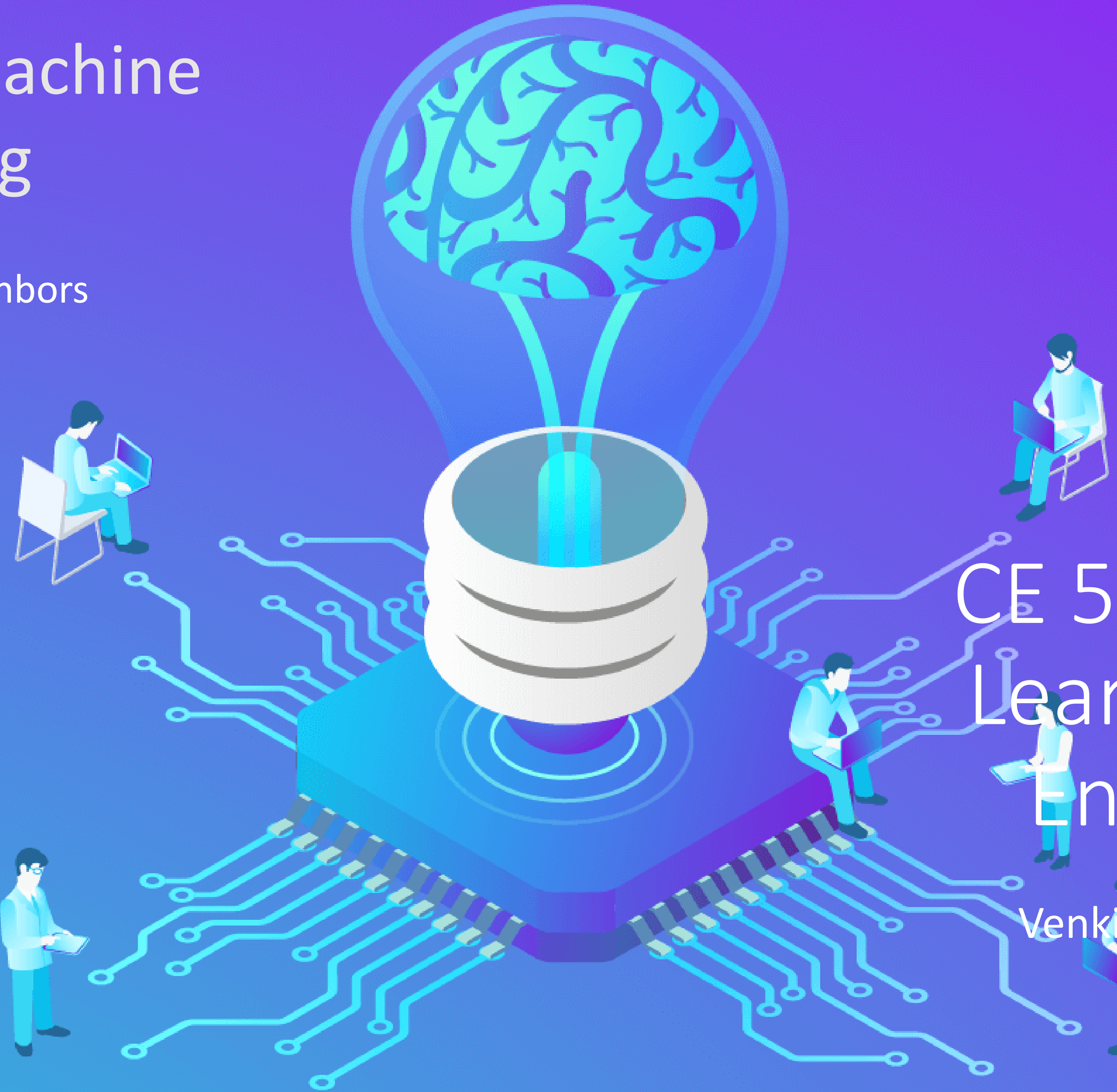
Python for Machine Learning

K Nearest Neighbors



CE 5331 Machine Learning for Civil Engineers

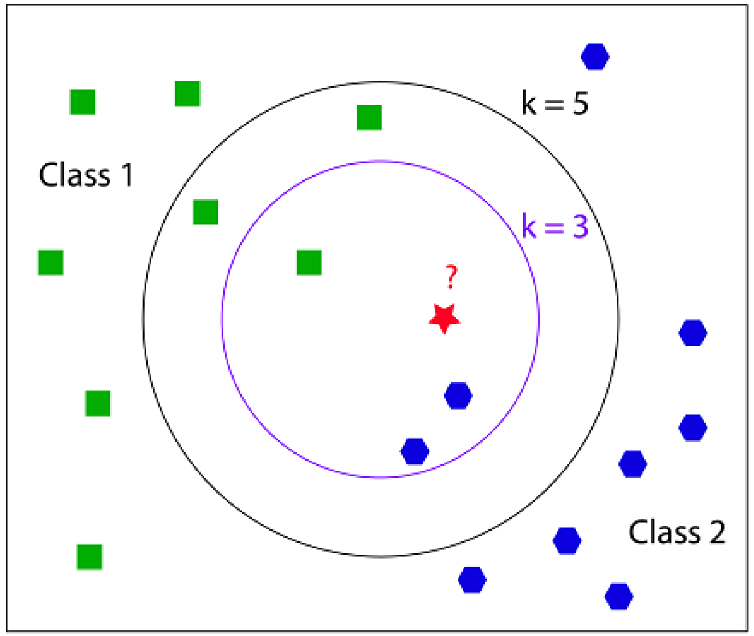
Venki Uddameri, Ph.D. , P.E.

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 Bayes Classifier

Python – Introduction
Python – Functions
Python - Pandas
Python – np, scipy, statsmodels
Python – Scikit learn – linear, metrics
Python – Matplotlib, seaborn
Python – Scikit learn – Naïve Bayes
Python – mixed_naive_bayes

In this module we shall look at Lazy and Easy Learners



K-Nearest Neighbor Algorithm

- K-Nearest Neighbor (KNN) is another easy to use ML algorithm
- It falls in the category of ‘memorization’ learning algorithms
    - Also called a lazy learner
        - Little to no training phase
- It is based on the idea that similar items are closer to each other than dissimilar items
- The proximity need not be in space or time but in relation to other features
    - Independent variables

KNN Algorithm

- KNN algorithm is largely used for classification but can also be used for regression problems as well
  - Classification – Output variable is discrete
  - Regression - Output variable is continuous

- KNN algorithm can also be used for unsupervised classification as well
  - K-mean or K-medoid clustering
  - Cluster the data into several groups
  - Number of clusters may be specified a priori
    - Can be obtained via optimization

For regression problems the output is the weighted average of K-Nearest Neighbors

The value of K in KNN has to be specified a priori or obtained via optimization

In classification problems the category is picked based on the majority vote of K nearest neighbors

