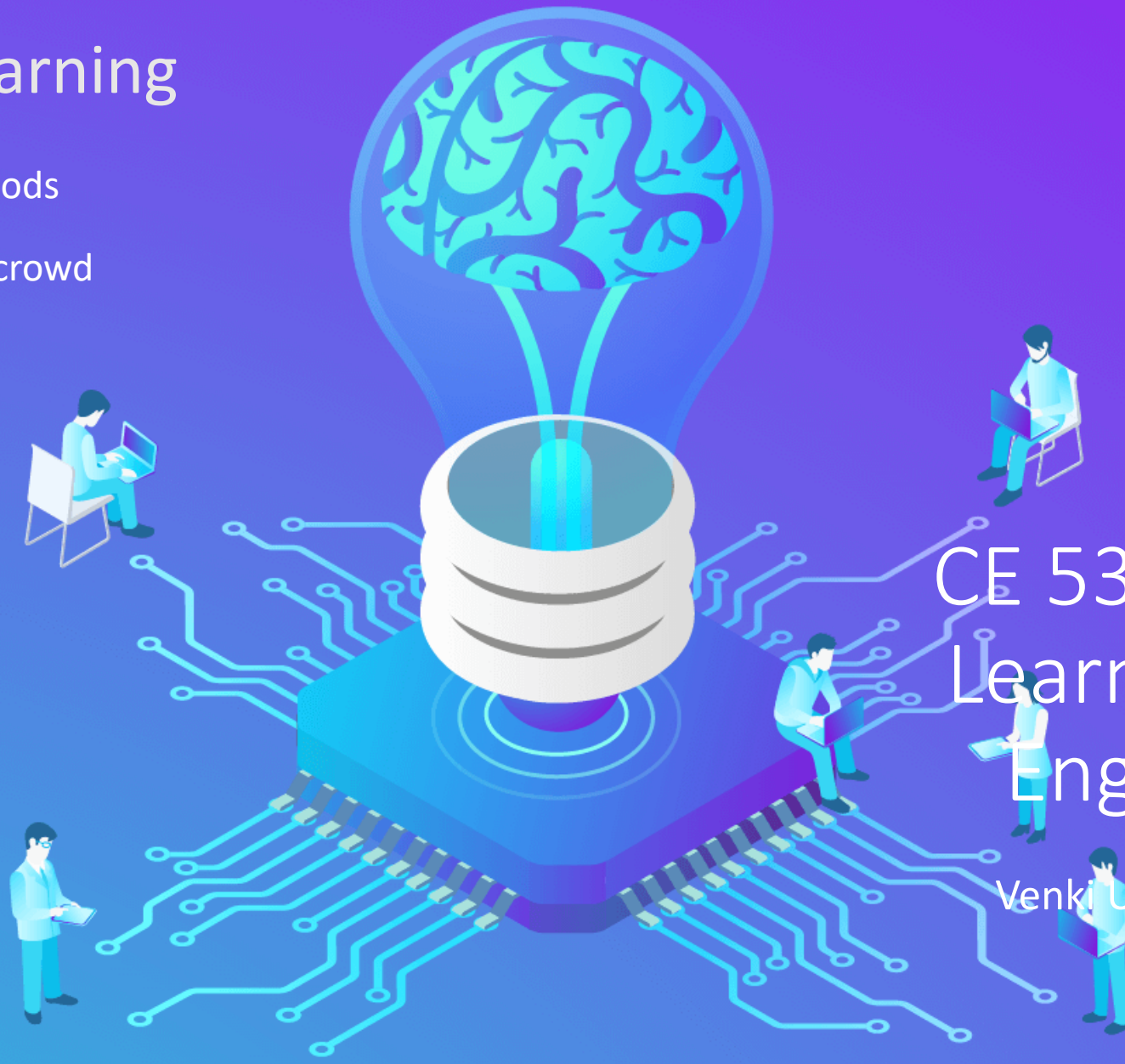
# Machine Learning

Bagging Methods

Wisdom of the crowd
Learner



# 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

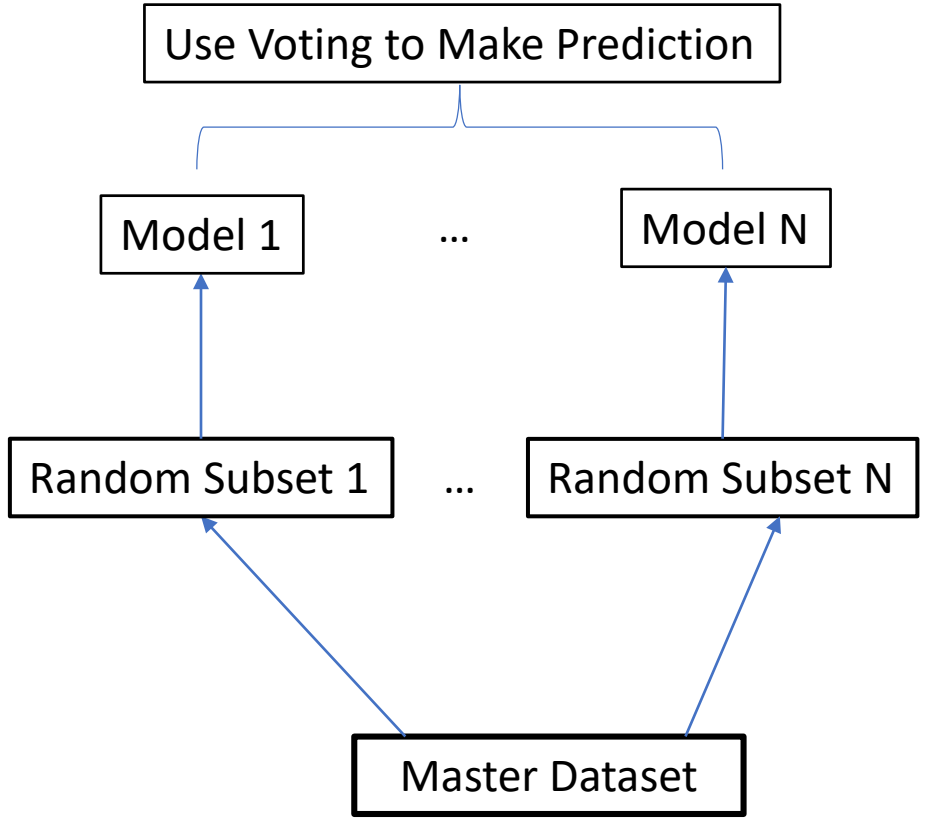
Python – Introduction
Python – Functions
Python - Pandas
Python – np, scipy, statsmodels
Python – Scikit learn – linear, metrics
Python – Matplotlib, seaborn
Python – Mixed_Naive_Bayes
Python – scikit learn neighbors module
Python – sckkit learn ensemble voting

