CHAPTER 2

Unsupervised Learning

(Unsupervised Learning with Python)

Content

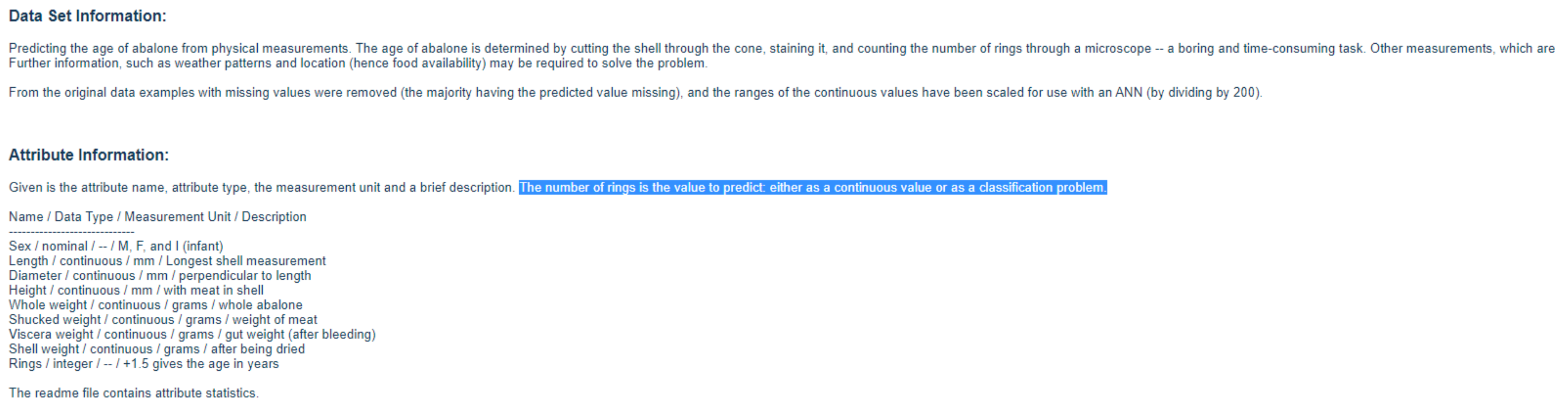
Python

Cluster Analysis for two real-world datasets

- Abalone Data Set
- Raisin Data Set

Abalone Data Analysis

- Downloaded from https://archive.ics.uci.edu/ml/datasets/abalone



Data Set Information:

Predicting the age of abalone from physical measurements. The age of abalone is determined by cutting the shell through the cone, staining it, and counting the number of rings through a microscope -- a boring and time-consuming task. Other measurements, which are
Further information, such as weather patterns and location (hence food availability) may be required to solve the problem.

From the original data examples with missing values were removed (the majority having the predicted value missing), and the ranges of the continuous values have been scaled for use with an ANN (by dividing by 200).

Attribute Information:

Given is the attribute name, attribute type, the measurement unit and a brief description. The number of rings is the value to predict: either as a continuous value or as a classification problem.

Name / Data Type / Measurement Unit / Description

Sex / nominal / -- / M, F, and I (infant)
Length / continuous / mm / Longest shell measurement
Diameter / continuous / mm / perpendicular to length
Height / continuous / mm / with meat in shell
Whole weight / continuous / grams / whole abalone
Shucked weight / continuous / grams / weight of meat
Viscera weight / continuous / grams / gut weight (after bleeding)
Shell weight / continuous / grams / after being dried
Rings / integer / -- / +1.5 gives the age in years

