**Unit 10 – Machine Learning**

Data Camp – ML using scikit-learn

* Supervised learning has labels on the data while unsupervised does not
* Classification with categorical data, regression for quantitative data
* ML needs to learn from already classified data
* Need to convert categorical features to numerical?
* *Classified data:* KNeightborsClassifier, can have different number of k, higher k is smoother, smaller k is in danger of overfitting
* Get data into proper formatted array prior to using scikit-learn
* OLS – ordinary least squares – minimize the square of the errors
* K-fold cross validation – divide into 5 groups, train on groups of four and test on one, 5 times
* Regularized regression – models are penalized for large coefficients
  + Picking alpha similar to picking k in k-NN
  + Lasso regression similar to ridge regression with the alpha
    - Shrinks coefficients of unimportant variable to 0
    - Lassos in only important variables
  + Alpha of 0 is OLS
* Classification problems
  + 99% of emails are not spam so if we build a model that labels all emails as not spam, our model would be 99% accurate, but useless
  + High precision: not many real emails predicted as spam
  + High recall: predict most spam emails correctly
  + Confusion matrix for categorial data
  + Logistic regression of binary category regression
    - Comes up with a probability for each
    - Have a probability threshold to categorize yes/no
  + A perfect model would maximize data under the AUC
* Scikit-learn does not accept categorical variables, need to convert them to dummy variables
  + pd.get\_dummies in pandas
* K nearest neighbors takes the actual distance, so a larger range will make the variable more impactful
  + We may want to scale our data to make each variable have equal have similar impact
* Minimize a log loss function (binary prediction) rather than maximizing accuracy (spam email predictor with 99% chance of success)

DataCamp Machine Learning with the Experts School Budget

* You cannot just pipeline in modifications to text and numeric data as the functions will try to operate on the entire data set
* FunctionTransformer
* The FeatureUnion brings the two separate pipelines together
* Tokens can also check and see if terms appear together (interaction terms
  + Beta1 \* x1 + Beta2 \* x2 + Beta3 (x1\*x2); x1 and x2 dummies must both be one only if they appear together

Bias and Regression Lecture

* Publication bias for p values (0.05 is convention)
* Survey non-response bias
* Selection bias
* Length bias (prisoners with longer sentence more likely to be in prison)
* Bias(theta-hat) = E(theta-hat) – theta; how far from the actual average
* MSE(theta-hat) = Var(theta-hat) + bias^2(theta-hat)
* I DON’T REALLY UNDERSTAND BIAS AT THIS POINT
* More complicated model has more variance although it could minimize bias

Regression Continued

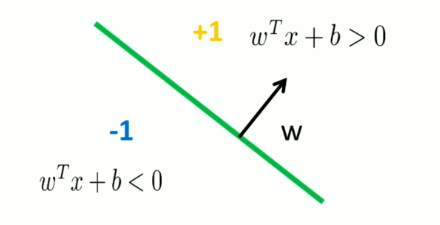
* Logistic Regression is for binary outcome; logit(p) = ln(p/(1-p)) log of the odds
  + Standard way to fit the model is MLE (maximum likelihood estimation)
  + Great property of logistic regression e^Beta gives you the odds ratio holding everything else equal except for the associate Beta
    - Easy to interpret the parameters
* Taking the log of variables of continuous variables is common for linear regression
* 5 dummies for five cases would be a blunder because of collinearity
  + All five would always add up to one and would always be a perfect linear relationship
* Collinearity makes the parameters meaningless and unstable
* Odds ratio – comparing probabilities across groups
  + Could there be other (confounding) factors? Fen/Phen users are more likely to be obese which is associated with heart disease; the users are different
* Ridge Regression: subtract out the sum of squared Betas; try to penalize model complexity; will reduce Betas towards 0
* LASSO Regression: subtract out the absolute value of Betas; penalize model complexity. Helps induce sparsity, reducing the number of variables one has to deal with; this will “brutally” reduce Betas to 0

Classification Lecture

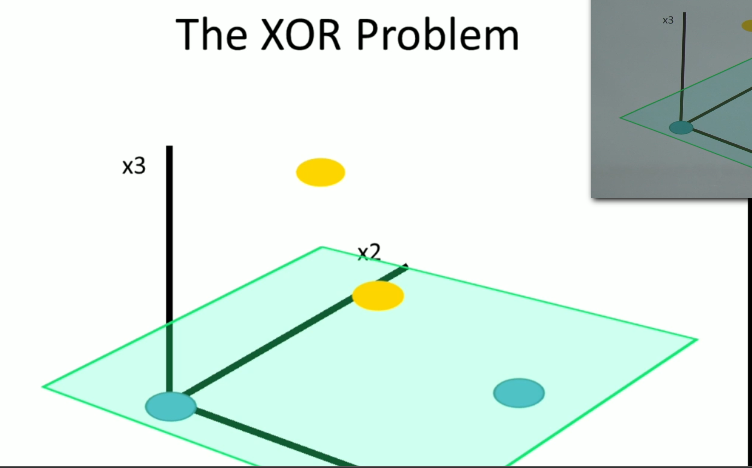
* Machine learning helped with image classification massively
* Humans can look through images and then the machine can learn from this
* You need to pick the right Feature (Selection) to help you distinguish in your model
* kNN: how many neighbors? Majority voting which is why we make k odd
* 1-NN: can create islands which isn’t ideal; the more complex the smoother the decision curve
* We would like training to take as long as necessary, but we want testing to be rapid
* We pick the k that has the lowest error?
* You need to hold out test data otherwise your results are not generalizable
* How can we bring the number of dimensions of our data down while persevering distance?
* Linear Regression vs PCA (principal component analysis)
  + Regression minimizes the vertical distances, PCA minimizes the orthogonal distance from the fit line

SVM and Trees

* Support Vector Machines
* kNN has the problem of needing the entire data set to predict. Training is fast because no calculation is really necessary, prediction is very slow
* How can we address this problem? Classifiers that have knowledge of the decision boundry



* this model will predict very quickly however we have lost a lot because we have a hyperplane (line) rather than the curved shape of kNN
* Perceptron is a mathematical representation of a neuron in the brain



* You could have a simple 2d problem that could not be separated by a hyperplane. If you lift to an additional dimension you could separate the two. You want to keep only the dimensions that are useful, not too few or too many.
* Support Vector Machines
  + some say it is the best off the shelf classifier
  + however, you may need to tune hyperparameters for best performance
  + Maximum margin classification to max the distance from the hyperplane to the data
* An outlier (or mislabeling) could radically impact the hyperplane
* We can tell the machine to give slack with slack variables; large C means less slack; higher gamma means more df (more dimensions); tune these together
* Only need support vectors which define your decision boundary
* You should normalize data for SVM (mean: 0, std: 1)
* Plotting ROC curve
* Recall: if I pick a random positive example what is the probability of making the right prediction
* Precision: if I take a positive prediction example, what is the probability that it is a positive example?
* Run n classifiers for n categories, one against all
* Compute a confusion matrix