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

Bagging Methods

# Bagging and Pasting Methods

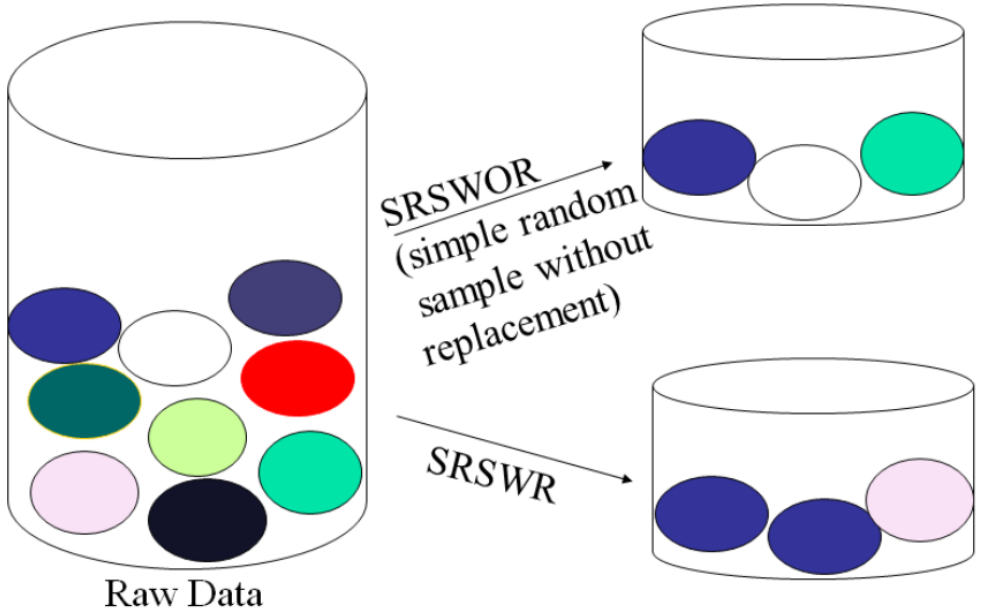
- Bagging and Pasting methods are a special class of ensemble methods
  - Same algorithm (technique) is used over all ensembles
  - Each ensemble model is built with a different dataset
- Bagging stands for ‘Bootstrap Aggregation’
  - The original dataset is ‘randomly’ sampled **with** replacement
- Pasting
  - The original dataset is ‘randomly’ sampled **without** replacement




Bagging methods are often referred to as the ‘Wisdom of the Crowd’ Learner

# Sampling with our Without Replacement

- Sampling with replacement
  - The maximum sample size is not restricted
  - Can be bigger than the original dataset
- Sampling with replacement
  - The maximum sample size = size of the dataset
- The data are typically assumed to be uniformly distributed
  - All elements have the same chance of being selected
- One could sample from a known distribution just as easily



# Univariate Sampling Illustrative - Example

- There are many ways to resample a univariate list
  - Random library
  - Numpy.random
- Sampling can be done with or without replacement
  - Probabilities can be assigned to each element of a list to increase (or decrease) their chances of being selected
- Sampling sequence can be repeated by setting the random seed to a constant number
  - Computers only generate pseudorandom numbers

```
import numpy as np
a = [1,2,3,4,5,10,11,12,25,36]
np.random.seed(10)
a_replace = list(np.random.choice(a,size=5,replace=True))
a_noreplace = list(np.random.choice(a,size=5,replace=False))
```

a_replace = [36, 5, 1, 2, 36]
a_noreplace = [1, 2, 25, 36, 11]

# Multivariate Sampling

- As most models have more than one input, it is common to resample rows of a data_frame (comprising of multiple columns or attributes)
- One approach is to create a list of row numbers
  - Row numbering in Python starts with zero
- Sample the row numbers list with or without replacement
  - This is a univariate list
- Create a new data_frame with selected row numbers
- Pandas library has a ‘sample’ function that makes the entire process very easy