The readme file contains attribute statistics.

```
################################
################################
import pandas as pd #Import pandas module
import matplotlib.pyplot as plt #Import the visualization module
from sklearn.cluster import KMeans #Import K-means module
from sklearn.metrics import confusion_matrix #Import confusion matrix module
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import completeness_score
from sklearn import metrics
import numpy as np
from sklearn.mixture import GaussianMixture #Import GMM module
from sklearn.cluster import AgglomerativeClustering

n_clusters = 15

data = pd.read_csv('abalone.csv') #Load the data file
X = pd.get_dummies(data.iloc[:,:-1],drop_first=True)
#Apply K-means to dataset
kmeans = KMeans(n_clusters=n_clusters).fit(X) #perform K-means clustering with number of clusters = 3
centroids = kmeans.cluster_centers_ #Extract the cluster centroids
labels = kmeans.labels_ #Extract the labels of clusters
label_encoder = LabelEncoder()
# results = confusion_matrix(label_encoder.fit_transform(data.iloc[:,-1]),labels)
# print(results)
acc = metrics.completeness_score(label_encoder.fit_transform(data.iloc[:,-1]),labels)
print(acc)


###Apply hierarchical clustering to the dataset
clusters = AgglomerativeClustering(n_clusters=n_clusters).fit(X) #perform K-means clustering with number of clusters = 3
labels = clusters.labels_ #Extract the labels of clusters
label_encoder = LabelEncoder()
results = confusion_matrix(label_encoder.fit_transform(data.iloc[:,-1]),labels)
# print(results)
acc = metrics.completeness_score(label_encoder.fit_transform(data.iloc[:,-1]),labels)
print(acc)

###Apply GMM to the dataset
gmm = GaussianMixture(n_components=n_clusters).fit(X)
labels = gmm.predict(X)
label_encoder = LabelEncoder()
# results = confusion_matrix(label_encoder.fit_transform(data.iloc[:,-1]),labels)
# print(results)
acc = metrics.completeness_score(label_encoder.fit_transform(data.iloc[:,-1]),labels)
print(acc)


#Apply Ensemble clustering to Ecoli dataset
no_samples = data.shape[0]
no_estimators = 50
#Declare the weight of each vote
vote = 1/no_estimators
#co_association matrix is no_estimators X no_estimators (no_estimators patterns)
co_association = np.zeros((no_samples, no_samples))

#for each of your estimators
for est in range(no_estimators):
    #fit the data and grab the labels
    kmeans = KMeans(n_clusters=30,init='random',n_init=1).fit(X)
    labels = kmeans.labels_
    #find all associations and transform it into a numpy array
    res = [[int(i == j) for i in labels] for j in labels]
    res = np.array(res)
    #Vote and update the co_association matriz
    res = res * vote
    co_association = co_association + res
distance_matrix = 1-co_association
cluster = AgglomerativeClustering(n_clusters=n_clusters, affinity='euclidean',compute_distances=True).fit(distance_matrix)
labels = cluster.labels_ #Extract the labels of clusters
label_encoder = LabelEncoder()
# results = confusion_matrix(label_encoder.fit_transform(data.iloc[:,-1]),labels)
# print(results)
acc =  metrics.completeness_score(label_encoder.fit_transform(data.iloc[:,-1]),labels)
print(acc)
```

Abalone Data Analysis

- Apply K-means, Hierarchical clustering, GMM and Ensemble Clustering to the data

