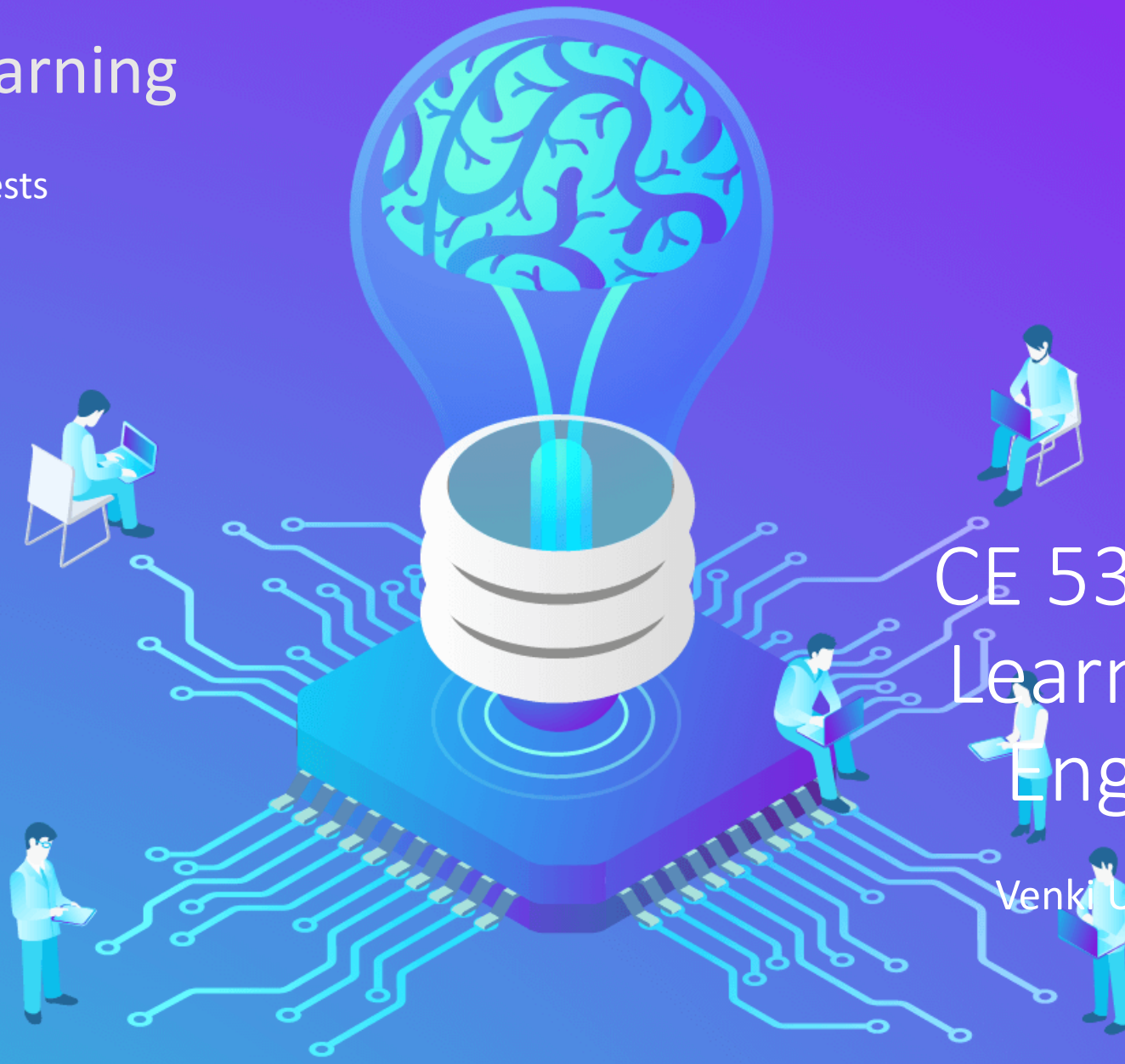
Machine Learning

Random Forests



CE 5331 Machine Learning for Civil Engineers

Venki Uddameri, Ph.D. , P.E.

