Homework 2

1. Nearest neighbor predictors and Voronoi sets. Nearest neighbor, \( k \)-nearest neighbor, and tree predictors are piecewise constant functions. This means that we can partition \( \mathbb{R}^d \) into \( N \) regions, \( \mathcal{R}_1, \ldots, \mathcal{R}_N \), and we have \( g(x) = \hat{y}_k \) for all \( x \in \mathcal{R}_k \). The regions don’t overlap, except for their boundaries, and every point in \( \mathbb{R}^d \) is in one of the regions. (We are not concerned with how these predictors work when a point is right on the boundary between two regions; in any case, whichever value the predictor takes on these boundaries makes no difference at all in practice.) In this problem we explore this idea.

(a) The set of points closer to one given point than another. Suppose \( u \) and \( v \) are given vectors in \( \mathbb{R}^d \), with \( u \neq v \). We define the set of all points in \( \mathbb{R}^d \) that are closer to \( u \) than \( v \) as \( S(u, v) = \{ x \in \mathbb{R}^d \mid \|x - u\|_2 \leq \|x - v\|_2 \} \).

Show that \( S(u, v) \) is a halfspace, which has the form \( S(u, v) = \{ x \in \mathbb{R}^d \mid a^T x \leq b \} \).

(You should say explicitly what the vector \( a \) is, and what the scalar \( b \) is.) The boundary of \( S(u, v) \) is a hyperplane, i.e., the set of points that satisfy \( a^T x = b \).

Sketch this for the specific example with \( u = (1, 1) \) and \( v = (2, 4) \). Show the points \( u \) and \( v \), shade the set \( S(u, v) \), and indicate its boundary, which is a line (since a hyperplane in \( \mathbb{R}^2 \) is a line).

(b) Voronoi sets. Suppose \( u^1, \ldots, u^m \) are given points in \( \mathbb{R}^d \). Define \( \mathcal{V}_i \), the set of points in \( \mathbb{R}^d \) closer to \( x^i \) than the others (i.e., \( u^i \) is the nearest neighbor of all points in \( \mathcal{V}_i \)), for \( i = 1, \ldots, m \). We can express these sets as

\[
\mathcal{V}_i = \bigcap_{j \neq i} S(u^i, u^j),
\]

\( i.e., \mathcal{V}_i \) is the intersection of the \( m - 1 \) halfspaces \( S(u^i, u^j) \) for \( j \neq i \). An intersection of halfspaces is called a polyhedron. The specific polyhedra \( \mathcal{V}_i \) are called Voronoi sets or Voronoi regions associated with the collection of points \( u^1, \ldots, u^m \). (Polyhedra is the plural of polyhedron.) They form a partition of \( \mathbb{R}^d \) into a set of polyhedra, called the Voronoi partition of \( \mathbb{R}^d \).

Sketch the Voronoi regions for the collection of points \( (1, 1) \), \( (2, 4) \), \( (5, 3) \).

(c) Nearest neighbor predictor. Let \( g \) be the 1-nearest neighbor predictor for the data set \( x^1, \ldots, x^n, y^1, \ldots, y^n \). Explain why \( g \) has the form \( g(x) = \hat{y}_k \) for \( x \in \mathcal{V}_k \), where \( \mathcal{V}_k \) is the Voronoi region associated with \( x^k \). In other words, \( g \) is piecewise constant, with the same value inside each of the Voronoi regions associated with the vectors \( x^i \), \( i = 1, \ldots, n \).

(d) \( k \)-nearest neighbor predictor. Let \( g(x) \) be the \( k \)-nearest neighbor predictor for the data set \( x^1, \ldots, x^n, y^1, \ldots, y^n \). Explain why \( g \) is piecewise constant on a regions that are polyhedra. You can do this for the case \( k = 2 \), to make things simpler.
**Hint.** The set of points $x$ for which $x^q$ and $x^l$ are its two nearest neighbors has the form

$$
R_{ql} = \{ x \in \mathbb{R}^d \mid \|x - x^q\|_2 \leq \|x - x^j\|_2, \|x - x^l\|_2 \leq \|x - x^j\|_2, \ j \neq q \text{ or } l \}.
$$

This is an intersection of $2(n-1)$ halfspaces.

**Hint.** Drawing these sets in in 2D is encouraged. You can use such drawings as part of your explanation.

2. **All-pairs interactions.** The following problem will use $U$, and $v$ found in all_pairs_data.json. The data has $U \in \mathbb{R}^{n \times 3}$. Throughout this problem, use a 50-50 train/test split.

(a) Fit a linear least-squares model directly to the data matrix, with the first feature being a constant feature $x_1 = 1$. Since we’ve given you enough data and the data is approximately standardized, you do not have to worry about regularization or standardization.

The linear least-squares model is:

$$
\sum_{i=1}^{N} (\theta^T u^{(i)} - v^{(i)})^2
$$

If we define $v$ as the vector of $v^{(i)}$’s and $U$ as a matrix of $u^{(i)}$’s (every row of $U$ is one observation $u^{(i)}$), we can rewrite this in matrix form:

$$
\|U\theta - v\|_2^2
$$

Report the train and test RMSE of this predictor.

**Hint:** In Julia, you can compute the least-squares solution as $\text{theta} = U \backslash v$.

**Hint:** In this case, it is fine to split the data in half naively, using the first half as the train dataset and the second half as the test dataset (or vice-versa).

(b) Create an embedding which includes all of the interactions (products) between every pair of distinct variables, along with a constant feature and the variables themselves. For example, if $u \in \mathbb{R}^2$, then the embedding should be

$$
\phi(u) = (1, u_1, u_2, u_1u_2).
$$

You’ll have to construct a new data matrix. This data matrix should include a column of ones and $d$ columns for $u_1, \ldots, u_d$. The remaining columns will contain the products $u_1^2$, $u_1u_2, \ldots, u_3u_2$, $u_3^2$.

Fit a new $\theta$ to this data matrix. Report the train and test RMSE of this predictor. Compare it with the RMSEs you got in (a).

(c) Is it necessary to use all of the features? Print the $\theta$ you found in part (b) and inspect it. Remember $\theta$ represents the weights assigned to each feature. Are there any features that aren’t being used in the predictor? Remove them. (You’ll have to construct a smaller data matrix - see the hint.)
Report which indices \( \theta_i \) you removed. Remember that in Julia, vectors are 1-indexed, so \( \theta = [\theta_1, \theta_2, \ldots \theta_{13}]^\top \).

Fit a new \( \theta \) with the unnecessary features removed and report the train and test RMSE of this predictor. Compare it with the RMSEs you got in (a) and (b).

**Hint:** To remove column \( i \) from a matrix \( U \) in Julia, you can use array slicing:

\[ U = [U[:,1:i-1] U[:,i+1:end]] \]

Be careful to slice the columns (second index) and not the rows. In the above line, we use : to index all the rows.

**Hint:** It is perfectly fine to hard-code the array slicing: if you decide to remove index 9, you may hard-code the value 9.

**Hint:** You should remove at least 3 features. You may remove more if you like.

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

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

(b) *Splitting into positive and negative parts.* Consider the feature engineering transform \( T(x) = ((x)_+, (x)_-) \), where \( (x)_+ = \max\{x, 0\} \) and \( (x)_- = \min\{x, 0\} \), both of these acting elementwise, i.e., separately on each entry. 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 \( \|T(x) - T(x^i)\|_2 \) related?