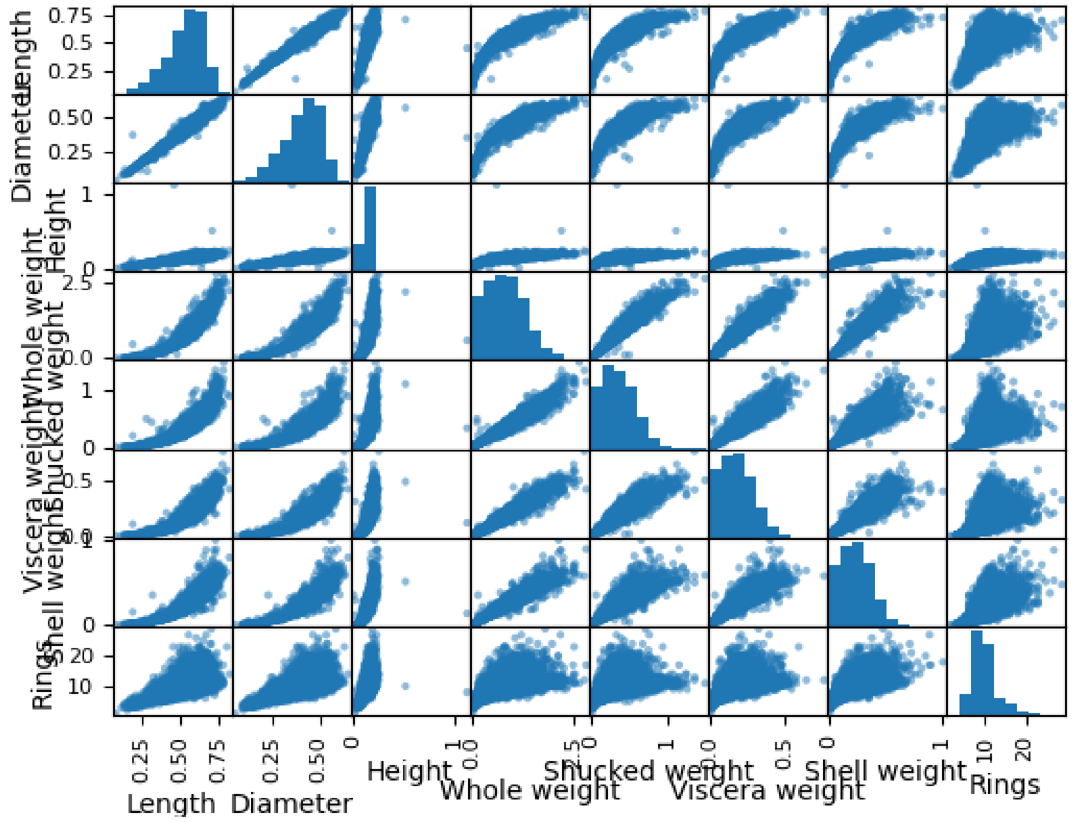
Method	Completeness Score
K-means	0.147
Hierarchical clustering	0.159
GMM	0.169
Ensemble Clustering	0.143

- Most methods performed poorly.
- Note: the performance of ensemble clustering is the worst.

```
#################################
#################################
import pandas as pd #Import pandas module
import matplotlib.pyplot as plt #Import the visualization module
from sklearn.cluster import KMeans #Import K-means module
from sklearn.metrics import confusion_matrix #Import confusion matrix module
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import completeness_score
from sklearn import metrics
import numpy as np
from sklearn.mixture import GaussianMixture #Import GMM module
from sklearn.cluster import AgglomerativeClustering

n_clusters = 15

data = pd.read_csv('abalone.csv') #Load the data file
pd.plotting.scatter_matrix(data);


markers = ['b.','g.','r.','c.','m.','y.','k.','bv','gv','rv','cv','mv','yv','kv','b1','g1','r1','c1','m1','y1','k1','b2','g2','r2','c2','m2','y2','k2','b3','g3','r3','c3','m3','y3','k3']
labels = data.iloc[:,-1]
ulabels = labels.unique()
d1,d2 = 1,2 ###The dimensions of the data
plt.figure();
for ii in range(ulabels.shape[0]):
    label = ulabels[ii];
    plt.plot(data[labels==label].iloc[:,d1],data[labels==label].iloc[:,d2],markers[ii]);


markers = ['b.','g.','r.','c.','m.','y.','k.','bv','gv','rv','cv','mv','yv','kv','b1','g1','r1','c1','m1','y1','k1','b2','g2','r2','c2','m2','y2','k2','b3','g3','r3','c3','m3','y3','k3']
labels = data.iloc[:,-1]
ulabels = labels.unique()
d1,d2 = 4,6 ###The dimensions of the data
plt.figure(); #Plot the figure with label = 1
for ii in range(ulabels.shape[0]):
    label = ulabels[ii];
    plt.plot(data[labels==label].iloc[:,d1],data[labels==label].iloc[:,d2],markers[ii]);
```

