CPSC 392– Introduction to Data Science

Exam II Review

* Supervised Classifiers
  + k-Nearest Neighbors
    - Need k value and distance metric
      * K should be an odd number assuming binary classification
    - Difficult to determine k
      * Can find using error term for each k
    - Can be used for classification or regression
    - Does not work well with imbalanced data (gives preference to majority class)
    - Becomes slow as data increases in size
    - Does not need to train data
    - Given N training data points, kNN identifies k nearest neighbors of some data point c, regardless of labels
      * Does not use probabilities
      * Uses majority vote
      * Classify based on distances from other data points
    - Algorithm
      * Determine the value of k
      * For each testing data point
        + Find the distance between the test point and all training points
        + Sort the distances (small to large) and select the top k testing points
        + Find the simple majority label from the k candidates
        + This is the predicted label for test data point
  + Decision Trees
    - Need to calculate probabilities
    - Does not require normalization/standardization
    - Provides clear flow of how data is being classified
    - When prompted with a question, classify/label based on answer.
    - Better with small dataset
      * Can create too many branches
      * Complex and time consuming
    - Does not work well with categorical data
      * One Hot Encoding creates n columns when column has n attributes
    - Determine what the root node should be
      * Find distribution of data for each column to predict goal
      * Goal is to find distribution where there is a 0 (pure node)
        + In which, when making a decision, answer prompts only one answer
      * Nodes that have distribution > 0 are called impure nodes
    - Calculate "impurity" value to determine which column should be root node
      * Gini score = 1 - Σ(Pi^2)
      * Find weighted average of gini score's of each node to find "impurity"
      * Column with lowest "impurity" is the node that should be root
    - Next goal is to determine where the tree splits using the same method.
  + Naïve Bayes
    - Assumes attributes are conditionally independent
    - Has benefit of being really fast (sacrifice in accuracy)
    - Works well with high-dimensional data (spam emails)
    - If row exists in testing set but not training set => 0% of that event occurring
      * Can add all combinations of values and give the output a false value
    - Need to calculate probabilities
    - P(class | data) = P(date | class)\*P(class)/P(data)
    - P(class | data1, data2) = P(data1, data2 | class)\*P(class)
      * Naïve Bayes: = P(data1 | class)\*P(data2 | class)\*P(class)
* Unsupervised Clustering Algorithms
  + kMeans
    - Use if less noise compared to Hierarchical Clustering
    - Scalable
    - Hard to define k
    - Step 0: define k
      * specify how many clusters to look for in the data
    - Step 1: randomly initialize k centroids
      * centroids = μi
      * data points = xi
    - Step 2: find distance between each data point and each centroid
    - Step 3: assign cluster to data point based on minimum distance
      * min ||(xi-μi)||
    - Step 4: find the average of each cluster
      * his becomes the new centroid
    - Step 5: repeat steps 2-4 until the centroids obtain convergence
      * position of centroids does not change after additional iterations
  + Hierarchical Clustering
    - Difficult to implement
    - Cannot be used on large data
    - Flexible
    - Computationally expensive
    - Intelligent
    - Linkage
      * Ward: Differing density data
      * Average
      * Complete
      * Simple: Spherical or oddly shaped data (not uniform data
    - Computer Proximity Matrix
    - Let each point be a cluster
    - Merge 2 closest clusters (based on linkage) and update proximity matrix
    - Repeat previous step until only one cluster remains
    - Dendrogram
    - Health: largest distance range within which there are no splits
* Preprocessing Algorithms
  + Principal Component Analysis

For supervised classifiers, you need to know:

* How they work, what aspect of the data they are targeting (probabilities, distance metrics etc.)
* What kind of data they work best with
* What the underlying assumptions are for applying each algorithm
* Pros and cons of each algorithm
* Preprocessing steps that need to be taken before applying these algorithms
* Computing the accuracy of these models

For unsupervised clustering algorithms, you need to know:

* How they work, what aspect of data they are targeting
* How to measure accuracy of these models
* Elbow-method for kMeans and how to compute the optimal k value
  + SSE vs k
* Dendrogram generation for Hierarchical clustering and how to use it to find the k value
* Linkage and distance methods
* Pros and cons of each algorithm

For PCA, you need to know the math behind reducing a feature space using eigen decomposition. And when should PCA be applied.

* Preprocessing step
  + Standardization (same units)
  + Normalization (0 at origin for every axis)
* Process
  + Multiply matrix by transpose to get covariance matrix
  + Use eigen decomposition on covariance matrix to get W (matrix of eigenvectors)
    - Order matrix by eigenvalues
  + Multiply original matrix by Wk
    - Where k is number of features desired

I can also give you a scenario where you are given a data set and you need to figure out which machine learning technique to apply, or comment on if it is viable to use either one given a special condition. (Similar to Exam I)