## [CAP4611-21Spring](https://webcourses.ucf.edu/courses/1369384/calendar_events/2158980)

# Day 10 (Thurs, Feb 11):

(?) = missing details needed to be filled in

Pre-recording stuff:

Review over quiz:

* “The results [of the last quiz] are a little bit left skewed” - Dr. Hollander
* Pruning, bagging, boosting used to reduce overfitting in decision trees
* A decision tree has 1 root node
* Decision trees can only be used if the target vector contains categorical data: false
* The Gini index is used to determine which feature is used to branch
  + Gini index **replaces** Entropy
* Bagging Aggregates predictions, sample columns and rows from the dataset
  + This is used for random forests, selecting a random subset of the rows and the columns
* Entropy measures the amount of uncertainty remaining after selecting a random element
* When building a decision tree, you should use the feature with the largest info gain

Administrative stuff:

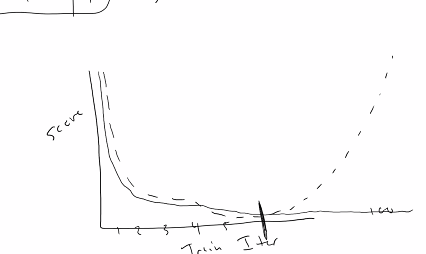
* Homework to be posted **tonight or tomorrow**
* If we finish evaluation today, quiz will be on Tuesday

Recording starts:

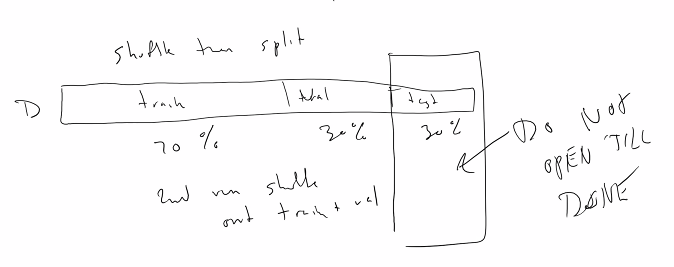
Today’s topic:

How do we split the data into training value and test sets?

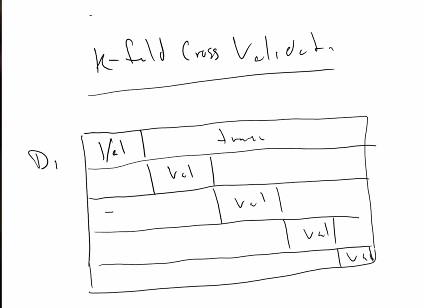
Recall:

* The easiest measurement to tell how good our algorithm is the misclassification rate
* We talked about if we build our model on the entire dataset, we’re going to overfit
* You build your model with the training set, you use it to predict the model on the validation data and comparing your values based on the predicted values. (?)
* The curves you draw are the score against your training iteration
* 
* Precision and recall are ways of determining how good your algorithm is
* Knowing that we can have a training and evaluation data training set is a way we can tell how well our algorithm does as we can
* Take our final model derived from the training data and test it on the evaluation data. That result will be its true performance.
* [Assignments] For the assignments, I’m going to give you a training dataset, and a testing dataset.
  + The testing dataset will not have the target column because
  + you will have to split up the training dataset, build your model, predict some features, submit your model and score it
  + To figure out how well your model does, it will be compared with a baseline (?) and posted on a leaderboard
* Once you go through the process of submitting your score, you will be able to see if you need to redo your model or reevaluate some things
* The training data will contain the target vectors and features
  + **You decide how to split the training data**
  + You have two weeks to figure out how to split the data and train your model
  + The right answer will get you the highest score
* Scikit has a few methods to split up data
  + train\_test\_split(x, y, test\_size= value between 0 - 1 , random\_state= INT)
* The general rule for your algorithm performance is, if you get 100%, something is probably wrong
* If you don’t find something wrong and it is 100% congrats on your PHD
* “The only way not to get an A on the first assignment is if you have no idea or start late”

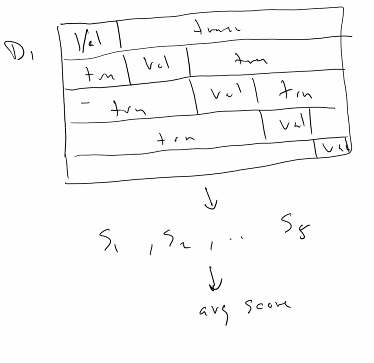
How od we actually split this test, valid split? (?)