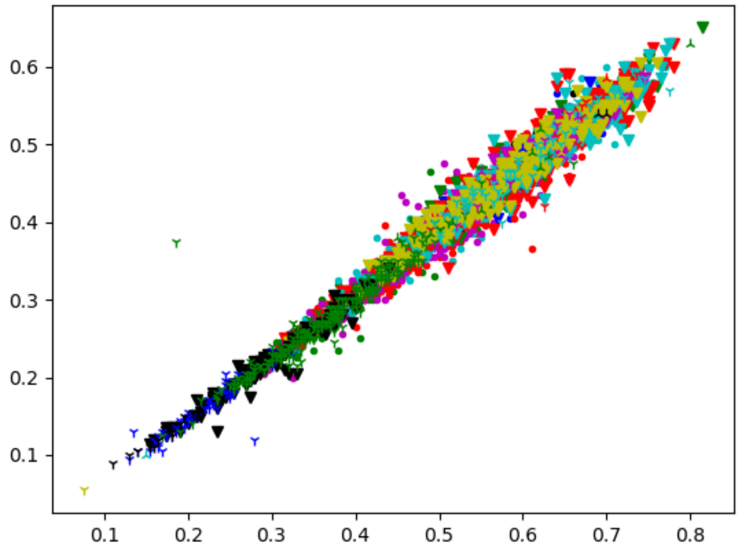
Abalone Data Analysis

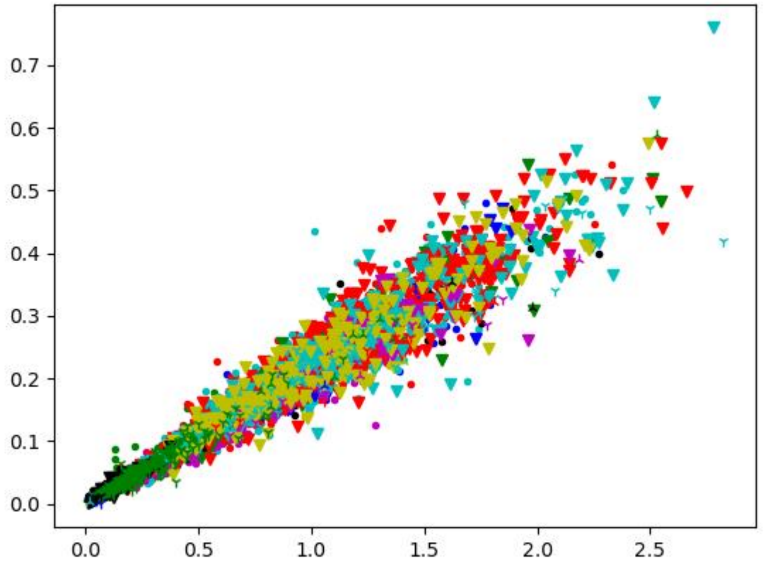
- Visualization (Scatter matrix plot)
- No well-separated clusters



Abalone Data Analysis

- Samples from different classes are highly overlapped.
- This is the reason why most methods didn’t perform well.





