CM146, Fall 2018
Problem Set 1: Decision trees and k-Nearest Neighbors

Rodrigo Valle

29 October 2018

1 Problem 1: Splitting Heuristic for Decision Trees

Solution: Please check CCLE to see answer for this question.

2 Problem 2: Entropy and Information

(a) Problem 2a

Solution:

$$IG(S, X_j) = \mathrm{H}(S) - \mathrm{H}(S|X_j)$$

where S is the set of our training examples and X_j is the attribute that we're splitting on.

Also given: our examples S are drawn from a Bernoulli random variable, with p positive and n negative examples. Its entropy is given as:

$$B(q) = -q \log q - (1-q) \log(1-q)$$

And the entropy of S is given as:

$$H(S) = B\left(\frac{p}{p+n}\right)$$

Given that attribute X_j splits S into k disjoint subsets S_k, with p_k positive and n_k negative each, we see that splitting on X_j gives information gain as follows:

$$IG(S, X_j) = B\left(\frac{p}{p+n}\right) - \sum_{v \in vals(X_j)} \Pr(v)\mathrm{H}(S_k)$$

$H(S|X_j)$, or the conditional entropy, is the sum of the entropies of the subsets formed after the split, weighted by the probability that attribute $X_j = v$.

Plugging in $\Pr(v) = \frac{p_k+n_k}{p+n}$, we have

$$IG(S, X_j) = B\left(\frac{p}{p+n}\right) - \frac{p_k+n_k}{p+n} \sum_{v \in vals(X_j)} \mathrm{H}(S_k)$$

$$IG(S, X_j) = B\left(\frac{p}{p+n}\right) - k\frac{p_k+n_k}{p+n}\mathrm{H}(S_k)$$

noting that $k(p_k+n_k) = p+n$ follows from the given information, we simplify further:

$$IG(S, X_j) = B\left(\frac{p}{p+n}\right) - \mathrm{H}(S_k)$$

$$IG(S, X_j) = B\left(\frac{p}{p+n}\right) - B\left(\frac{p_k}{p_k+n_k}\right)$$

Now we argue that $B\left(\frac{p}{p+n}\right) = B\left(\frac{p_k}{p_k+n_k}\right)$ because $\frac{p}{p+n} = \frac{p_k}{p_k+n_k}$. We have that there are k disjoint subsets of S called S_k, each with the same p_k and n_k so

- $\sum p_k = p$
- $kp_k = p$

and

- $\sum n_k = n$
- $kn_k = n$

$$\frac{p_k}{p_k + n_k} = \frac{kp_k}{kp_k + kn_k} = \frac{p}{p+n}$$

Thus

$$\begin{aligned} IG(S, X_j) &= B\left(\frac{p}{p+n}\right) - B\left(\frac{p_k}{p_k + n_k}\right) \\ &= B\left(\frac{p}{p+n}\right) - B\left(\frac{p}{p+n}\right) \\ &= 0 \end{aligned}$$

3 Problem 3: k-Nearest Neighbors and Cross-validation

(a) Problem 3a

Solution: The value $k = 0$ will minimize the training error on this dataset. If a point can be its own neighbor, then when given a point from the training set $k = 0$ will re-select itself from the training set and give the correct label. Training error will be zero.

(b) Problem 3b

Solution: Using values of k that are too large will result in looking at extra points (outliers, almost) that are significantly further away from where we're trying to predict, resulting in more misclassifications. e.g $k > 7$ on a point at say (10, 1) causes us to begin considering the further group of points that have (ideally) less relevance, clouding our final prediction.

Using values of k that are too small can cause K Nearest Neighbors to overfit by not considering enough surrounding data to make a prediction that will generalize.

(c) Problem 3c

Solution: Assume L2 norm.

- $k = 1$, 10 wrong out of 14
- $k = 3$, 6 wrong out of 14
- ANSWER: $k = 5$, 4 wrong out of 14

