K Nearest Neighbors

* Dist bw new example and every other example
* K=number of examples
* Leverages distance calculation. Common types:
  + Euclidean: two vectors, most common
  + Hamming: binary, one hot encoded data
  + Manhattan: taxicab distance
  + Minkowski: adds p, generalizes Euclidean distance
* How to find k: run several times, choose number that reduces error
* K=1 may be inaccurate and unstable, odd k is good
* K too low may just model noise

K Nearest Neighbors:

* Calculate distance, get number of instances, make predictions
* KNN is computationally expensive
* Setting number of neighbors to 3
* My model is about 66%, given the imbalance in the data. Not bad.
* Soo my error rate function took very long to run. So I’m commenting it out.
* Using 17 neighbors actually increased my model accuracy to 71%,
  + while predicting positive numbers for all classes, hurray! Getting there
* Dropping it back down to 5 results in the same lower accuracy of around 69%

Model Metrics:

* Accuracy: Correctly classified points/total points in the data
  + Typically as a %
* Precision: how many of the predicted positive points were actually positive
* Recall: true positive rate. Ratio of positive points correctly predicted by our model
* F1: harmonic mean of precision and recall
* Keep in mind imbalance in our dataset
* For k=15, we get 71% accuracy in our model. With a few other tradeoffs.
* Accuracy doesn’t tell the full story

Searching for the best model:

* Sklearn and gridsearch to find best value of k
* Grid search: exhaustive search
* So with n\_neighbors = 8, the best model accuracy we can get is 70%,
  + without compromising other parameters
* Getting about 71% accuracy. However, still not doing that well with our positive cases
* Using rsearch takes about 5 minutes to run here. Anyhow.
* Large imbalance in classes makes it quite hard to predict which class it is