Odd values of K are recommended to avoid ties

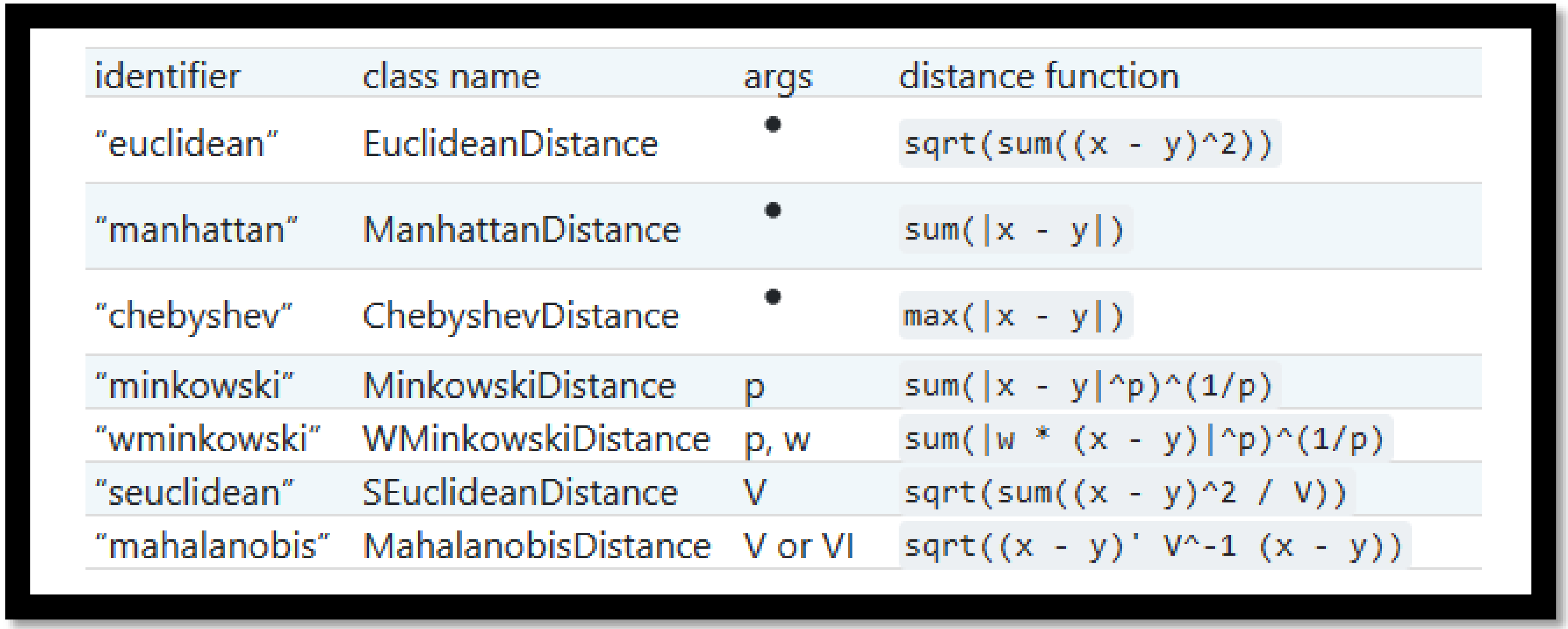
Note that KNN is an intuition based algorithm rather than a mathematical one

KNN Algorithm – Distance Measures

- The measurement of distance is fundamental to the implementation of the KNN algorithm
- The KNN algorithm is sensitive to the distance metric chosen
- There are many distance metrics that can be used
  - Depends upon whether the data is real, integer or real-valued vector spaces
- Distance metric is a ‘hyperparameter’ when implementing KNN algorithm

Distance Metrics

<u>Metrics Based on Real Valued Vectors</u>

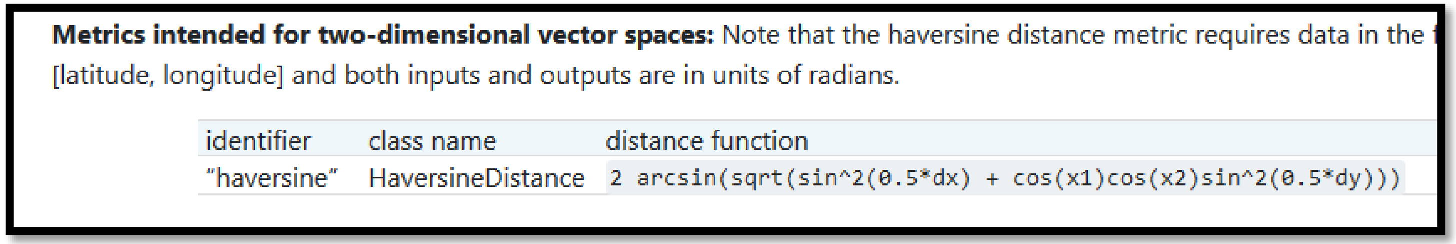


identifier	class name	args	distance function
"euclidean"	EuclideanDistance	•	`sqrt(sum((x - y)^2))`
"manhattan"	ManhattanDistance	•	`sum(|x - y|)`
"chebyshev"	ChebyshevDistance	•	`max(|x - y|)`
"minkowski"	MinkowskiDistance	p	`sum(|x - y|^p)^(1/p)`
"wminkowski"	WMinkowskiDistance	p, w	`sum(|w * (x - y)|^p)^(1/p)`
"seuclidean"	SEuclideanDistance	V	`sqrt(sum((x - y)^2 / V))`
"mahalanobis"	MahalanobisDistance	V or VI	`sqrt((x - y)' V^-1 (x - y))`