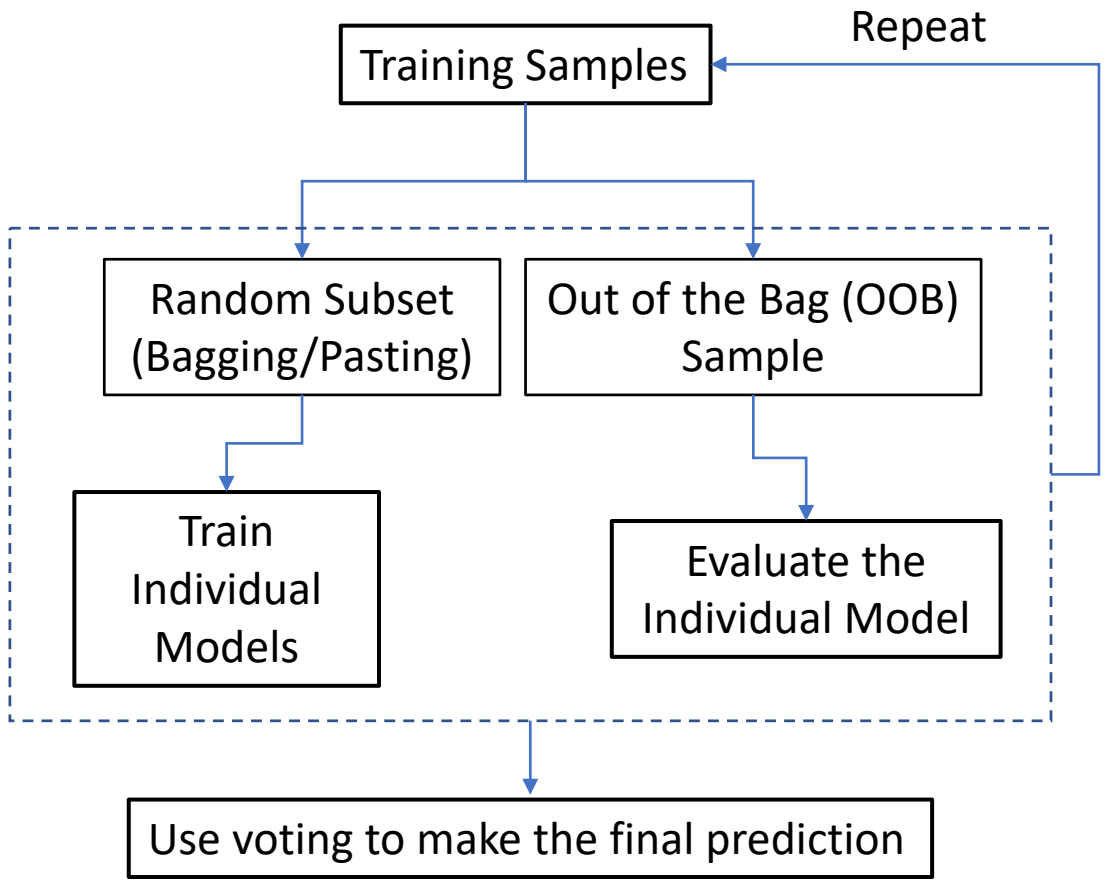
```
# Sampling rows of a dataframe or an array
import numpy as np
import pandas as pd
np.random.seed(10)
a = pd.read_csv('TXculvertdata.csv')
rows = a.shape[0] # number of rows
idx = list(range(rows))
smp = np.random.choice(idx,size=10,replace='true')
a_smp = a.iloc[smp,:]

# Using Pandas
a_smpd = a.sample(n=10,replace=True,random_state=10)
```

By default this is false

# Bagging / Pasting

- Generic bagging classifier (and a bagging regression) methods are available within scikit learn library
- Can be used with a suite of models available with scikit learn
- Can do both bagging and pasting by selecting sampling type
- Can also do out of the bag (oob) evaluation




Soft voting is typically used in classification
Average value across ensemble is used for regression

Note: OOB evaluation is different from independent sampling that is done using the ‘testing’ dataset

# Illustrative Example

- Predicting damage to culverts in Texas
- Create a Bagging LR model with 500 estimators using a max. sample of 500 samples

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.