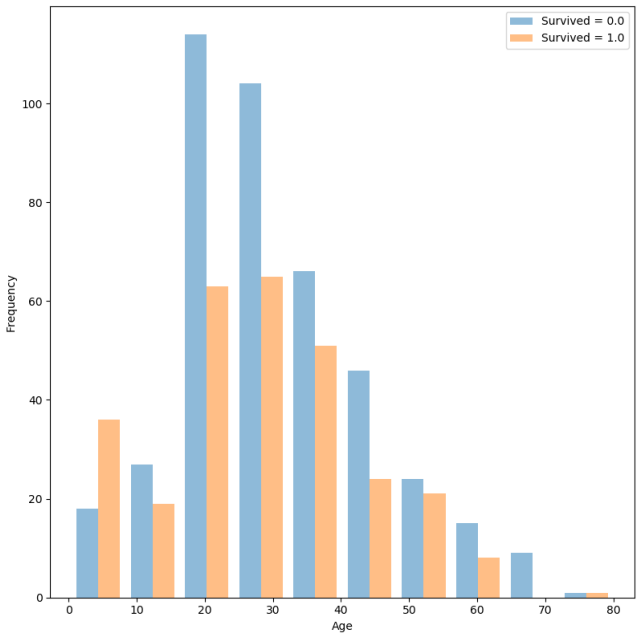
4 Problem 4: Programming exercise

- 4.1 Visualization

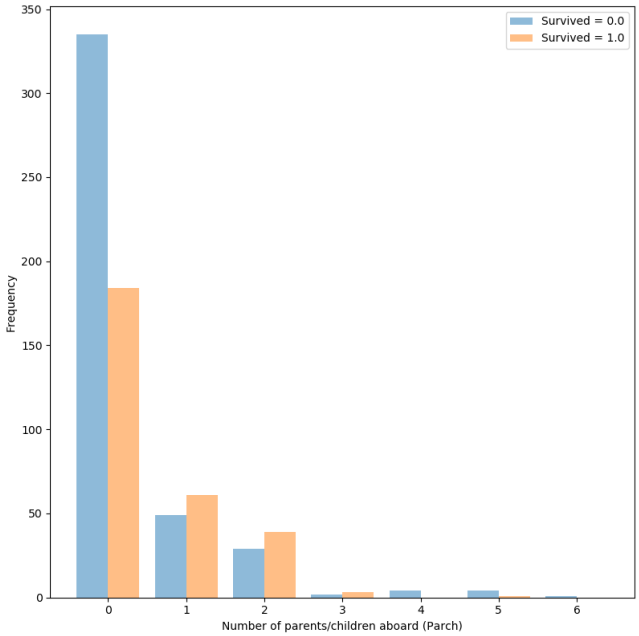
  (a) See Figure 1 below.

  From these plots, we can see

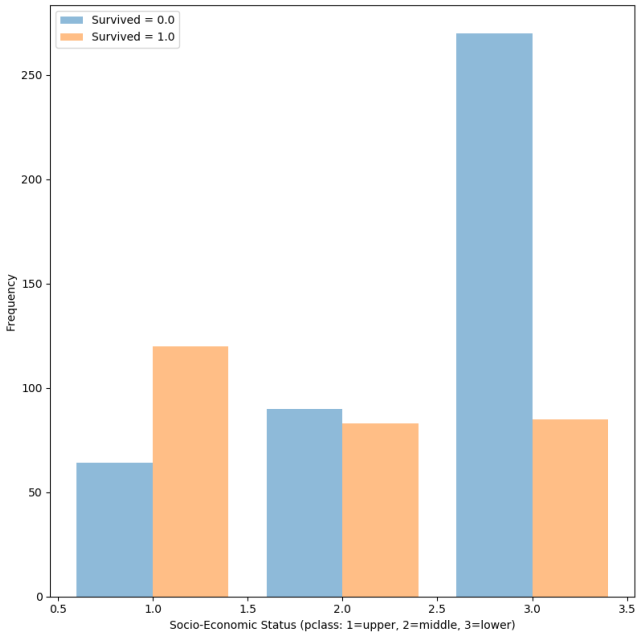
  - The only age group more likely to survive than die was in the 0-10 year range. Those between ages 20-30 were significantly more likely to die than live.
  - Those with at least one parent or child on-board were more likely to survive.
  - Upper-class passengers were more likely to live, middle-class passengers had closer to a 50/50 chance, and lower-class passengers were significantly more likely to die.
  - Women were more likely to survive than men.
  - Those with at least one sibling or spouse on board significantly decreased their chances of death.
  - Those who paid upwards of $50 for fare had increased chance of survival.
  - Those who embarked at the very beginning of the Titanic's trip had increased chances of survival. Others who embarked along the way were more likely to perish.