The Euclidean Distance is the most common of these metrics
These metrics are also often used with integer vectors as well

Metrics for Two-Dimensional Vector Spaces

- For Latitudes and Longitudes the Haversine Distance can be taken
  - Shortest distance between two points on the earth with a latitude and a longitude

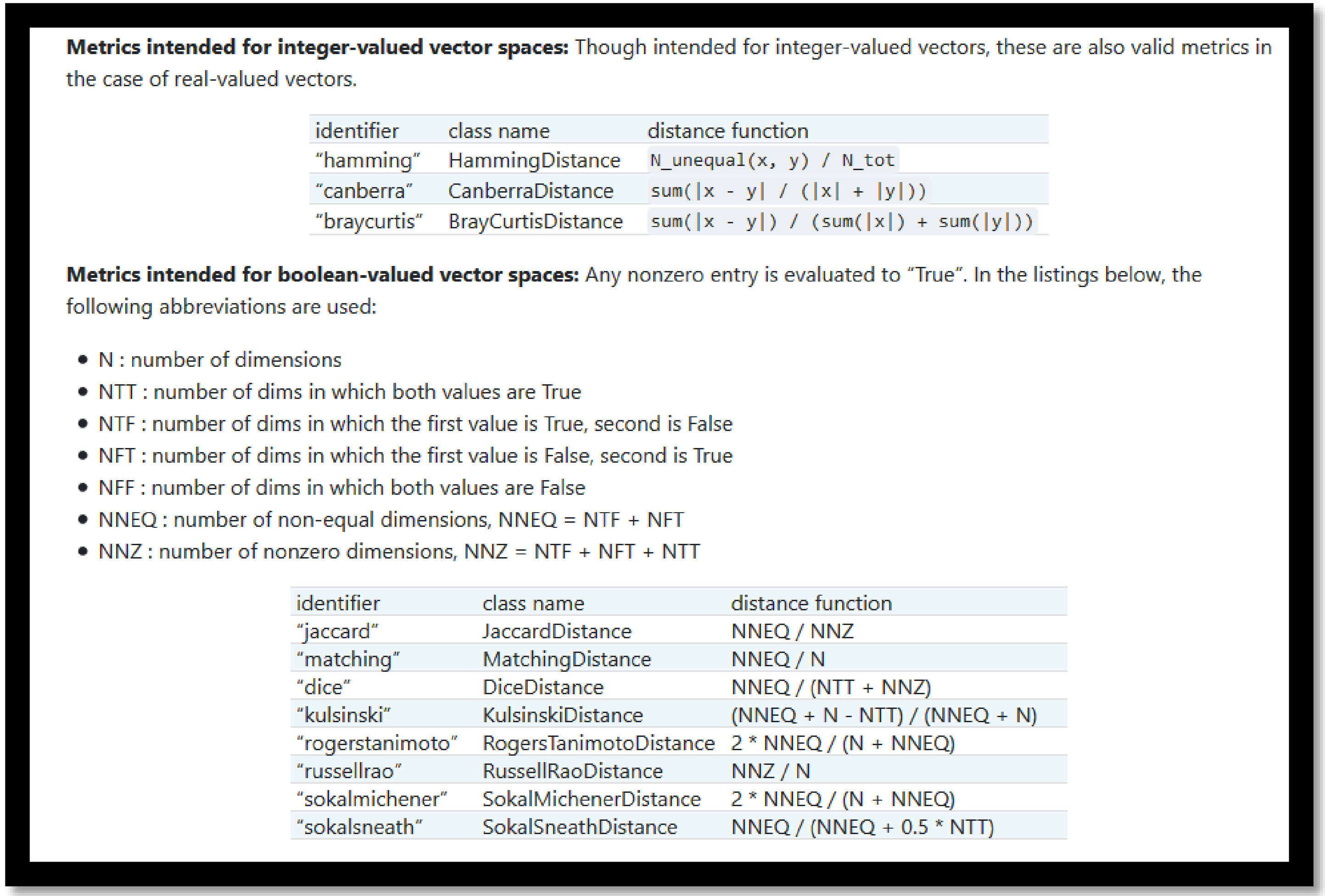


Metrics intended for two-dimensional vector spaces: Note that the haversine distance metric requires data in the
[latitude, longitude] and both inputs and outputs are in units of radians.

identifier	class name	distance function