* Imagine this is our data:
* 
* We can split it up into train, test, and validation sets
* The best way to populate these sets is to randomly add values to these splits
* There is no general idea on the percentages of these sets,
* If you shuffle and split, you could reset and do it again.
* Sometimes you would generate the sets, and separate the testing set
* Then shuffle the train and the validation sets again.
* **Splits can have a huge impact on the data**.
* If the splits are “imbalanced” or small
  + You might be missing certain labels/ranges within the data
* You might shuffle and split once,
* Separate the test set, and shuffle the train and validation sets
* 
* It could be that in the 100 runs that you build, X1 could always occur in the training data but never used in the validation dataset
* The way you could get around this is cross-fold validation

Cross-fold validation, k-fold cross validation :

* Popular to use on small amount of rows [1000s of rows]
* K is a number, with 5 and 10 being the most popular
* K-fold cross validation:
  + Takes your dataset
  + It says, “this is going to be your validation data, this is going to be your training data”
  + It then does 5 more runs [in 5-fold (?) cross validation] with 5 different chunks of your data
  + 
  + The idea with k-fold Cross validation is that you break up your data into K-Chunks, use one of those chunks as your validation, and use the rest as training
  + **This will guarantee that all your data will be used to train the data**

Summary of process:

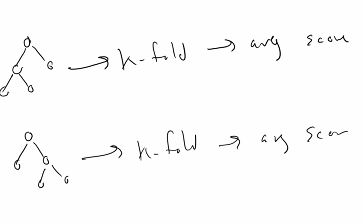
* 1. Split train and test
  2. Run k-fold on train
  3. Evaluate best model on test
* At the end of this process, you will get a score on the cross validation
* Each fold has its own score
* You end up with a series of scores for each fold, the final score is the average of this
* 

Regarding decision trees and k-fold cross validation

Decision tree -------> k-fold --------> avg score

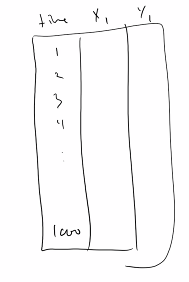
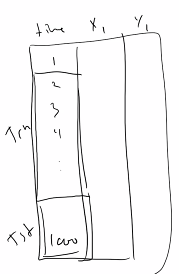
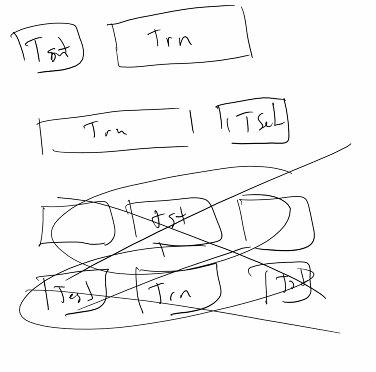
Different decision tree -----------> k-fold ---------------> avg score

Pick the best one

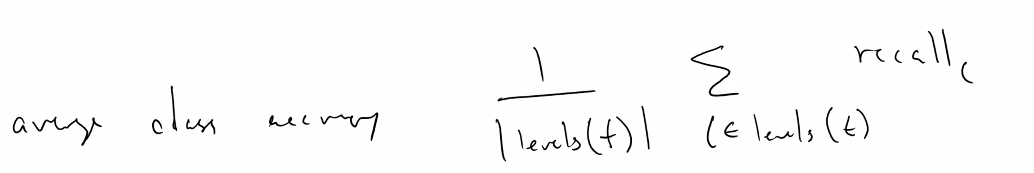
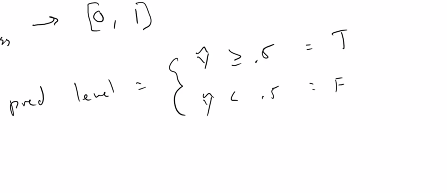
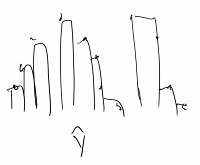