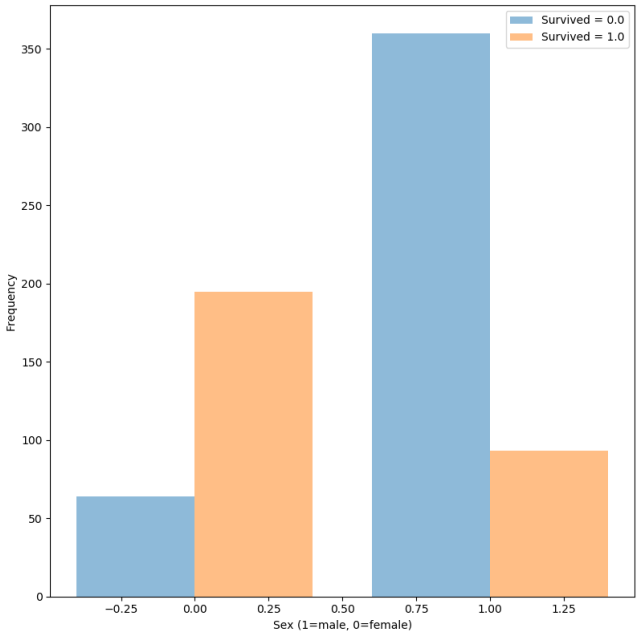
(a) Age



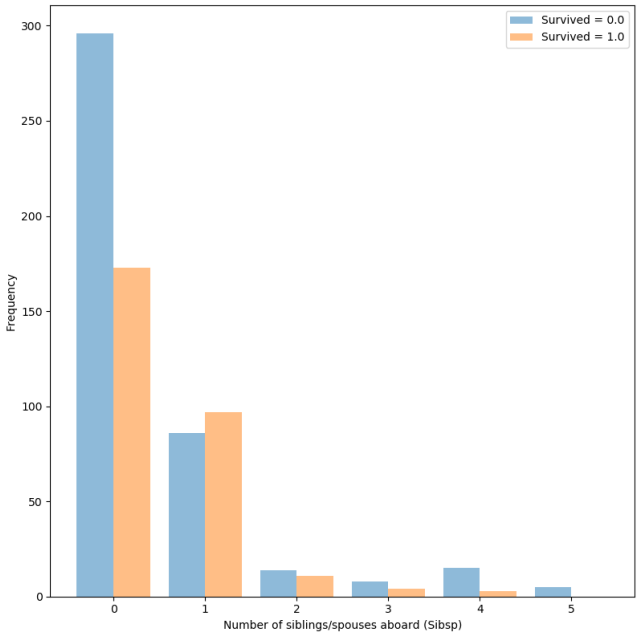
(b) Parch



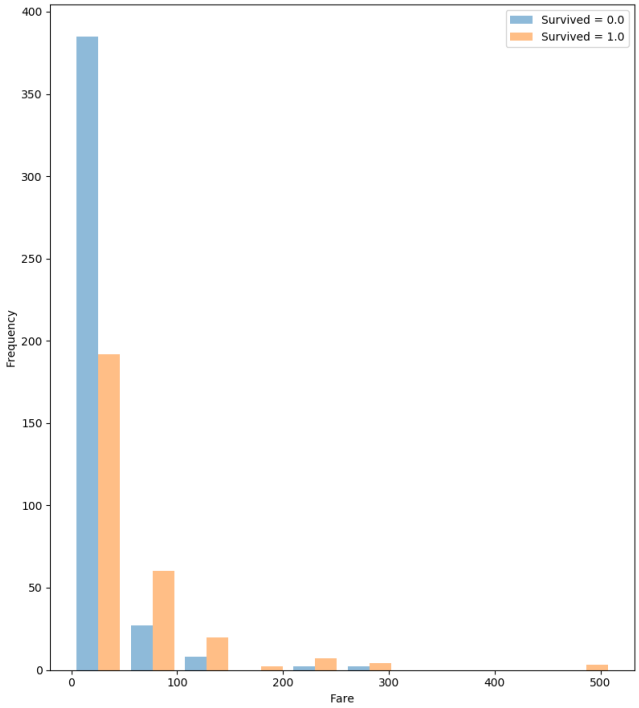
(c) Pclass



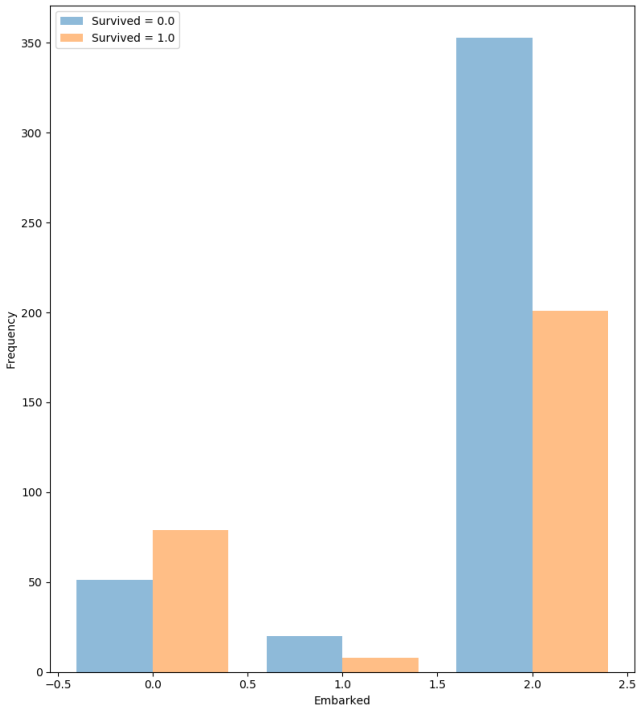
(d) Sex



(e) Sibsp



(f) Fare



(g) Fare

Figure 1: Distributions of all attribute splits

(b) Implemented `RandomClassifier`.

(c) Training error of `DecisionTreeClassifier`: 0.014

(d) Training error of `KNeighborsClassifier`:

- $k = 3$: 0.167
- $k = 5$: 0.201
- $k = 7$: 0.240

(e) Average training and test error of each classifier:

- `MajorityVoteClassifier`
  - train: 0.404
  - test: 0.407
- `RandomClassifier`
  - train: 0.486
  - test: 0.478
- `DecisionTreeClassifier`
  - train: 0.012
  - test: 0.241
- `KNeighborsClassifier`
  - train: 0.212
  - test: 0.315

(f) See Figure 2. Best k was found to be $k = 7$.

Observations: $k = 1$ is the worst k we found, clearly looking at more data helps us make a better estimate. Around $k = 33$ though, we see that looking at more data (increasing k) starts adding error to our estimates — this is where we've started to consider enough data points that are too far away from the point we want to classify that they outnumber the closer data points.

(g) See Figure 3. Best depth was found to be depth=3 (note: depth would change somewhat sporadically between 3 and 6 — cause unknown, could be a floating point hardware/math issue) (even better sidenote: switching to a different linux desktop would reliably change the result)

After depth=6, we can observe some overfitting: test error starts going up as train error trends down predictably. Our model has become too overfitted to the training data set and will perform amazingly well on training data that it has seen, but fails to generalize to the test data that it hasn't seen.

(h) See Figure 4.

Observations: With K Nearest Neighbors, training error slopes predictably downwards, and test error follows suit for the most part with some noise —

this means that training KNN with more data causes its predictive performance to increase, although not as reliably as we would hope.

With the Decision Tree Classifier, training error starts low and testing error high with a small training data set. With more training data, they converge to the point where training error closely mirrors test error, clearly indicating that increasing the amount of data available to train a decision tree will likely increase the quality of the prediction. It also showcases the decision tree to learn quickly, with quick convergence of training and test error at low fractions of the training data.

