Lesson 14 – Supervised Learning

**Questions for Mentor:**

**Choosing a Machine Learning Classifier:**

* Naïve Bayes
  + Very simple
  + Need less training data
  + Can’t learn interactions between features
* Logistic Regression
  + Don’t have to worry as much about features being correlated
* Decision Trees
  + Easy to interpret & explain
  + Don’t support online learning – need to rebuild with new data
  + Easily overfit
* SVMs
  + High accuracy
  + Especially popular in text classification problems
  + Hard to interpret
  + Memory intensive

**Supervised Learning with scikit-learn:**

* Classifications
  + K-Nearest Neighbors
    - Creates decision boundaries
    - Training a model is done with .fit() method
    - .predict() method is how we predict
    - Must be either numpy array or dataframe
    - Must be numbers, not categories
    - Can’t have missing data
    - .values will turn data into numpy array (won’t have to worry about target variable being a series
    - Higher K becomes less complex model – more smoothed
  + Accuracy = fraction of correct predictions
* Regressions
  + Continuous variable as target
  + Np.linspace() allows you to see results of linear regression model
  + Linear regression is an Ordinary least squares model
  + Default scoring method for lin reg is r^2
    - Syntax: .score(X\_test, y\_test)
  + Cross validation
    - 5 folds
    - Do 5 different iterations, using each fold as a test set and remaining for training set
    - 5-fold cross validation
    - Can use more folds (k-folds CV)
  + Loss functions
    - Large coefficients can lead to overfitting
    - Penalize models with large coefficients
    - Alpha in ridge regression is like choosing k in k-NN
  + Lasso regression
    - Great for feature selection
    - Must choose alpha
  + Ridge regression
    - Should be first choice when building regression models
    - Must choose alpha
* How good is classification model?
  + Confusion matrix
  + True positives, true negatives, false positives, false negatives
    - True positive = recall
    - = precision
    - Precision=TP / TP+FP
    - Recall=TP / TP+FN
  + Syntax: confusion\_matrix(y\_test, y\_pred)
* Logistic regression works for binary classification (not regression)
* Receiver operating characteristic (ROC) curve
  + Syntax: (roc\_curve(y\_test, y\_pred\_prob)
  + Predict\_proba(test set)
  + Larger the area under the curve = better model
  + Area under ROC curve (AUC) is popular metric
* Hyperparameter tuning
  + Parameters that can’t be learned by fitting the model
  + Usually have to try a bunch separately and choose best performing one
  + Grid search
    - GridSearchCV()
    - Can be computationally expensive
    - RandomizedSearchCV() is an alternative
* Dealing with categorical features
  + Dummy variables
  + OneHotEncoding
* Handling missing data
  + Replace with nans, 0s or whatever fits
  + Can impute with the mean or other calculation using a transformer
  + Can fill (forward or backward)
* Centering and scaling data
  + Many models use distance to inform them
  + With features much larger than others, it can influence model
* Combat imbalanced classes in datasets for classification models
  + 70% in class A (USA), 25% in class B (Europe), 5% in class C (Asia)
  + SMOTE or the Synthetic Minority Over-sampling Technique is a great way to oversample (get more data)
  + L1 and L2 distance – distance between 2 pictures?
  + Too many features = curse of dimensionality
    - Project high dimensional data onto lower-dimensional subspace that best fits data
  + Principal components analysis (PCA)
    - Dimensionality reduction
      * Used for the numbers and facial recognition
    - Compression
    - Visualization
    - Principal components chosen to minimize orthogonal distances
    - Best fit line like regression but based on orthogonal distances so the line will be different from lin reg
    - Steps:
      * Scale
      * Compute cov matrix S
      * Compute k largest eigenvectors of S
      * These eigenvectors are the k principal components
  + Multidimensional Scaling
    - Find a set of points whose pairwise distances match a given distance matrix
* Bias
  + Selection bias
    - How you collect and select your data
  + Publication bias
    - Mostly only get to publish if meeting an arbitrary ‘significant’ threshold (p-val)
  + Non-response bias
    - Not getting responses from portions of population who are needed to make it a representative of population
  + Length bias
    - Enough data
  + Must consider what’s unobserved (plane example)
  + Bias is how far away you are from the average
  + Want a good bias variance trade off
  + Bonferroni correction – divide significance level threshold by # of hypothesis (typical significance level is 0.05 alpha)
  + R^2 measures goodness of fit within sample for regression
    - Also called ‘Explained’ Variance
* Regression
  + Collinearity
    - Should avoid having predictor variables that are highly correlated with each other
    - Results in instability, high variances in estimates and worse interpretability
