**IST- 5520 Data Science and Machine Learning with Python**

**Data Competition**

**By Austin Funcheon and Viraj Rane**

**Predictive Model for detecting Spam / Not spam emails.**

The dataset for detecting spam or not spam emails in predictive analytics is a classification problem. We have implemented supervised machine learning models and trained those on the training dataset. Amongst all the models, we chose Random Forest Classifier as the best model based on its optimal performance determined by the highest AUC score.

**Random Forest Classifier without hyper parameter tuning:**

1. from sklearn.ensemble import RandomForestClassifier, : we use ensemble method from sklearn package and import the Random Forest Classifier supervised ML model.
2. rf = RandomForestClassifier(n\_estimators=7, max\_features=9, random\_state=123) : n\_estimators are the number of decision trees that RF uses, max\_features are no. of predictors and random\_state takes random samples from the data, it makes code reproducible and does not change the result when used anytime later.
3. rf.feature\_importances\_ : returns all the important/ significant features(predictors) from the dataset.
4. feature\_importances=pd.DataFrame(rf.feature\_importances\_,index=train\_X\_scale.columns,columns=['importance']) :

AND

1. feature\_importances = feature\_importances.sort\_values('importance',ascending=False)

feature\_importances.plot(kind = 'bar',figsize=(15,5),title= ‘ ') : We select and plot/ display all the important features(predictors) with the .plot method in the form of a bar chart.

1. rf.fit(train\_X\_scale, train\_y) : Using .fit method to fit the training data in the RF model.
2. pred\_y\_rf = rf.predict(test\_X\_scale) : The .predict method is the probability prediction of the model given a new number of instances, in this case (test dataset).

**RF Classifier with hyper parameter tuning:**

1. parameters={'criterion':['gini','entropy'],'n\_estimators':[100,200,300,400,500],'max\_features':[1,3,5,7,9,13,15,17,20,25,30,40,50,57],'random\_state':[123]} : These are a set of hyper parameters that we choose to enhance the performance of the RF Classifier.

Gini, entropy: both are measure of impurity of a node. In a decision tree, a node having multiple classes is impure, whereas a node with one class is a pure node.

1. from sklearn.model\_selection import GridSearchCV : The GridSearch cross-validation is a HP tuning technique that helps loop through predefined HPs and fits the estimator(model) optimally on the training dataset.
2. grid\_rf = GridSearchCV(rf, parameters, scoring='roc\_auc', cv=10) : via GridSearchCV method we give parameter input and set model scoring/ evaluating method as ‘roc\_auc’.

cv=10 is the K-fold validation technique that defines the no. of cross-validations to be performed on the training data. It reserves a particular sample of dataset on which we do not train the model.

1. grid\_rf.best\_params\_ : Gives the best parameters from which it states that the model has the highest/ best performance with those parameters.
2. metrics.accuracy\_score(test\_y, pred\_y\_rf2) : It is the classification score. The function computes subset accuracy i.e., a set of labels predicted for a sample must match the corresponding set of labels.
3. metrics.roc\_auc\_score(test\_y, pred\_y\_rf2) : roc\_auc\_score is performance measurement for classification problems at various thresholds. ROC is the probability curve and AUC is the area under the probability curve i.e., a measure of separability. It tells us how much a ML model is capable of distinguishing between classes.