* Now the nice thing is that scikit has functions for cross validation
* Most ML APIs have functions for this.
* Basically you tell it how many different folds you want and how to split the dataset
* train\_test\_split(x, y, test\_size=.4, random\_state=42) # scikit stuff
* cv\_results = cross\_val\_score(someModel, features, targetVector, cv=5) #scikit stuff
* Now if you are in the situation where you have n-1 folds out (?) that is called **leave one out cross validation**
* Leave one out cross validation is:
  + You pick a single point as validation
  + You train on all the other data
  + You have n-1 folds
  + You train your model n-1 times, using each point as a prediction
* This is horribly slow for large datasets, but gives surprisingly good results
* [Recap] Bootstrapping:
  + You just pick a subsample of your data (200 elements train , 50 elements valid, 50 test)
  + Train your model
  + Pick another subsample, do it again
* The reason that you want to do cross validation (?) is so that you can artificially increase variety due to not having a lot of data.
* To really get a better estimate, build it a whole bunch of times and use the powers of statistics to average it together and see that it is more representative of the data we have
* But if I’m using data that has 1Billion points, the sheer amount of points should be good enough to represent the true answer than if I had a smaller dataset
* For k-fold cross validation, you don’t end up with K models, you use it on a single model. You’ll get a bunch of scores for one model, and you’ll need to take the average of that
* You’ll run cross validation 100 times, one for each of your models (?)
* Model selection - you’ll never know what is the best model for the dataset
* **No free lunch theorem** - Given no information all models are just as good as all others
* When you talk about **hyperparameters**, the things about a model you can change, they are not changed during training, but are set beforehand
* [Cross validation] the more folds you have, the more time it takes
* Parameters - the weights in a neural network, the number of training instances in decision trees, the minimum number of instances to branch with in a decision tree…
* Different algorithms have different hyperparameters
* When we talk about linear regression, they are in they are in the form y = ax +b
  + a and b are hyperparameters
  + You could also say ax +bx + c - (a, b,c are the hyperparameters)
* What if we make a randomizer to pick the hyperparameters?
  + That is a huge thing in machine learning, as it turns out, this is a high priority for automation
  + There is software out there to either systematially || randomly search the hyperparameter space to figure out which ones are the best ones to use
* So the whole idea with building machine learning models is:
  + You’re trying to find the best hyperparameters to give you the best model
  + “The more you can automate that, the easier your life is going to be”
  + It may take you a month to find a good model, unless you have access to a powerful cluster
* [reiterating] when you are dealing with ML problems
  + If its a complex problem, you won’t find the answer in a couple of seconds
  + It could take you a couple of hours, weeks, months, etc etc
* [tangent] why do you need a Graphics card for ML
  + Machine learning is mostly about matrix multiplication
  + Somebody figured out how to make a graphics card do none graphics things and pull the answers out
  + In kaggle, you can enable acceleration, that will allow you to access GPU
  + Problem with using a gpu is that it can get expensive
* [homework] will probably need to enable GPU for the neural network stuff

Other type of sampling you can do:

* Time sampling
* A lot of data you are dealing with in the real world has something to do with time.
* You might have a dataset that looks like this:
* 
* A time, x, and y column
* Think about bootstrap sampling & cross-fold sampling? What is the first thing you do?
  + Randomization
* **The thing is, if you try to use cross-fold validation on the data, you will lose temporal information**
* So, you got to split your data with respect to time:
* 
* The two sets are sequential, the training data comes before||after the test data
* However:
* 
* The testing data should not split the (?)
* “One of the great things about computer science is that you can try out your ideas and get your answers instantly”
* You want your test data, then your training data, or your training data then your test data
* The only thing that makes data time sensitive, depends on the context (?)
  + Data from filled out forms, doesn’t matter
  + Data from an active sensor, it matters
  + Data from a plane in flight, it matters
  + Data from an EKG, it matters

Other performance measures:

* Other performance measures that we…
* We can currently build a decision tree based model
* Split our dataset so we can train and evaluate our model for misclassification
* We have a couple of things from our confusion matrix:
  + True positive rate (TPR) = true positives / (true positives + false positives)
  + True negative rate (TNR) = true negatives / (true negatives + false negatives)
  + False positive rate () = Another metric
  + False negative rate () = Another metric, nice to know…
* There are two scores that are popular to combine FP and FN
  + Precision = True positives / (true positives + false positives)
    - Out of all the positive predictions, how many are correct
  + Recall = true positive / (True positive + false negative)
    - Out of all the positive instances, how many are correct (??????)
* We can then combine precision and recall to create the **F1 score:**
* F1 score = 2 \* (precision \* recall ) / (precision \* recall)
  + Works pretty well with imbalanced datasets
* Another metric for imbalance datasets where 90% is 0 and the other 10 is 1,
  + **Average class accuracy** = 1 / | levels(f) | sum( recall of the level) (consult book for a clearer explanation) 
  + The book uses the term levels:
    - **Levels - The unique values a feature can take on**
  + Profit and loss:
    - Sometimes its more important to calculate true positives rather than false positives
  + One thing that is really important:
    - When your dealing with classification models, decision trees are a little bit of an exception
    - Most classification models will give you a value between 0 and 1
    - At what threshold do you identify an output as class 1 or 0?
      * Some people say 0.5, = class 1. Otherwise class 0
      * The way you can determine this if you don’t want a flat 0 or 1 is
      * 
    - If you want to be thorough about this, you can plot the distribution of the predicted scores:
      * For every instance of your training data, you get a prediction of what that value is:
      * 
      * When you look at a distribution of this, instead of a curve you’ll see a histogram. You can use these histograms to see where these distributions are
      * For more information, look at **Bayesian Statistics** for more advanced ways to do this
* **Quiz moved to Thursday**

Tuesday:

R squared

Rock (?) curves

Assignment will be posted soon

Notes for what we covered will be posted soon

Recording stopped

Assignment 1:

1. Here is the dataset
2. Use decision trees
3. Play around with it, get a model
4. Beat the baseline score

More talk about Kaggle, and how to