"haversine"	HaversineDistance	`2 arcsin(sqrt(sin^2(0.5*dx) + cos(x1)cos(x2)sin^2(0.5*dy)))`

Greenland
Iceland
Finland
Sweden
Norway
United Kingdom
Canada
Poland
Germany
Ukraine
France
Kazakhstan
Italy
Spain
United States
North Atlantic Ocean
1
2
Iraq
Iran
Afghanistan
Pakistan
Algeria
Libya
Egypt
Saudi Arabia
India
Mexico
Mali
Niger
Sudan
Chad
Nigeria
Ethiopia
Venezuela
Colombia
Kenya
DR Congo

Metrics for Integer and Boolean Variables



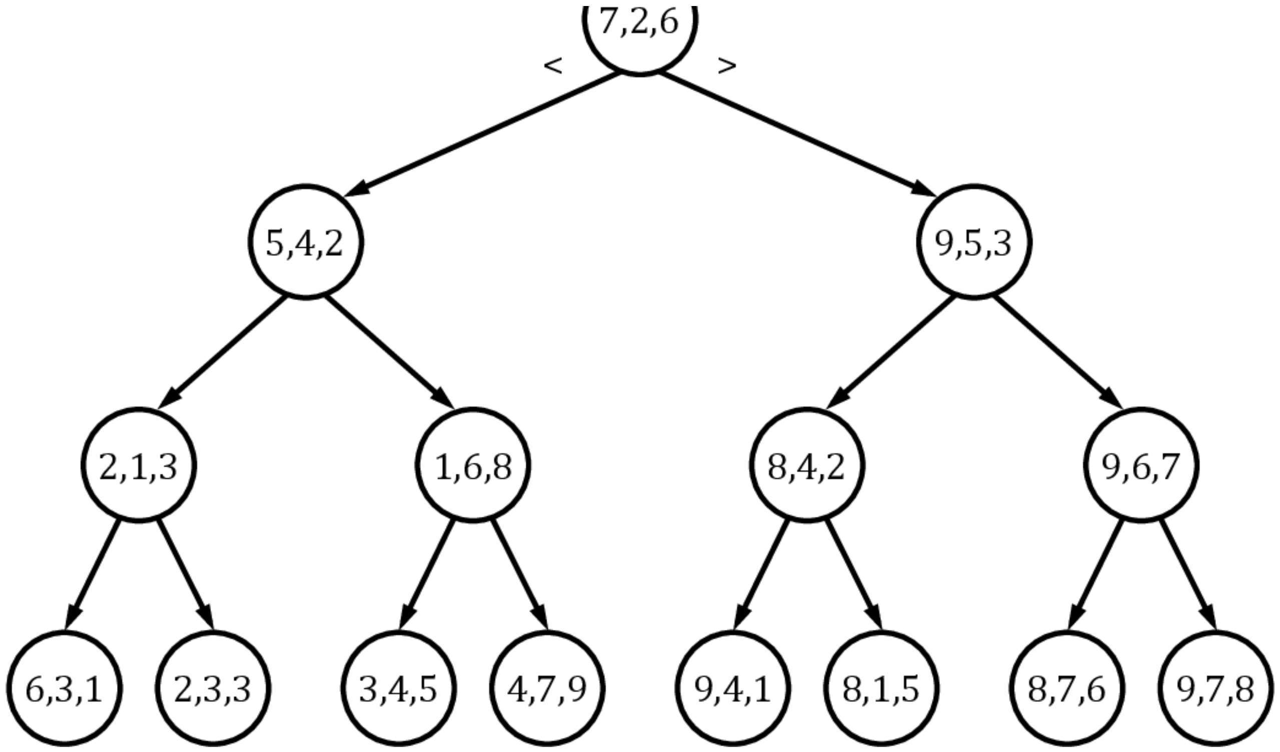
Metrics intended for integer-valued vector spaces: Though intended for integer-valued vectors, these are also valid metrics in the case of real-valued vectors.

