# Exercises for chapters 7-9

1. What is the idea behind ensemble methods?

It uses the wisdom of the crowd; essentially, it relies on taking multiple opinions from diverse subjects to make aggregated predictions, hence improving accuracy and reducing the variance of predictions that an individual predictor would have.

1. Define: Bagging, Random Subspaces

Bagging: Bagging is training multiple instances of the same method on different subsets of your training data with replacement, (which allows an individual training instance to be used multiple times)

Random Subspaces: Same as bagging, but instead of sampling the data’s instances, you sample the data’s features.

1. Define: Boosting, Stacking

Boosting: Training predictors sequentially, with the goal of each one to improve the previous one by correcting its mistakes (hence giving relatively weak learners a boost).

Stacking: Training (a) predictor(s) to find the best weights for the votes of the previous layer of the stack, helping us to make better use of our ensemble.

1. What is the curse of dimensionality?

Problems that deal with a high number of dimensions can extremely slow machine learning algorithms and prevent them from finding the best solution.

1. Write code (in this document) to use PCA to reduce the dimensionality of a dataset X\_train while retaining 95% of its variance.

pca = PCA(n\_components=0.95)

X\_train\_reduced = pca.fit\_transform(X\_train)

1. Define unsupervised learning. Why can it have much more potential than supervised learning?

Unsupervised learning is a method used to train algorithms on training data that doesn’t come with labels. (This is a much broader spectrum; the set of all problems with training data is a superset of all problems with training data & labels, and unsupervised learning CAN technically be used for either.) Labelling data can require a ton of work, resulting in companies settling for smaller datasets (and hence, lower accuracy models).

1. Write the algorithm for K-means or DBSCAN clustering. What are the limitations of that method?
   1. Initialize *k*centroids to random instances
   2. Label the instances (assign them to the clusters based on their distance to each centroid and picking the closest)
   3. Move the centroids to the mean of all the instances of the cluster
   4. Repeat b & c until the centroids stop moving
   5. [ ???
   6. Profit :) ]
2. Write code (in this document) to cluster a dataset X\_train and then use the cluster affinities as input to a classifier (using a Pipeline as in the class code). Use Grid Search to find the best number of clusters.

# Copied code from the class notebook because it does exactly this… the only things that would’ve changed if I wrote it myself are the classifier type and the param\_grid format probably. Will go to office hours to ask if this is okay if I remember.

pipeline = Pipeline([

("kmeans", KMeans(n\_clusters=50)),

("log\_reg", LogisticRegression(max\_iter=5000)),

])

pipeline.fit(X\_train, y\_train)

# Grid search to find best k

param\_grid = dict(kmeans\_\_n\_clusters=range(2, 100, 10))

grid\_clf = GridSearchCV(pipeline, param\_grid, cv=3)

grid\_clf.fit(X\_train, y\_train)

*Due Monday 11/2 11:59pm EST. Submit by sharing your copy of this document with* [*rebeccalevitan@share.brooklyn.edu*](mailto:rebeccalevitan@share.brooklyn.edu) *and* [*denys.katerenchuk@gmail.com*](mailto:denys.katerenchuk@gmail.com)*.*