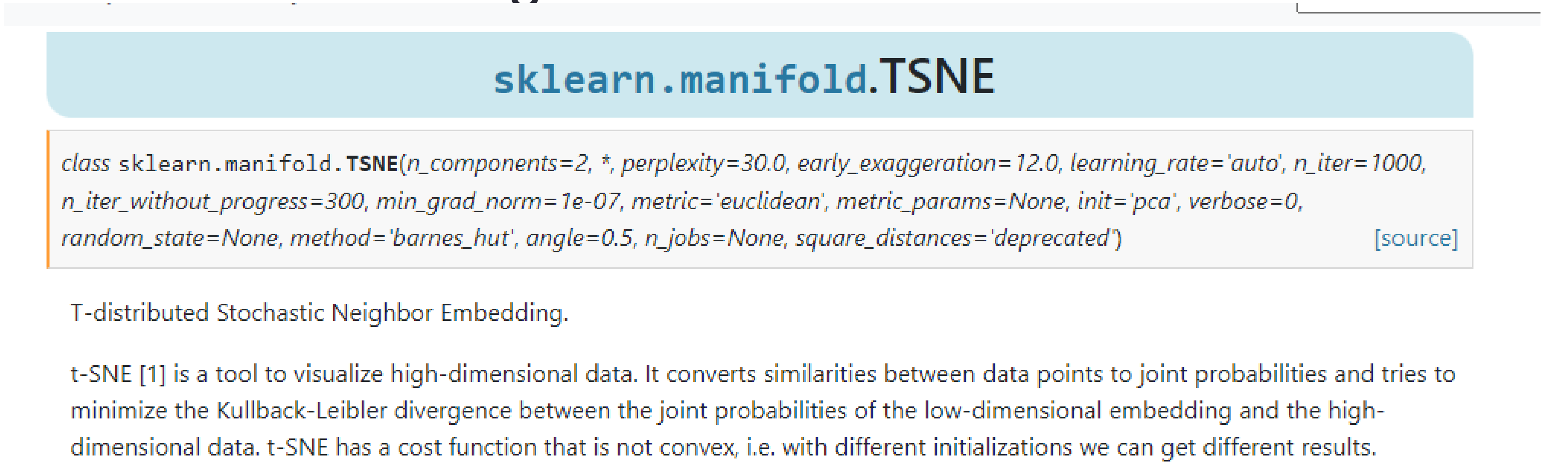
Abalone Data Analysis

- Other visualization techniques: t-SNE and LLE

```
###################################
###################################
#Other Visualization Tools t-SNE
from sklearn.manifold import TSNE
X = pd.get_dummies(data.iloc[:,:-1],drop_first=True)
X_embedded = TSNE(n_components=2,init='random', perplexity=3).fit_transform(X)

markers = ['b.','g.','r.','c.','m.','y.','k.','bv','gv','rv','cv','mv','yv','kv','b1','g1','r1','c1','m1','y1','k1','b2','g2','r2','c2','m2','y2','k2','b3','g3','r3','c3','m3','y3','k3']
labels = data.iloc[:,-1]
ulabels = labels.unique()
plt.figure();
for ii in range(ulabels.shape[0]):
    label = ulabels[ii];
    plt.plot(X_embedded[labels==label][:,0],X_embedded[labels==label][:,1],markers[ii]);


#Other Visualization Tools LLE
from sklearn.manifold import LocallyLinearEmbedding as LLE
X = pd.get_dummies(data.iloc[:,:-1],drop_first=True)
X_embedded = LLE(n_components=2).fit_transform(X)

markers = ['b.','g.','r.','c.','m.','y.','k.','bv','gv','rv','cv','mv','yv','kv','b1','g1','r1','c1','m1','y1','k1','b2','g2','r2','c2','m2','y2','k2','b3','g3','r3','c3','m3','y3','k3']
labels = data.iloc[:,-1]
ulabels = labels.unique()
plt.figure();
for ii in range(ulabels.shape[0]):
    label = ulabels[ii];
    plt.plot(X_embedded[labels==label][:,0],X_embedded[labels==label][:,1],markers[ii]);
```

Abalone Data Analysis

- Visualization using t-SNE.