identifier	class name	distance function
“hamming”	HammingDistance	`N_unequal(x, y) / N_tot`
“canberra”	CanberraDistance	`sum(|x - y| / (|x| + |y|))`
“braycurtis”	BrayCurtisDistance	`sum(|x - y|) / (sum(|x|) + sum(|y|))`

Metrics intended for boolean-valued vector spaces: Any nonzero entry is evaluated to “True”. In the listings below, the following abbreviations are used:

- N : number of dimensions
- NTT : number of dims in which both values are True
- NTF : number of dims in which the first value is True, second is False
- NFT : number of dims in which the first value is False, second is True
- NFF : number of dims in which both values are False
- NNEQ : number of non-equal dimensions, NNEQ = NTF + NFT
- NNZ : number of nonzero dimensions, NNZ = NTF + NFT + NTT

identifier	class name	distance function
“jaccard”	JaccardDistance	NNEQ / NNZ
“matching”	MatchingDistance	NNEQ / N
“dice”	DiceDistance	NNEQ / (NTT + NNZ)
“kulsinski”	KulsinskiDistance	(NNEQ + N - NTT) / (NNEQ + N)
“rogerstanimoto”	RogersTanimotoDistance	2 * NNEQ / (N + NNEQ)
“russellrao”	RussellRaoDistance	NNZ / N
“sokalmichener”	SokalMichenerDistance	2 * NNEQ / (N + NNEQ)
“sokalsneath”	SokalSneathDistance	NNEQ / (NNEQ + 0.5 * NTT)



- Finding nearest neighbors quickly is an active area in computer science
- There are many algorithms available
  - Brute Force
    - Calculate all distances
    - For N neighbors in D dimensions the algorithm – O[DN^2]
    - Competitive for small datasets
  - KD Tree algorithm
    - Use a tree based algorithm to reduce computations
    - Use aggregate distance measures
    - If d(A,B) >>> 0 and d(B,C) ~ 0 THEN A and C are also far apart (not nearest neighbors)
    - Time varies from O[Dlog(N)] to O[DN]