Recap

- What is Machine Learning
- How is it useful for Civil Engineers
- Overview of Machine Learning Methods
- Linear Regression
  - Bivariate
  - Regression interpretation
  - Multivariate
- Logistic Regression
  - Maximum likelihood estimation
  - Regularization (introduction)
- Naïve Bayesian Classifier
  - What is it
  - What makes it naïve
  - Bayes theorem
  - Prior, likelihood and posterior
- K-Nearest Neighbor
  - How does the algorithm work
  - Why is it a lazy learner
  - How to do regression and classification
- Introduction to Decision Trees
  - Fundamentals
  - Information Gain, Entropy and Gini Index
  - ID3 algorithm
  - Classification and Regression Trees (CART)
  - Multi-Adaptive Regression Splines (MARS)

- Ensemble learners
  - Introduction
- Their benefits and drawbacks
- Simple (voting) ensemble learners
- Bagging and Pasting
- Generic bagging classifiers

Python – Introduction
Python – Functions
Python - Pandas
Python – np, scipy, statsmodels
Python – Scikit learn – linear, metrics
Python – Matplotlib, seaborn
Python – Mixed_Naive_Bayes
Python – scikit learn neighbors module
Python – scikit learn ensemble voting
Python – scikit learn bagging classifier

R – Classification and Regression Trees using rpart
R – Drawing trees using rpart.plot
R - Multiadaptive Regression Splines (MARS) using Earth Algorithm

Random Forests

Random Forests

- At a high level - Random Forests (RF) are an ensemble of decision trees based on bootstrap aggregation (bagging)
- They are based on the idea that a group of classifiers (ensemble) provide a better result than a single model
- They used simple voting or “average” value to make predictions
- They are based on the notion that a large number of uncorrelated classifiers can overcome each other’s deficiencies
- While RF models are broadly based on bagging they also use several other tricks to increase the diversity of models within a ensemble
- RF models trade bias for lower variance to improve their predictions

Random Forests

- RF algorithms use an ensemble of decision trees
  - Decision trees are referred to as unstable learners
  - Small changes in the dataset can significantly alter the structure of the decision tree
  - Use of Decision Trees help create a diverse set of learners
- RF algorithms typically use the same sample size as the training dataset
  - Data is not ‘chunked’ but sampled with replacement
  - Data diversity affects the tree structure
- Feature Randomness
  - Not all features are used every time
    - Each tree is constructed using a subset of features
  - The use of a subset of features for construction of each tree helps minimize correlation effects among trees

Random Forests – Hyperparameters

- Random Forests algorithms have Several sets of hyperparameters
  - Overall Hyperparameters
    - Number of trees
    - OOB evaluation or not
  - Hyperparameters to control bagging
    - Number of samples (typically the size of the training dataset)
    - Replacement = True (bootstrap)
    - Pasting can be used
      - Sample size has to be smaller than the training dataset
      - Samples are held constant
  - Hyperparameters to control feature selection
    - Number of features
      - Typically $\sqrt{(K)}$ or $log_2 K$
  - Hyperparameters to control trees
    - Quality of spitl (Gini index or entropy)
    - Depth of the tree
    - Minimum number of data for split
    - Other parameters controlling the error structure

The size of the OOB samples varies For each bootstrap simulation and depends upon the number of repeated samples

Random Forest – Impurity-based Variable Importance

- As Random Forest (RF) algorithm creates several trees with different features
  - The role of each feature in making predictions can be assessed
- The RF algorithm uses Gini Importance or Mean Decrease in Impurity (MDI) to identify important features
  - Gini Importance (or MDI) is a measure of how much loss of a fit (or accuracy) occurs when a variable is dropped
    - The larger the drop the more important the variable

Gini Importance can be computed in different ways

In scikit.learn:

total decrease in node impurity (weighted by the probability of reaching that node (which is approximated by the proportion of samples reaching that node)) averaged over all trees of the ensemble.

In R package randomforest

the actual decrease in node impurity is summed and averaged across all trees. (weighted by the number of samples it splits).