  + Logistic regression
    - Better when trying to predict binary response
    - Fits an ECDF shape
  + Odds ratio = p / (1-p)
  + Confounding factors
    - What if fen/phen users are more likely to be obese and obesity increases risk of heart disease?
    - Can use logistic regression to compare, controlling for other predictors
  + Logit(p) function
    - Log of the odds
    - Rather than y on left side (continuous var) – we have the logit of probability
    - No longer need error term
  + Ridge regression
    - Sum of squared residuals – ridge wants to minimize
  + Steins paradox and shrinkage estimation
    - Effron and ‘not sure’ project on baseball average estimation
    - Can use ridge regression for the shrinkage estimation
  + Lasso regression
    - Induces sparsity which reduces variables one has to work with
* Decision Trees
  + Advantages
    - Fast training
    - Fast prediction
    - Easy to understand
    - Easy to interpret
  + Disadvantages
    - Sensitive to small changes in data
    - Overfitting
    - Only axis aligned splits
  + Focuses on only one feature at a time
    - Benefit: Performance, intuitive, sequential process, categorical features allowed, don’t need to normalize
    - Cost: all straight lines – axis aligned split – can’t go diagonal
  + Learning tree structure
    - Which feature to query
    - Which threshold to choose
  + Node purity
    - Want to split up data so splits only have one class (i.e. only greens, no red dots)
  + Misclassification
    - Smaller number of the results of a tree branch
* Singular Value decomposition
* Ensemble methods
  + Single decision tree don’t perform well
  + But its super fast
  + What if we learn multiple times?
    - Bootstrapping? - no
      * Bootstrap has overlap in datasets
      * Don’t use simple bootstrap to generate train and test data from same data set
    - Bootstrap aggregating – Bagging
      * Sample with replacement for data set
      * Learn classifier for each bootstrap sample
      * Avg results
      * Reduces variance without introducing too much bias
      * Normally uses one type of classifier
      * Decision trees are popular – but not limited to DTs
      * Not helping with linear models
        + Want a classifier that’s easy to overfit
      * Easy to parallelize
    - Random Forest
      * Build upon idea of bagging
      * Each tree build from bootstrap sample
      * Node splits calculated from random feature subsets
* Random forests in Python
  + Aggregation of simpler decision trees
  + Tree order is based on entropy minimization aka maximizing information gain
  + Ensemble models allows us to avoid overfitting
  + Strong suits
    - Robust with different types of data
    - Great out of the box tool
  + Neural nets
    - Tougher to tune parameters
    - Random forests inherently parallelizable and very well supported for distributed deployment
* Ensemble methods and Random forests
  + Ensemble learning
    - Combine meany weak learners that may be weak on their own
    - Weighted average, weighted majority voting
    - Wisdom of crowds?
    - Several methods like bagging, random forests, boosting
  + Random forests
    - Randomizes training set as well as predictors
    - # of random predictors chosen to be of order the square root of total # of predictors
  + Bagging
    - Training a bunch of individual models in a parallel way
    - Each model trained by random subset of the data
  + Boosting
    - Trying to learn over time
      * Sequential in nature
    - Votes are weighted
    - Higher weighted on better performers
    - Tuning parameters for boosting
      * # of trees
      * # of splits in tree (stumps often work well)
      * Parameters controlling how weights evolve
  + AdaBoosting
    - Adaptive boosting
    - Fit a set of decision trees to a dataset
    - Calculate the weighted error rate of each decision tree
    - Calculate each decision tree's weight in the ensemble
    - Increase the weight of the incorrectly classified data points
  + Gradient boosting
    - Ensemble learning method that takes an iterative approach to combining weak learners to create a stronger learner by focusing on mistakes of prior iterations
    - Predicts using residual error of the models rather than weighing data points
    - Similar to random forest but not same
    - Differences:
      * GB uses boosting, RF uses bagging
      * GB trains iteratively, RF training done in parallel
      * GB uses weighted voting for final prediction, RF uses unweighted voting for final prediction
      * GB harder to tune, easier to overfit, RF easier to tune, harder to overfit
    - GB can be more powerful if fit correctly
    - GridSearchCV
      * Param is a dictionary
        + Keys = hyperparameters
        + Values = list of settings you want to try out
      * We only care about the test score, not as much the train score. So we can see if they’re good generalizers
* Time series analysis
  + Time series – ordered sequence of values of a variable at equally spaced time intervals