# Illustrative Example

```
# Step 1: Load Libraries
import os
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression # Logistic regression
import sklearn.ensemble as ens
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
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)

baglr = ens.BaggingClassifier(LogisticRegression(),n_estimators=500,
                    max_samples=500,bootstrap='true',n_jobs=-1,
                    oob_score='true') # instantiate the object
baglr.fit(X_train,y_train) # Fit the model
baglr.oob_score_  # oob score
y_pred = baglr.predict(X_test) # make predictions
```

```
# Step 5: Create a confusion Matrix
cnf_matrix = metrics.confusion_matrix(y_test, y_pred)
cnf_matrix # y_test is going be rows (obs), y_pred (predicted) are cols

# Step 6: 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)
```

```
# Step 6: ROC Curve
y_pred_proba = baglr.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()
```

# Results

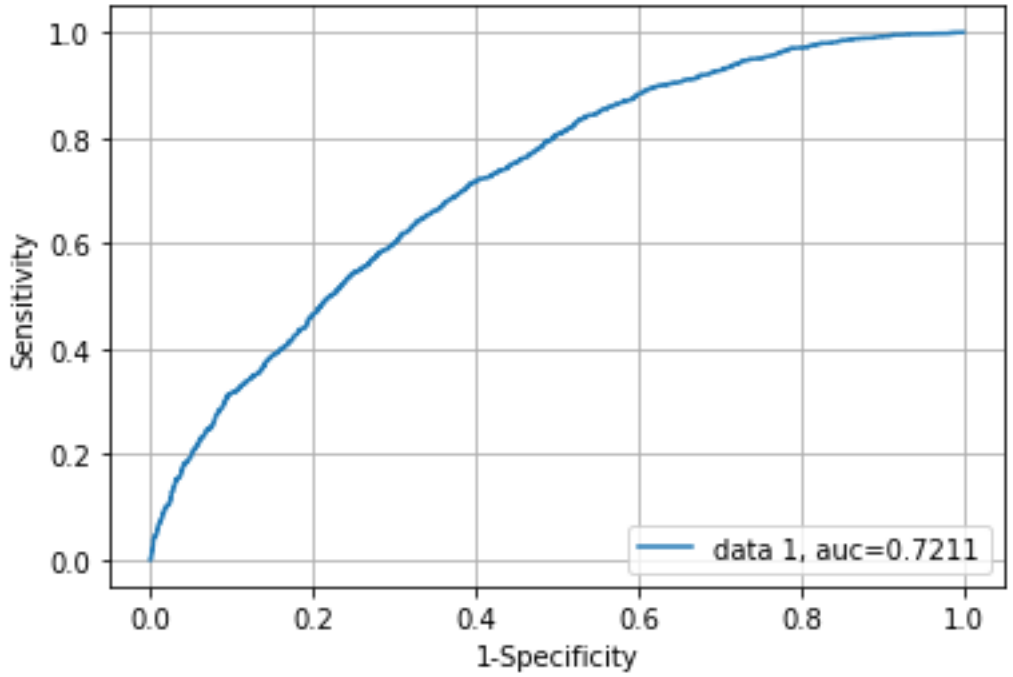
Bagging LR

| | Predicted | |
|---|---|---|
| Obs | 0 | 1 |
| 0 | 2759 | 746 |
| 1 | 1272 | 1209 |

Single LR

| | Predicted | |
|---|---|---|
| Obs | 0 | 1 |
| 0 | 2716 | 789 |
| 1 | 1230 | 1251 |

Bagging LR - AUC




Bagging LR did not improve much over single LR model

Tuning of hyperparameters – Number of ensembles; Number of samples needs to be evaluated

# Bagging – Some Thoughts

- Bagging approaches leads to a lower variance learner
  - Variance of an average is smaller than individual variances
  - However leads to an increase in bias
- Bagging may not necessarily improve predictions over the best individual learner within the ensemble
  - This happens when the learners within the ensemble are stable learners
  - Stable learners - Models undergo little change in output predictions even with significant changes in the input dataset
    - Regression models are often stable as they are controlled by extreme points (leverage) and the function does not change substantially unless these data are removed
- Bagging works best with unstable (high variance) learners
  - Decision trees are unstable as they depend upon the data for splits

# You should know

- What is bagging and pasting
- Differences between sampling with and without replacement
- Why and When does Bagging Work?
- How implement bagging and pasting in Python
  - Scikit learn ensemble module