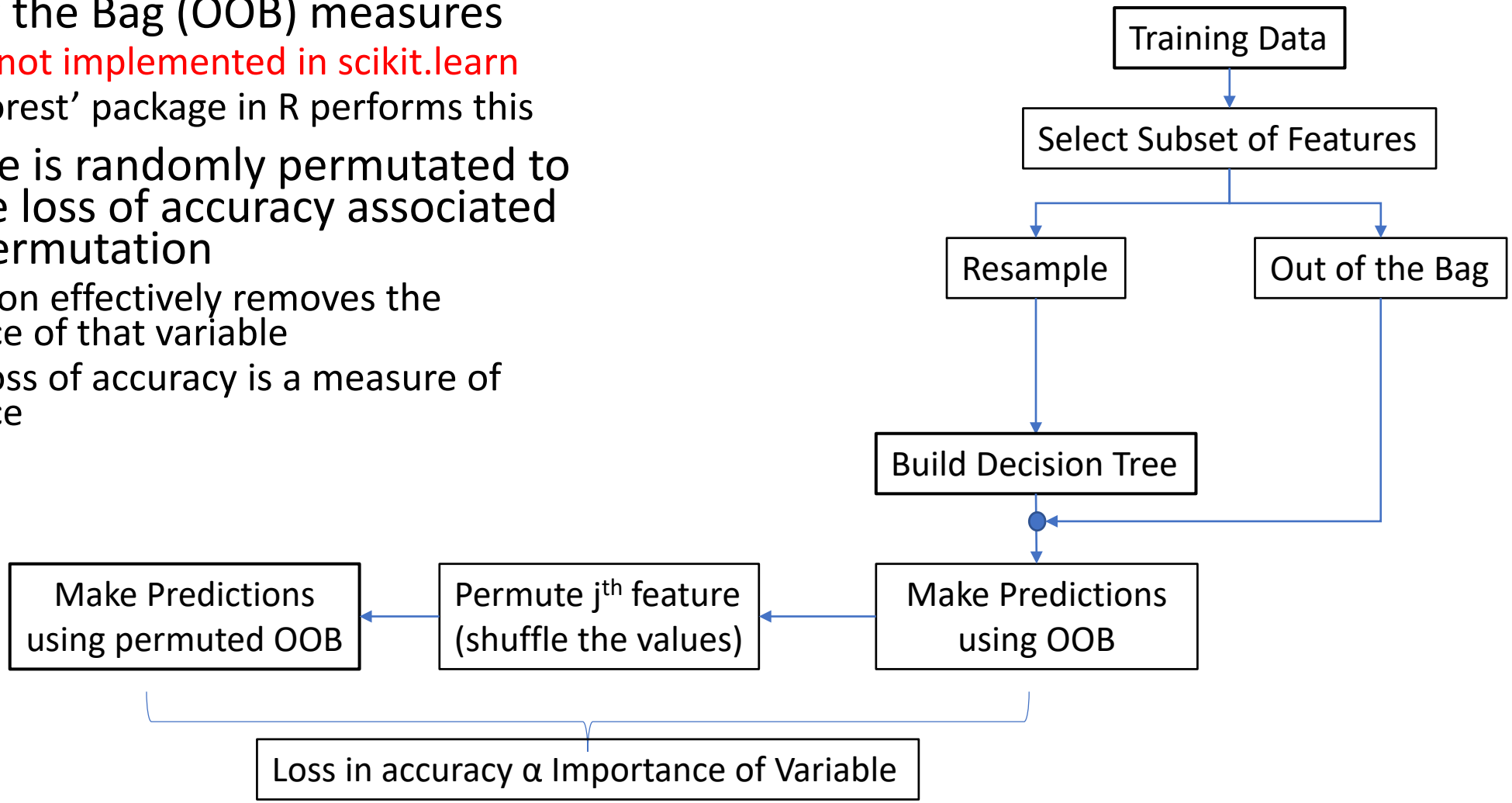
Variable selection will depend upon how Gini Importance (or MDI) is computed

Random Forest – Impurity-based Variable Importance

- Impurity based methods generally have a bias towards variables that have more number of categories
- Variable importance cannot discern the importance of correlated variables
  - Both variables exert similar influence on the output
  - If impurity is removed by one variable then there is not much impurity left to remove from the other variable
    - Incorrect to interpret that the second variable is not as important as the first
    - Random selection of features minimizes this effect but does not remove it completely

Random Forest – OOB - Variable Importance

- Variable Importance can also be computed using Out of the Bag (OOB) measures
  - Currently not implemented in scikit.learn
  - 'randomforest' package in R performs this
- Each variable is randomly permutated to measure the loss of accuracy associated with such permutation
  - Permutation effectively removes the importance of that variable
  - Average loss of accuracy is a measure of importance