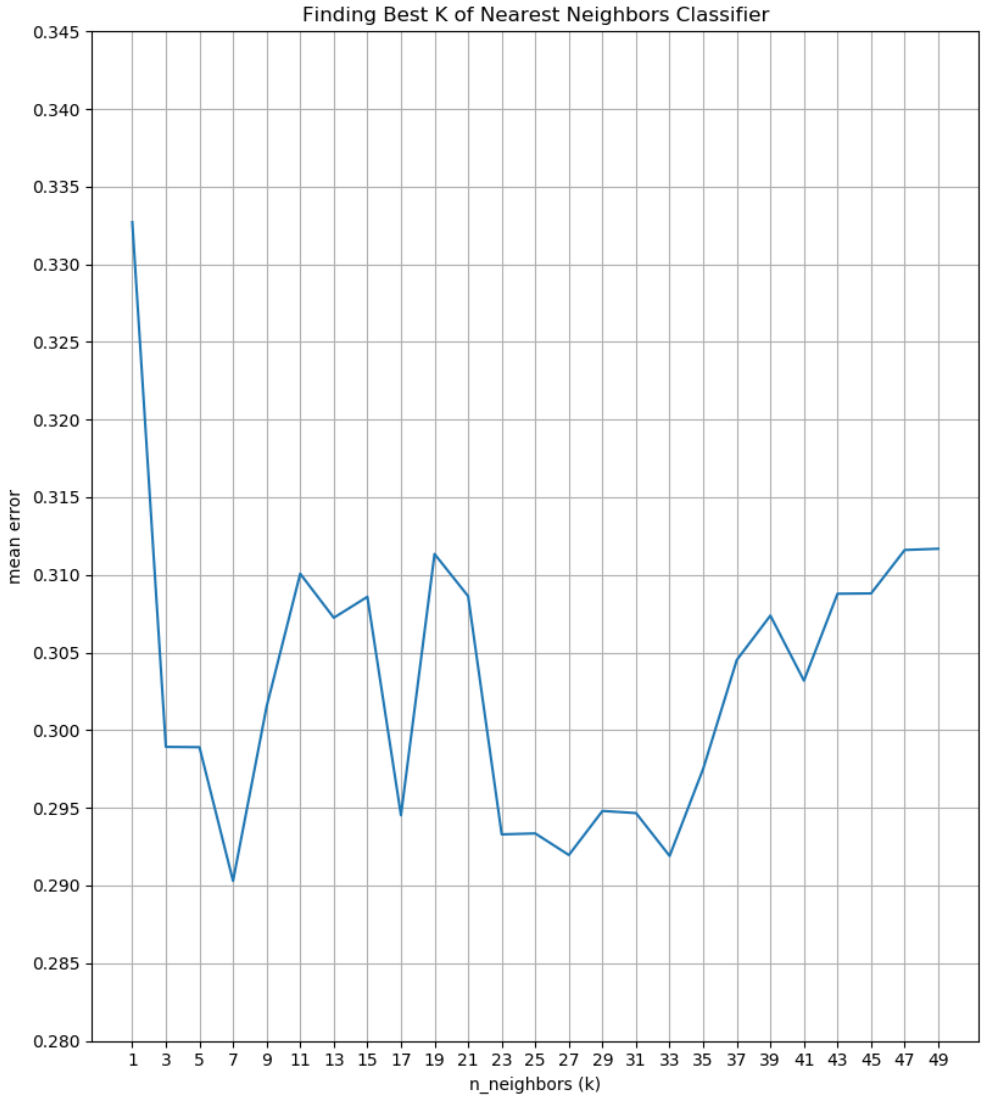


Figure 2: Best k

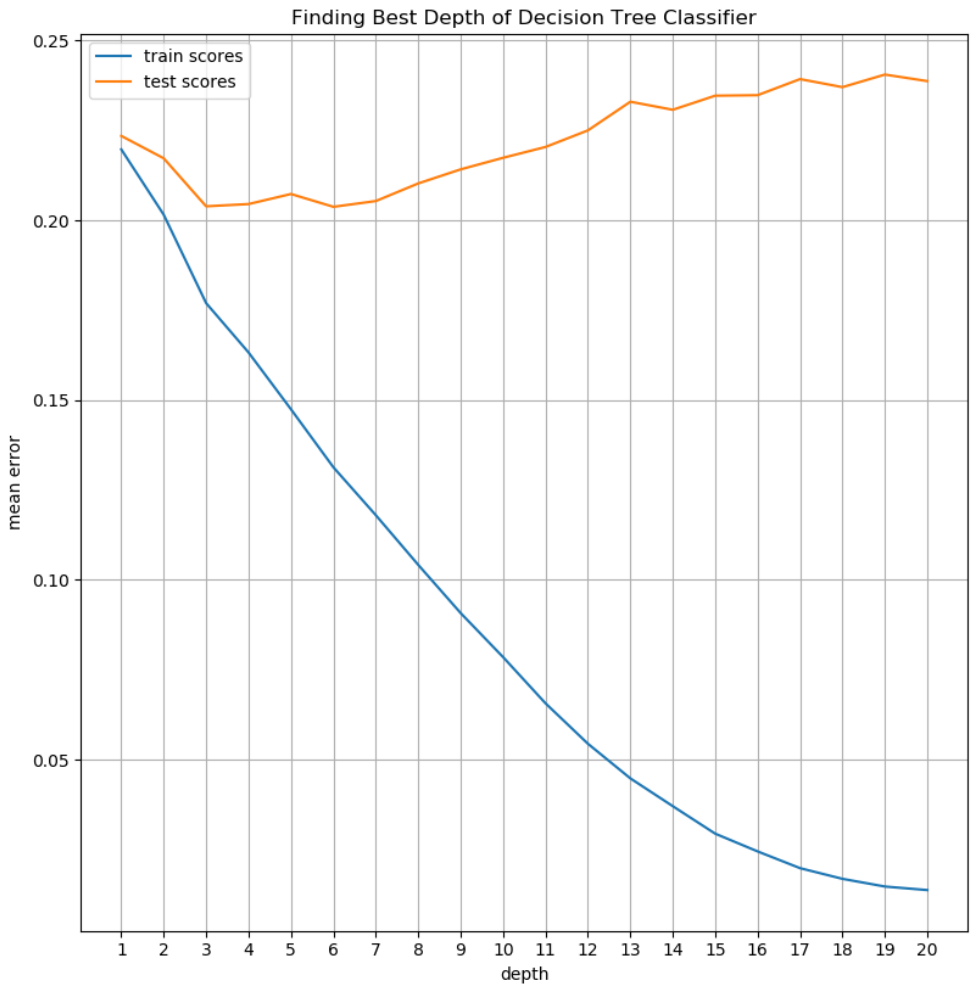


Figure 3: Best depth