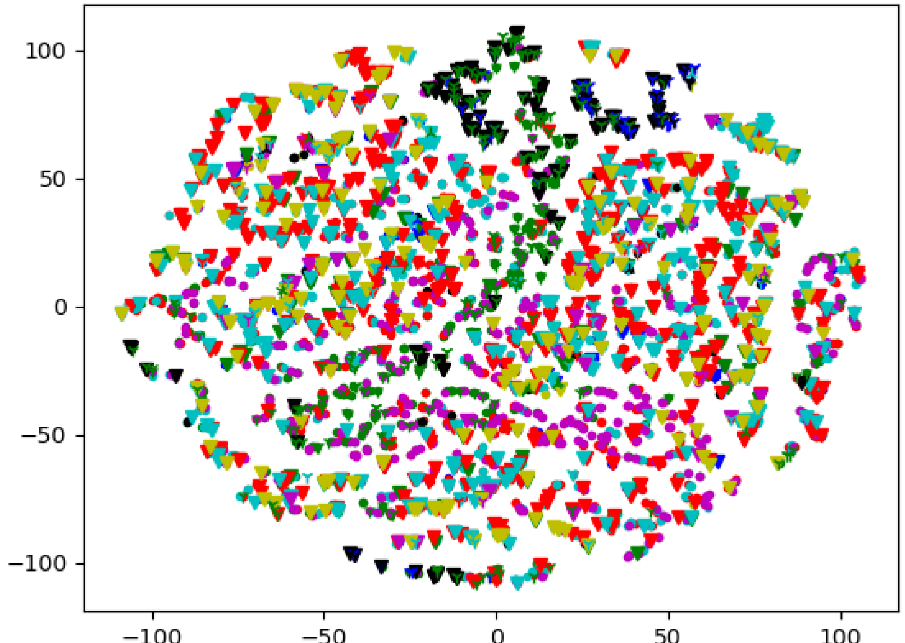
sklearn.manifold.TSNE

class sklearn.manifold.**TSNE**(*n_components=2, *, perplexity=30.0, early_exaggeration=12.0, learning_rate='auto', n_iter=1000, n_iter_without_progress=300, min_grad_norm=1e-07, metric='euclidean', metric_params=None, init='pca', verbose=0, random_state=None, method='barnes_hut', angle=0.5, n_jobs=None, square_distances='deprecated'*) [source]

T-distributed Stochastic Neighbor Embedding.

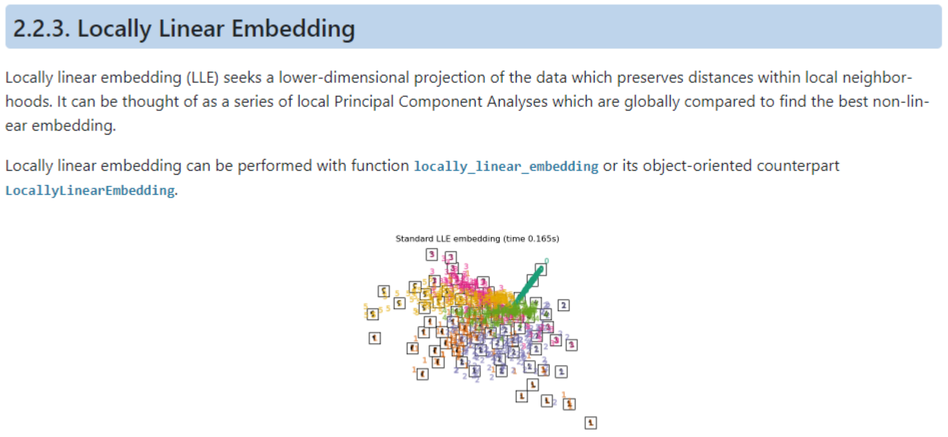
t-SNE [1] is a tool to visualize high-dimensional data. It converts similarities between data points to joint probabilities and tries to minimize the Kullback-Leibler divergence between the joint probabilities of the low-dimensional embedding and the high-dimensional data. t-SNE has a cost function that is not convex, i.e. with different initializations we can get different results.

- Several classes are close to each other.



Abalone Data Analysis

- Visualization using LLE

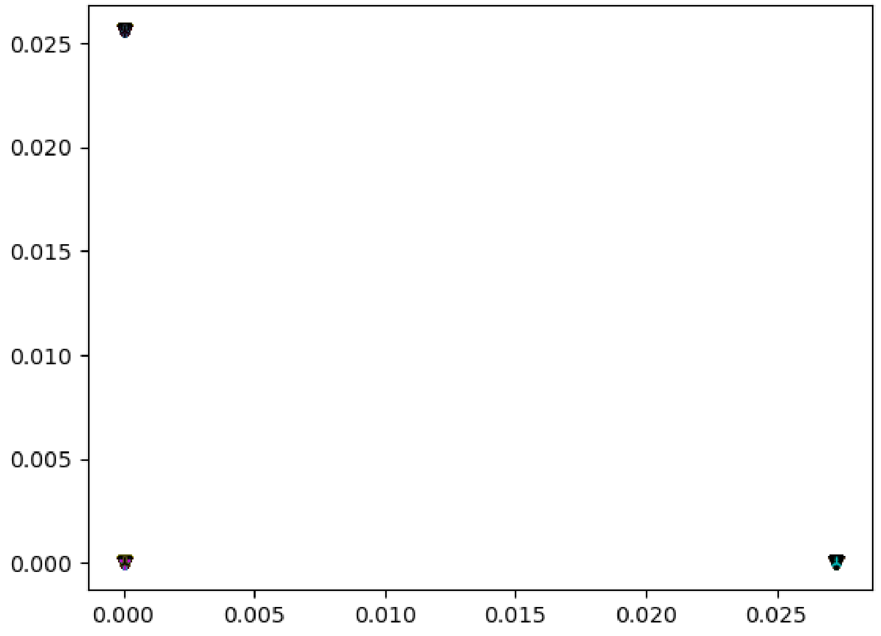


2.2.3. Locally Linear Embedding

Locally linear embedding (LLE) seeks a lower-dimensional projection of the data which preserves distances within local neighborhoods. It can be thought of as a series of local Principal Component Analyses which are globally compared to find the best non-linear embedding.