Random Forest Implementation

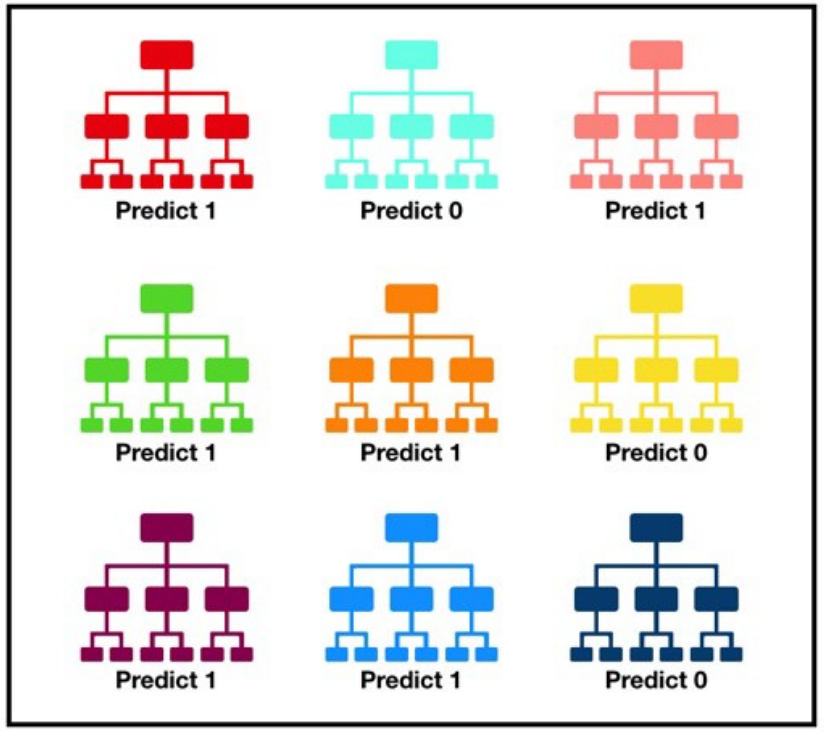
- Python scikit-learn library has implementations of random forests
  - Classifier – RandomForestClassifier in ensemble module
    - https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html#
  - Regression – RandomForestRegressor in ensemble module
    - https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html
- Variable Importance
  - *obj*.feature_importances_ method can be used to extract feature importances
  - Where obj is a Random Forest object (either a classifier or a regressor)
  - https://scikit-learn.org/stable/auto_examples/ensemble/plot_forest_importances.html#sphx-glr-auto-examples-ensemble-plot-forest-importances-py

Random Forest - Variants

- Extremely Randomized Trees
  - Variables are subsetted as usual
  - Instead of computing splits – Random splits are assigned to each variable
    - The best variable (that provides the highest predictive accuracy) is selected based on these randomly assigned splits
  - Adds greater variability to the modeling process
    - Helps reduce variance further but at the cost of bias

Random Forests

Bagging + Feature Randomness



Tally: Six 1s and Three 0s

Prediction: 1

Illustrative Example

- Predicting damage to culverts in Texas
- Create a Random Forest Classification Model to predict satisfactory and unsatisfactory states
- Use Gridsearch to identify optimal parameters for ‘max_depth’ and ‘n_estimators’

Satisfactory

Unsatisfactory

Code	Meaning	Description
9	Excellent	As new
8	Very Good	No problems noted.
7	Good	Some minor problems.
6	Satisfactory	Structural elements show some minor deterioration.
5	Fair	All primary structural elements are sound but may have minor section loss, cracking, spalling or scour.
4	Poor	Advanced section loss, deterioration, spalling or scour.
3	Serious	Loss of section, deterioration, spalling or scour has seriously affected primary structural components. Local failures are possible. Fatigue cracks in steel or shear cracks in concrete may be present.
2	Critical	Advanced deterioration of primary structural elements. Fatigue cracks in steel or shear cracks in concrete may be present or scour may have removed substructure support. Unless closely monitored it may be necessary to close the bridge until corrective action is taken.
1	Imminent Failure	Major deterioration or section loss present in critical structural components or obvious vertical or horizontal movement affecting structure stability. Bridge is closed to traffic but with corrective action may put back in light service.
0	Failed	Out of service, beyond corrective action.