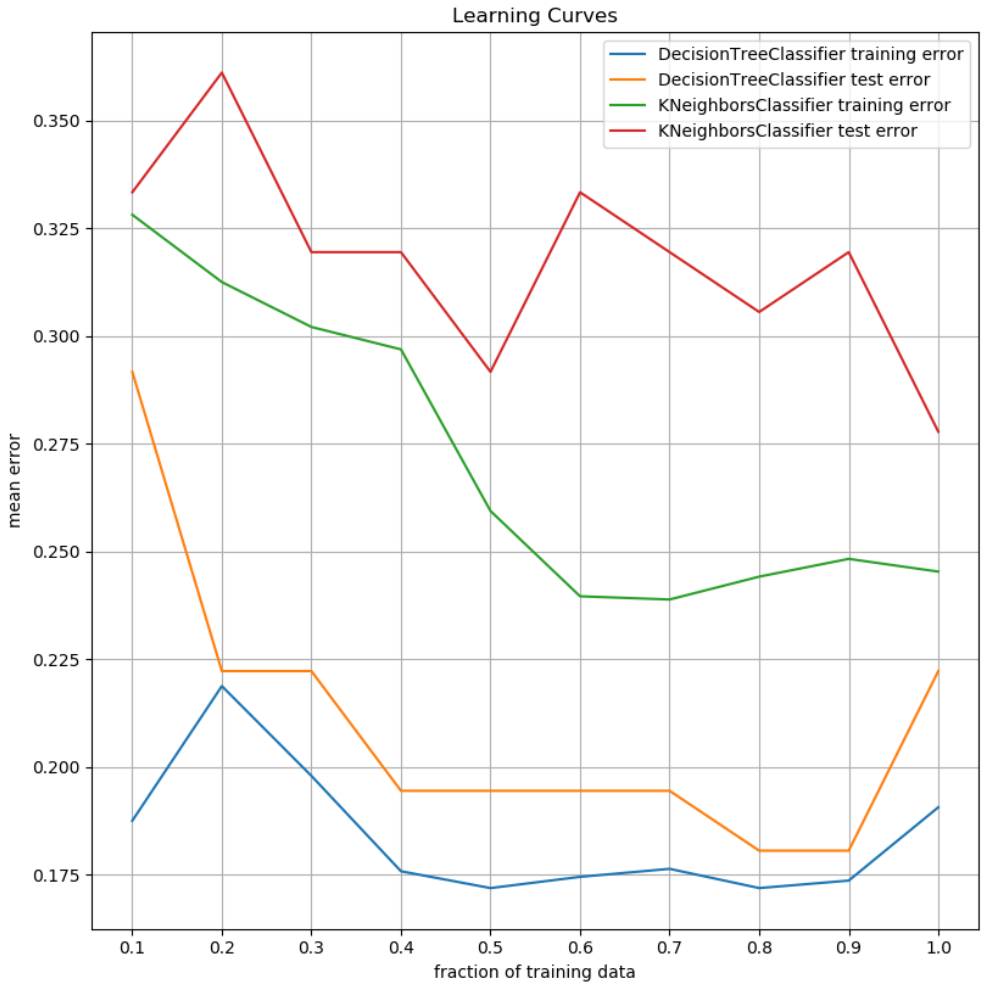


Figure 4: Learning Cruves for Decision Tree and K Nearest Neighbors