  + Like regression, time series analysis is often focused on identifying underlying trends and patterns, describing them mathematically, ultimately making a prediction or forecast about what happens next
  + Time series modeling decomposes mathematical process (y = bx) into a combination of signals and noise
  + Signal – underlying trend that’s visually apparent
  + Noise – random process that can be described as a monte carlo experiment (randomly sampling from normal dist)
  + Terms describing nature of signal and noise:
    - Statistical moments
      * Mean and standard dev
    - Stationary vs non stationary
      * Trends in mean and/or standard dev
    - Seasonality
      * Periodic patterns
    - Autocorrelation
      * Degree to which time series values in period (t) are related to time series values in periods (t+1, t+2, …)
  + Preprocessing and filtering of time series
    - Sometimes need to change attributes of a TS or deconstruct into component parts.
    - Often needed quite a bit before we can run TS analysis
    - Apply filters to data depending on our needs
  + Differencing – Yt – Y(t-1) – will only show difference between t and the prior t
  + Random walk – Yt = Y(t-1) + Wt – W = white noise
    - Some cases, W goes down, other cases it goes up
  + Can take out seasonality
    - Subtract out mean after linear detrending
  + Autocorrelation
    - To detect non-randomness in data
    - To identify an appropriate time series model if data are not random
    - Can model autocorrelation as a function of lag – can test many different lags to test their autocorrelation
  + Smoothing
    - Form of filtering which produces a time series in which the importance of spectral components at high frequencies is reduced
    - One logistical issue is that you lose data when smoothing
  + Statistical filter – series of weights that when cumulatively multiplied by consecutive values of a time series gives the filtered series
* Data Transformation and forecasting
* Time Series Analysis in Python
  + Pd.to\_datetime(df.index) to transform index string to datetime object
  + Autocorrelation
    - Correlation of time series with lagged copy of itself
  + Autocorrelation function
    - Can forecast the future using the past
    - Plot\_acf(x, lags=, alpha=) plots autoforrelation function
    - Can also see numerical values using acf() function instead of plotting
  + White noise
    - Series with
      * Constant mean
      * Constant variance
      * Zero autocorrelation at all lags
    - White noise can be generated with np.random.normal()
  + Random walk
    - Todays price = yesterdays price + noise
    - Can’t forecast
  + Stationarity
    - Entire distribution is time-invariant
    - Mean, variance and autocorrelation are time-invariant
    - Why do we care?
      * Difficult to model if parameters change with time
  + AR Model
    - AutoRegressive model
  + Estimating and forecasting AR model
    - ARMA(simulated\_data, order=(1,0))
      * Order=(1,0) means it is an AR1 model, Order=(2,0) means it is AR2 model
  + Choosing the right model
* Support Vector Machines
  + Distance between edge of cluster and threshold is called the margin
  + Maximal margin classifier – using a threshold directly between the eges of clusters
    - Not always optimal – does not handle outliers well
  + Choosing a threshold that allows misclassifications allows us to handle outliers and pick a more appropriate threshold
  + Soft margin – distance b/w observations and threshold when we allow misclassifications
    - Use cross validation to set threshold and determine that we are classifying well
  + Soft margin classifier aka support vector classifier
  + Support vector classifier is a line when data is 2d
    - i.e. rat height and weight
  + doesn’t handle data that has points on either side of a cluster (drug doses being just right)
  + Support vector machines solve this problem
  + Square each observation to make it 2D
  + Can now fit a line with support vector classifier that classifies one side or the other
  + Use kernel functions to find SVCs in higher dimensions
  + Kernels calculate relationship b/w observations in higher dimension without actually converting the data to higher dimensions
    - Saves computational power
    - Saves time
  + Perceptrons – ground work for deep learning
  + Support Vector Machine
    - Support vectors are the observations that guide the maximum margin classifier line to where it is
  + Slack variables are outliers that are on the wrong side of the classifier
  + Kernel trick
    - Arbitrary many dimensions
    - Little computational cost
    - Maximal margin helps with curse of dimensionality
  + SVMs are not scale invariant
    - Must be scaled, normalized (mean: 0, std: 1)
    - Test set must be normalized in same way
  + SVM parameter tuning
    - Which kernel?
    - Which kernel parameter values?
    - What value for C?
  + Multi-class SVM
    - Train n classifiers for n classes
      * Will fit n lines