Source: United States Department of Transportation. Recording and Coding Guide for the Structure Inventory and Appraisal of the Nation's Bridges. Washington, D.C., 1995, page 38.

Python Code

```
# Step 1: Load Libraries
import os
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestClassifier
from sklearn import metrics
import seaborn as sns
from matplotlib import pyplot as plt
```

```
# Step 2: Change working directory
dir = 'D:\\Dropbox\\000CE5333Machine Learning\\Week7EnsembleLEarning\\Code'
os.chdir(dir)

# Step 3: Read the dataset and add a variable
a = pd.read_csv('TXculvertdata.csv') # read our dataset
features = ['SVCYR','ADT','Reconst','PTRUCK'] # INPUT DATA FEATURES
X = a[features] # DATAFRAME OF INPUT FEATURES
SVCYR2 = a['SVCYR']**2  # Add SVCYR square to the dataset
X['SVCYR2'] = SVCYR2  # CALCULATE THE SQUARE OF AGE
Y = a['Culvert_Damage'] # ADD IT TO THE INPUT FEATURE DATAFRAME

# Step 4: Split into training and testing data
X_train,X_test,y_train,y_test=train_test_split(X,Y,test_size=0.30,
                                    random_state=10)
```

```
# Step 5: Perform Grid Search for Best Hyperparameters using 5-fold CV
gsc = GridSearchCV(
      estimator=RandomForestClassifier(),
      param_grid={'max_depth': range(3,7),
                  'n_estimators': (10, 50, 100, 1000)},cv=5, n_jobs=-1)
grid_result = gsc.fit(X_train, y_train)
best_params = grid_result.best_params_
```

```
#Step 6: Fit the RF model with best data and predict testing
rfr = RandomForestClassifier(max_depth=best_params["max_depth"],
                             n_estimators=best_params["n_estimators"],
                             random_state=False, verbose=False)
rfr.fit(X_train,y_train)
y_pred=rfr.predict(X_test)
```

```
# Step 7: Create a confusion Matrix and associated measures
cnf_matrix = metrics.confusion_matrix(y_test, y_pred)
cnf_matrix # y_test is going be rows (obs), y_pred (predicted) are cols

# Evaluate usng accuracy, precision, recall
print("Accuracy:",metrics.accuracy_score(y_test, y_pred)) # overall accuracy
print("Precision:",metrics.precision_score(y_test, y_pred)) # predicting 0 (Sat)
print("Recall:",metrics.recall_score(y_test, y_pred)) # predicting 1 (unsat)
```

Python Code

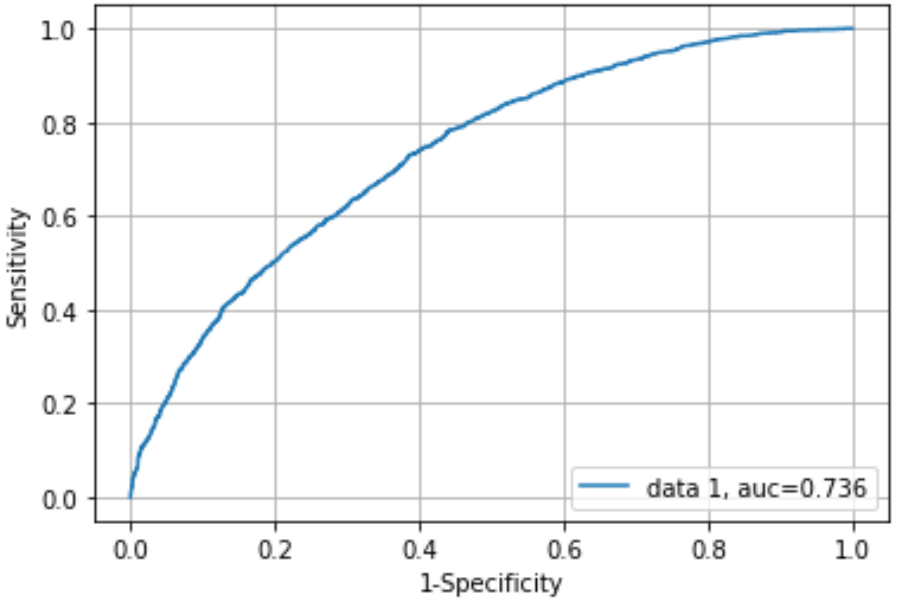
```
# Step 8: Plot ROC Curve
y_pred_proba = rfr.predict_proba(X_test)[::,1]
fpr, tpr, _ = metrics.roc_curve(y_test,  y_pred_proba)
auc = metrics.roc_auc_score(y_test, y_pred_proba)
plt.plot(fpr,tpr,label="data 1, auc="+str(round(auc,4)))
plt.legend(loc=4)
plt.xlabel('1-Specificity')
plt.ylabel('Sensitivity')
plt.grid()
plt.show()
```

```
# Step 9: Feature Importance
names = list(X.columns) # Get names of variables
imp = rfr.feature_importances_ # Obtain feature importance
impa = (names,imp) # Make a tuple
impadf = pd.DataFrame(impa) # Write to a dataframe
sns.set(style="whitegrid")
ax = sns.barplot(x=imp, y=names) # Make a barplot
ax.set(xlabel="Relative Importance")
```