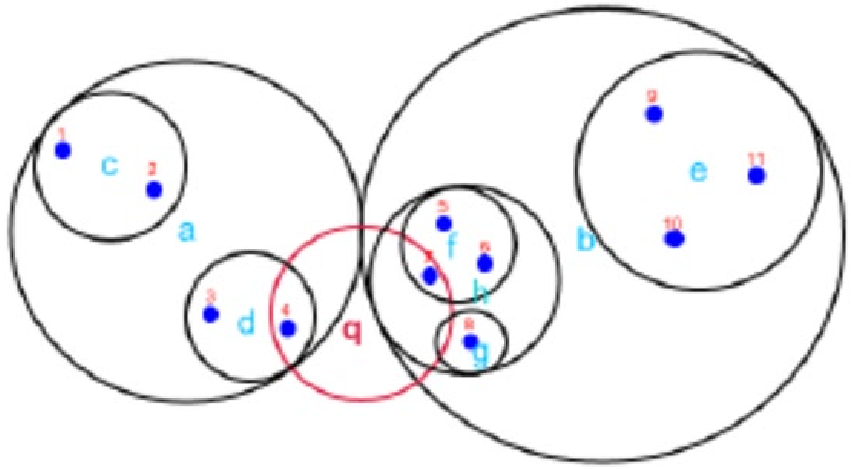
Implementing KNN

https://scikit-learn.org/stable/modules/neighbors.html

Implementing KNN Algorithms

- Ball Tree Algorithm
  - Overcomes the limitation of KD trees in higher dimensions
  - While KD tree uses cartesian distances ball tree uses nested hyperspheres
  - Tree construction is costly (time consuming) but helps speed up the search process
  - Recursively partition the data using the centroid C and radius R
    - A datapoints lie within at least one sphere
  - The distance from a test point to the centroid is sufficient determine whether the point lies within the sphere
    - Based on triangle rule

$$|x + y| \leq |x| + |y|$$



Implementation

- KNN can be implemented easily both in R and Python
- Scikit learn has tools for both classification and regression
  - https://scikit-learn.org/stable/modules/neighbors.html#nearest-neighbors-classification
  - https://scikit-learn.org/stable/modules/neighbors.html#nearest-neighbors-regression
- You can access the nearest neighbor algorithms using from sklearn import neighbors
- R also provides KNN classification
  - Package class function knn()

Illustrative Example

- Predicting damage to culverts in Texas
- Predict Binary classes (satisfactory, unsatisfactory)
- Predict all classes

Satisfactory

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.

Unsatisfactory

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.

Data

- Data was taken from Federal Highway Administration (FHWA)
  - https://www.fhwa.dot.gov/bridge/nbi/ascii.cfm
  - Year 2018 data release (updated 4/22/2019)
  - Downloaded csv file and cleaned up the dataset
- Previous studies indicate
  - Reconstruction Record (1 = Reconstructed; 0 = No Reconstruction)
  - Age (Inspection Date – (re)Construction Year)
  - Age^2
  - ADT (Average Daily Traffic per lane)
- The % of Truck Traffic on the Culvert could also be important
  - PTRCK
    - A measure of higher loads → greater damage potential

Culvert type could also be important but not considered here due to lack of data

Lu, Pan, Hao Wang, and Denver Tolliver. "Prediction of Bridge Component Ratings Using Ordinal Logistic Regression Model." *Mathematical Problems in Engineering* 2019 (2019). Available online: https://www.hindawi.com/journals/mpe/2019/9797584/

Python Code

- Step 1: Import Libraries
  - Usual suspects
  - import mixed_naive_bayes as mnb
- Step 2: Change Working Directory
  - Use os.chdir
- Step 3: Read the data
  - Use pandas
  - Subset the required variables
  - Add other variables as necessary
- Step 4: Split the data into training and testing
  - Use sklearn train_test_split
  - Use the same seed as before to ensure the same split
- Step 5: Train the model
  - Create a model object
  - Fit the model
  - Make predictions
  - Use mixed_naive_bayes library
- Step 6: Evaluate the model (testing data)
  - Predict testing data
  - Contingency table
  - Accuracy, precision, recall
  - ROC – AUC metric
  - Use sklearn metrics

Step 1: Import Libraries

```
# Import libraries
import os
import numpy as np
import pandas as pd
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn import metrics
import seaborn as sns
from matplotlib import pyplot as plt
```

Step 2: Change Working Directory

```
# Change Working Directory
os.chdir('D:/Dropbox/000CE5333Machine Learning/Week5-LazyEasyLearners/Code')
```

Step 3: Read the Data

```
# 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
```

You can compare the Naïve Bayes Model to the Results from the Logistic Regression Model

Python Code

Step 4: Split training and testing datasets

```
# 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: Train the Model

```
# Naive Bayes Model.  Note Reconst is a categorical variable X[2]
neigh = KNeighborsClassifier(n_neighbors=3)
neigh.fit(X_train, y_train)
```

Using 3 neighbors (tried 5 and 7
but 3 gave better results for recall

Other than step 5 and function import in step 1
The code is similar to the one used for LR and NB

Step 6a: Predict Testing Data

```
# Predict testing data
y_pred=neighpredict(X_test) # predict testing data
yprob = neigh.predict_proba(X_test) #output probabilities
```

Step 6b: Create a Contingency Table

