Supervised

1. Decision Stump
   fit: $O(nd\log n)$
   `PSEUDO CODE HERE with O(n) on steps`
2. Decision Tree
   fit: $O(mnd\log n)$ m depth `PSEUDO CODE HERE with O(n) on steps`
3. Naive Bayes
   Bayes Rule: $p(y_i|x_i) = \frac{p(x_i|y_i)\cdot p(y_i)}{p(x_i)}$
   runtime fit: $O(nd)$ go thru X and count

4. KNN
   - Non-Parametric
   - Prediction Cost: $O(n^2(d+k))$
   - depends on norm
   - Usage:

   for each x_i $O(n)$,
   find the distance $O(d)$
   to each example $O(n)$.
   compute the average of the closest k $O(k)$
5. Linear Regression
6. Non-linear Regression (Supervised)

Unsupervised

1. K-Means
   - Parametric (k, W weights)
   - fit cost: store $O(ndk)$
   - update: $O(ndk)$
   - depends on initialization
   - Usage: convex clustering, vector quantization, Outlier Detection

   for each x_i $O(n)$, find the distance $O(d)$ to k centers $O(k)$;
   assign x_i to closest center;
   update means (centers) $O(ndk)$;
   repeat this until the closest center does not change $O(T)$
2. DBSCAN
   - Non-Parametric (ϵ, MinNbr)
   - problem: different densities

   for each x_i $O(n)$,
   if already assigned, do nothing
   check if $\geq MinNbrs$ within ϵ
   if not, do nothing, if so 'expand' cluster.
   To expand,
   assign all ϵ nbrs to this cluster.
   repeat for each added points.
3. Outlier Detection Methods
   - Model-based methods
   - Graphical approaches (scatter plot)
   - Cluster-based methods (k-means DBSCAN)
   - Distance-based methods (KNN)
   - Supervised-learning methods
4. Recommender?

Something

1. Golden Rule:
   Test data must not influence in the training phase in any way.
2. Fundamental Trade Off:
   $E_{test} = (E_{test} - E_{train}) + E_{train}$
   E_{appx} gets smaller as n larger, grows as model gets complicated
3. Random Forest
   - Parametric (trees, depth of each)
   - fit cost: store $O(ndk)$
   - update: $O(ndk)$
4. Cross-Validation
   runtime (fit): k x runtime of the fit
   runtime (predict): k x runtime of the predict
   for each k fold $O(k)$,
   train on the rest $O(((k-1)/k)n) \times$ fit time
   predict on the fold $O(n/k) \times$ predict time

5. Definitions
6. Linear Algebra Notes
   Basics
   - $w^T x_i = \sum_{j=1}^{d} w_j x_{ij}$, x_i, w is d x 1
   - $a^T Ab = b^T B^T a$ both sides are vectors
   - $\frac{1}{2}\|Xw-y\|_2^2 = \frac{1}{2}\sum_{i=1}^{n}(w^T x_i - y_i) = \frac{1}{2}w^T X^T Xw - w^T X^T y + \frac{1}{2}y^T y$
   - $\nabla w^t b = w, \nabla\frac{1}{2}w^T Aw = Aw$ if A symmetric
   - $\nabla\frac{1}{2}\|Xw-y\|_2^2 = X^T Xw - X^T y$
   - Normal equation $X^T Xw = X^T y$
   - $(Xw-y)^T V(Xw-y) = \sum_{i=1}^{n} v_i(w^T x_i - y_i)^2$

   Run Time
   - $X^T y : O(nd)$
   - $X^T X : O(nd^2)$
   - solve d x d system of equations : $O(d^3)$
   - solve normal equation : $O(d^3 + nd^2)$

   Gradient Descent
   - $w^{t+1} = w^t - \alpha^t \nabla f(w^t) = w^t - X^T(Xw^t - y)$ (least square)
   - cost $O(nd)$ no need to form $X^T X$
   - total cost $O(ndt)$ t iterations
   - faster for large d, works generally
7. Multivariable Calc Notes
8. Probability Notes

Naïve Bayes Training Phase

$p(y_i = 1) = \frac{6}{10} \leftarrow n_1 = 6$

- Training a naïve Bayes model:

1. Set n_c to the number of times $(y_i = c)$.
2. Estimate $p(y_i = c)$ as $\frac{n_c}{n}$.
3. Set n_{cjk} as the number of times $(y_i = c, x_{ij} = k)$
4. Estimate $p(x_{ij} = k, y_i = c)$ as $\frac{n_{cjk}}{n}$.
5. Use that $p(x_{ij} = k \mid y_i = c) = \frac{p(x_{ij} = k, y_i = c)}{p(y_i = c)}$

$$= \frac{n_{cjk}/n}{n_c/n} = \frac{n_{cjk}}{n_c}$$

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 1 \\ 0 & 0 \\ 1 & 1 \\ 1 & 1 \\ 0 & 0 \\ 1 & 0 \\ 1 & 0 \\ 1 & 1 \\ 1 & 0 \end{bmatrix}, \quad y = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$n_{121} = 4$

$p(x_{i1} = 1, y_i = 1) = \frac{4}{10}$

$n_0 = 4$

$p(x_{i1} = 1 \mid y_i = 1) = \frac{4}{6} = \frac{2}{3}$ $\quad p(y_i = 0) = \frac{4}{10} \leftarrow n_0 = 4$