM. There are $365 \times 24 = 8760$ possible values of this categorical. We express this as $u = (d, h)$, where $d = 1, \ldots, 365$ is the day of the year, and $h = 1, \ldots, 24$ is the hour of the day. (We can also express the date and hour using the conventional notation.) We consider a given date/hour to be similar to its neighbors one hour ahead and one hour behind, as well as the date/hour exactly 24 hours earlier or later. Thus July 13 4AM is similar to July 13 3AM, July 13 5AM, July 12 4AM, and July 14 4AM. (For example, you might expect the temperature of similar hours to be close.) In terms of u, (d, h) is similar to $(d, h-1)$, $(d, h+1)$, $(d-1, h)$ and $(d+1, h)$, where the index arithmetic is circular, *i.e.*, modulo 365 for d and 24 for h. This means for example when $d = 365$ (December 31), we take $d+1$ to be 1 (January 1), and when $h = 1$ (1AM), we take $h - 1$ to be 24 (midnight).

   Suggest a faithful embedding for date/hour values. You can describe it informally or just draw it; you do not need to come up with any kind of formula. You might want to use the term 'torus' (which you should feel free to look up). *Hint.* You can get a good embedding in $\mathbf{R}^3$.

5. *Validating different feature mappings and different predictors.* In this exercise you will carry out a very common activity in machine learning, comparing two different feature mappings, and 4 different predictors, using validation.

   In `feat_valid.json`, you will find a 300×3 matrix `U` of raw input data, with rows $(u^i)^T$, with $u^i \in \mathbf{R}^3, i = 1, \ldots, 100$, and a 300-vector `v` with the raw output data. We will work with $y = \psi(v) = v$. Partition the data into a training set and a validation set by randomly assigning 80% of the data set into training and the remaining 20% into a validation set.

   Julia hint. The function `shuffle(x)` in the `Random` package can be used to randomly shuffle an array `x`.

   (a) Using the feature mapping $x = \phi(u) = (u_1, u_2, u_3)$ to fit a soft nearest neighbor model for $\rho = 0.25, 0.5, 1, 2$. Report the train and test RMS errors for each of these 4 predictors.

   (b) Do the same for the feature mapping

   $$\phi(u) = (u_1,\ u_2,\ u_3,\ u_1u_2,\ u_2u_3,\ u_1u_3,\ u_1^2,\ u_2^2,\ u_3^2).$$

   Julia hint. You can implement this feature map in Julia using

   ```
   phi(U) = [U U[:,1].*U[:,2] U[:,2].*U[:,3] U[:,1].*U[:,3] U.^2].
   ```

   (c) Which combination of feature mapping and soft nearest neighbor predictor parameter performs best, on the test set?