```
# Perform evaluation using contingency table
# 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 6c: Compute Accuracy, Precision, Recall

```
# 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 6d: Create AUC curve

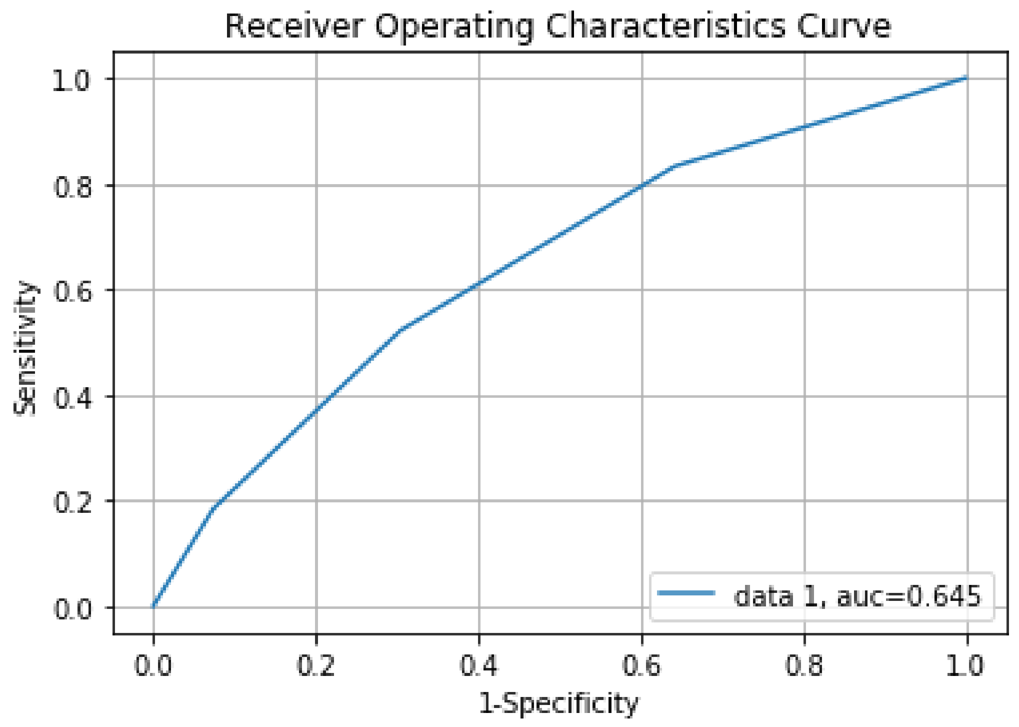
```
# ROC Curve
y_pred_proba = clf.predict_proba(X_test)[::,1]
fpr, tpr, _ = metrics.roc_curve(y_test,  y_pred_proba)
auc = metrics.roc_auc_score(y_test, y_pred_proba)
plt.plot(fpr,tpr,label="data 1, auc="+str(round(auc,4)))
plt.legend(loc=4)
plt.xlabel('1-Specificity')
plt.ylabel('Sensitivity')
plt.grid()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

Results

Contingency Table

		Predicted	
		0	1
Observed	0	2437	1068
	1	1185	1296

Metric	Value	Remarks
Accuracy	0.624	How well both 0 and 1 are predicted
Precision	0.548	How well 0 states are predicted
Recall	0.522	How well 1 states are predicted



Accuracy and Precision is slightly worse than LR and NB models

Recall is better than LR model, comparable to NB model (very slight improvement)

Model predicts the damage state better than the LR model

You should KNow

- What is KNN model
- The difference between supervised KNN models and unsupervised Nearest Neighbor clustering techniques
- How KNN performs classification
  - Majority vote
- How KNN performs regression
  - Weighted average as the prediction
- Different types of distance measures
  - Vector, integer and Boolean datasets
- Algorithms for calculating Nearest Neighbors
  - Brute force
  - KD tree
  - Ball Tree Algorithms

Why is KNN called a lazy Learner?

The learning mechanism of KNN