**Case #3**

Using the “emailDFrp” dataset:

1. Build and evaluate a tree-based model for predicting “spam”
2. Plot and analyze the paths through one (or many) of your trees
3. Explain the parameters involved in “tuning” your model
4. Which variables were “most” important?
5. How did you evaluate the “performance” of your model?

**NOTE:** You should use “split” your data when training your model.

\*\*In Python – the equivalent to split in R is …

Did an 80 / 20 training/test…

Did not split on evaluation of accuracy.

ROC – did CV predict, did a 10-fold.

Samples =

Removed = 282 (what are the ones that were removed? All the NAs?)

what did we use to split our data?

Did we use cross validation?

Accuracy – how many correct in total

Precision – how many correct “true predictions” divided by how many in total predicted to be true

Recall – how many correct “true predictions” divided by all true records

F1 – balanced precision and recall (for positive class)

Type I error – false positive

Type II error – false negative

Matthews Correlation Coefficient – balanced accuracy for binary problems

\*from class slides

Gini impurity – measure of how often randomly chosen element form the set would be incorrectly labeled if it was randomly labeled according to the distribution of labels in the subset.

Information gain – based on concept of entropy and information content, decide which feature to split on at each step in building the tree, simplicity is best.

<https://en.wikipedia.org/wiki/Decision_tree_learning>

root node – first best feature to split data, defined by splitting criteria

leaf node – partitions of data set from split at root

decision node is where the prediction is made, after all possible splits (by features or based on stop criteria), decision made about target variable

<https://medium.com/swlh/making-decisions-with-trees-559c8db5af59>

\*\* weighted gini split – to help decide what first split to make, select one with lowest probability of misclassifying the observation

Gini impurity – how heterogenous or mixed some value is over a set

<https://medium.com/@jason9389/gini-impurity-and-entropy-16116e754b27>

<https://towardsdatascience.com/gini-impurity-measure-dbd3878ead33>

<https://towardsdatascience.com/how-to-tune-a-decision-tree-f03721801680>

<https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html>

Hyperparameters:

Can be found by Grid Search, Random Search, Bayesian Optimization

* Criterion - How to split decision tree nodes, using impurity (gini, entropy for information gain)
* Splitter – strategy to split each node (“best” or “random”)
* Max Depth – max depth of tree, if none nodes expand until all leaves are pure or until all leave contain less than min-samples-split samples.
* Min Samples Split – minimum number of samples required to split internal node
* Min Samples Leaf – minimum number of samples required to be in leaf node
* Min Weight Fraction Leaf
* Max Features
* Random State
* Min Impurity Decrease
* Class Weight
* Presort

Pros of Decision Trees – robustness to noise, tolerance for missing data, handling of irrelevant/redundant predictive attribute values, low computational cost, interpretability, fast run time, robust predictors

<https://medium.com/@mohtedibf/indepth-parameter-tuning-for-decision-tree-6753118a03c3>

AUC – area under the curve

Features that are important –

Grid Search … model ranking. Could find something that is equivalent accuracy with less layers,

FOR GRAPH VIZ

import osos.environ["PATH"] += os.pathsep + 'C:/Program Files (x86)/Graphviz2.38/bin/'#https://graphviz.gitlab.io/\_pages/Download/Download\_windows.html

FOR ROC CURVE

ylist = Y.values.astype('int64')ylistfrom sklearn.preprocessing import label\_binarizeybinary = label\_binarize(ylist, classes=[0, 1])n\_classes = ybinary.shape[1]X\_train2, X\_test2, y\_train2, y\_test2 = train\_test\_split(X,ybinary, test\_size=0.2)y\_score = cross\_val\_predict(clf, X, ybinary, cv=10 ,method='predict')from sklearn.preprocessing import label\_binarizeybinary = label\_binarize(ylist, classes=[0, 1])n\_classes = ybinary.shape[1]X\_train2, X\_test2, y\_train2, y\_test2 = train\_test\_split(X,ybinary, test\_size=0.2)y\_score = cross\_val\_predict(clf, X, ybinary, cv=10 ,method='predict')

#X1\_train, X1\_test, y1\_train, y1\_test#y\_score = classifier.fit(X\_train3, y\_train3).decision\_function(X\_test3)# Compute ROC curve and ROC area for each classfpr = dict()tpr = dict()roc\_auc = dict()for i in range(n\_classes):#y1\_test,yhat fpr[i], tpr[i], \_ = roc\_curve(ybinary, y\_score) roc\_auc[i] = auc(fpr[i], tpr[i])# Compute micro-average ROC curve and ROC areafpr["micro"], tpr["micro"], \_ = roc\_curve(ybinary.ravel(), y\_score.ravel())roc\_auc["micro"] = auc(fpr["micro"], tpr["micro"])#Plot of a ROC curve for a specific classplt.figure()lw = 2plt.plot(fpr[i], tpr[i], color='darkorange', lw=lw, label='ROC curve (area = %0.2f)' % roc\_auc[0])plt.plot([0, 1], [0, 1], color='navy', lw=lw, linestyle='--')plt.xlim([0.0, 1.0])plt.ylim([0.0, 1.05])plt.xlabel('False Positive Rate')plt.ylabel('True Positive Rate')plt.title('Receiver operating characteristic Spam')plt.legend(loc="lower right")plt.show()

IMPORT

from sklearn.metrics import confusion\_matriximport numpy as npimport seaborn as snsfrom sklearn.metrics import accuracy\_score, confusion\_matrix, roc\_auc\_score, auc, roc\_curvefrom sklearn.model\_selection import cross\_val\_scorefrom sklearn.model\_selection import cross\_val\_predict

\*\* Limited by Max Depth – selected

\*\* Sorted by Mean Test Score, best score with lowest depth/most shallow tree

\*\*\*\*DEPTH, Min Leaf,

\*\*here is a tree with most of the features from the .94, complex model, used the key elements from that the define the final model.