1. The different types of machine learning problems. Determine whether the tasks described below involve supervised learning or unsupervised learning. For supervised learning problems, identify them as regression, classification, or probabilistic classification.

   (a) Predict the risk of an accident at an intersection, given features such as the time of day and weather.
   (b) Identify cars, bicyclists, and pedestrians in video taken by an autonomous vehicle’s cameras.
   (c) Determine the probability that there is a stop sign in an image.
   (d) Generate new road scenarios (generate streets, place stop signs and intersections) for testing autonomous vehicles in a simulation.

2. Train vs test datasets. Suppose you are building a classifier that identifies cats and dogs. You have a dataset of 3,000 images containing cats, dogs, or other objects (neither cat nor dog). You randomly split the data into a 2,500 image training set and a 500 image test set.

   (a) Why is it important to “reserve” some images for the test dataset? (Why shouldn’t we use all 3,000 images to train the classifier?)
   (b) After training your classifier for a while, you observe it performs well on the training images, but poorly on the test images. What is one possible explanation?
   (c) Suppose we have a classifier with good performance on the training set but high error on the test data. For each of the following methods, briefly explain if they can solve this issue.
      i. Increasing the size of the test set from 500 to 1,000, while the total number of images remains the same.
      ii. Adding 2,000 more images to the dataset, training over a set of 4,000 images and testing with the other 1,000 images.
      iii. Using a more complex feature mapping to generate features from raw data.

3. Some true or false questions. For each of the statements below, state whether it is true or false. If true, give a brief explanation why. If false, give a counterexample.

   Note that the nearest neighbor predictors in this problem are meant for regression (not classification).

   (a) Adding a single data point to our dataset can change the k-NN predictor output for every possible input.
(b) A soft nearest neighbor predictor $g$ with the parameter $\rho \to \infty$ becomes a constant function, i.e., $g(x)$ will be independent of $x$.

(c) Let $g$ be a $k$-NN predictor trained over a data set. If $(x,y)$ is a data point from the training set, then $y = g(x)$.

(d) The computation time needed to predict the output of the $k$-NN predictor for a new sample depends on the size of data set, making this algorithm computationally expensive for large data sets.

4. *Fitting a known function using samples.* In this problem you will use various nearest neighbor methods to predict $y \in \mathbf{R}$ given $x \in \mathbf{R}$, for a simple case in which we know the exact relation between $x$ and $y$. (This is never the case in practical prediction problems.)

Consider the function $f(x) = \sin(10x)$ over $x \in [0,1]$.

(a) Randomly sample 30 points $x^i$ from $[0,1]$ using a uniform distribution, and let $y^i = f(x^i)$. Plot these data points as dots, along with $f$ as a curve. (To plot $f$, evaluate it for 500 points uniformly spaced in $[0,1]$, i.e., $x = (k - 1)/499$, $k = 1, \ldots, 500$.)

(b) On eight separate plots, plot the $k$-nearest neighbor predictors for $k = 1, 2, 3$ and the soft nearest neighbor predictors for $\rho = 0.001, 0.003, 0.01, 0.03$. Include the 30 data points, shown as dots, in these plots.

(c) *RMS error.* For each of the eight predictor functions in part (b), evaluate the RMS error on the 500 uniformly spaced points used to plot the functions, given by

\[
\left( \frac{1}{500} \sum_{k=1}^{500} (\hat{y}_k - y_k)^2 \right)^{1/2},
\]

with $y_k = f((k - 1)/499)$ and $\hat{y}_k = g((k - 1)/499)$, where $g$ is your predictor.

*Julia hints.*

- `rand(N)` generates $N$ points from a uniform distribution on $[0,1]$.
- To generate a uniformly spaced set of $N$ values between $a$ and $b$ (with $a < b$), use `range(a, stop=b, length=N)`.
- To apply a function $f : \mathbf{R} \to \mathbf{R}$ elementwise to a vector $x$, use `f.(x)`. 