Results

Random Forest – Confusion Matrix

	Predicted	
Obs	0	1
0	2697	808
1	1131	1350

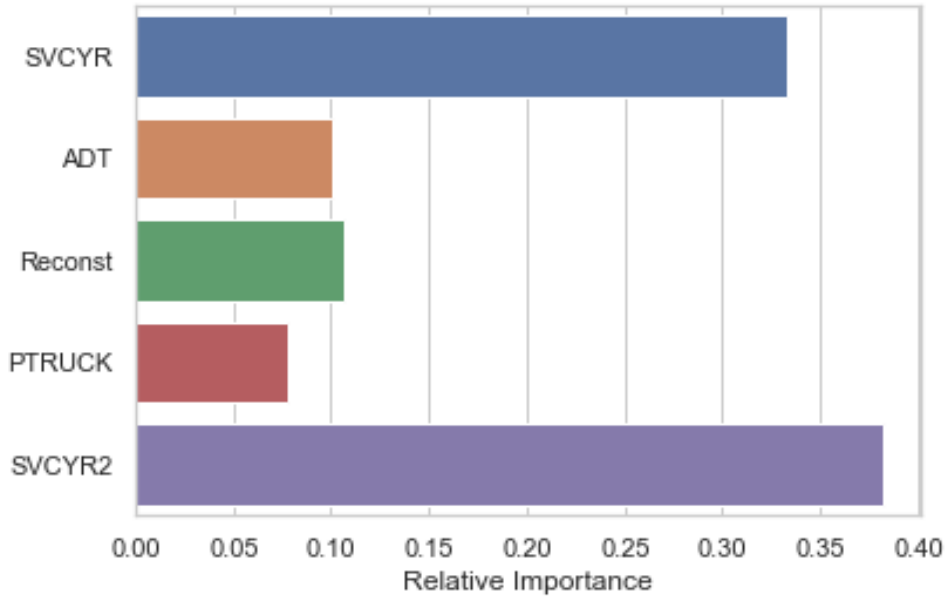


Random Forests

Accuracy: 0.676
Precision: 0.626
Recall: 0.544

Comparison to Other Models

Model	Accuracy
Logistic	0.6657
Naïve Bayes	0.6557
KNN	0.6236
Ensemble	0.6676



Random Forests – Closing Thoughts

- Typically a versatile algorithm that can compete in terms of classification and prediction accuracy with most other ML algorithms
- Not as interpretable as single tree models
  - Variable importance is however a good bonus of running this method
- The model is computationally expensive must easy to parallelize
- Python and R implementations have slight differences so at this point the results may not be fully reproducible even with same staring seed across platforms
  - Results should however be close

You Should Know

- What are random forests
- How Bagging and Feature Subsets improve predictions
- Hyperparameters of Random Forest Models
- How variable importance can be ascertained using RF models
  - Use of Gini Importance
  - Use of OOB estimates
- RF Variants
  - Extremely randomized trees
- How to implement Random Forests in Python
  - Scikit.learn.ensemble
  - Variable importance