Locally linear embedding can be performed with function `locally_linear_embedding` or its object-oriented counterpart `LocallyLinearEmbedding`.

- There are three well-separated classes.



Abalone Data Analysis

- Because of this, we combine several classes together.
- Class 1 to 7 -> Group 1
- Class 8 to 11 -> Group 2
- Class 12 to 29 -> Group 3

# Abalone Data Analysis

```
################################
################################
import pandas as pd #Import pandas module
import matplotlib.pyplot as plt #Import the visualization module
from sklearn.cluster import KMeans #Import K-means module
from sklearn.metrics import confusion_matrix #Import confusion matrix module
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import completeness_score
from sklearn import metrics
import numpy as np
from sklearn.mixture import GaussianMixture #Import GMM module
from sklearn.cluster import AgglomerativeClustering

n_clusters = 3

data = pd.read_csv('abalone.csv') #Load the data file

###Combine the labels
labels = data.iloc[:,-1]
ind = (labels>=1) & (labels<=7); data.loc[ind,-1] = 1
ind = (labels>=8) & (labels<=11); data.loc[ind,-1] = 2
ind = (labels>=12) & (labels<=29); data.loc[ind,-1] = 3

X = pd.get_dummies(data.iloc[:,:-1],drop_first=True)
#Apply K-means to dataset
kmeans = KMeans(n_clusters=n_clusters).fit(X) #perform K-means clustering with number of clusters = 3
centroids = kmeans.cluster_centers_ #Extract the cluster centroids
labels = kmeans.labels_ #Extract the labels of clusters
label_encoder = LabelEncoder()
results = confusion_matrix(label_encoder.fit_transform(data.iloc[:,-1]),labels)
print(results)
acc = metrics.completeness_score(label_encoder.fit_transform(data.iloc[:,-1]),labels)
print(acc)


###Apply hierarchical clustering to the dataset
clusters = AgglomerativeClustering(n_clusters=n_clusters).fit(X) #perform K-means clustering with number of clusters = 3
labels = clusters.labels_ #Extract the labels of clusters
label_encoder = LabelEncoder()
results = confusion_matrix(label_encoder.fit_transform(data.iloc[:,-1]),labels)
print(results)
acc = metrics.completeness_score(label_encoder.fit_transform(data.iloc[:,-1]),labels)
print(acc)

###Apply GMM to the dataset
gmm = GaussianMixture(n_components=n_clusters).fit(X)
labels = gmm.predict(X)
label_encoder = LabelEncoder()
results = confusion_matrix(label_encoder.fit_transform(data.iloc[:,-1]),labels)
print(results)
acc = metrics.completeness_score(label_encoder.fit_transform(data.iloc[:,-1]),labels)
print(acc)


#Apply Ensemble clustering to Ecoli dataset
no_samples = data.shape[0]
no_estimators = 50
#Declare the weight of each vote
vote = 1/no_estimators
#co_association matrix is no_estimators X no_estimators (no_estimators patterns)
co_association = np.zeros((no_samples, no_samples))

#for each of your estimators
for est in range(no_estimators):
    #fit the data and grab the labels
    kmeans = KMeans(n_clusters=30,init='random',n_init=1).fit(X)
    labels = kmeans.labels_
    #find all associations and transform it into a numpy array
    res = [[int(i == j) for i in labels] for j in labels]
    res = np.array(res)
    #Vote and update the co_association matriz
    res = res * vote
    co_association = co_association + res
distance_matrix = 1-co_association
cluster = AgglomerativeClustering(n_clusters=n_clusters, affinity='euclidean',compute_distances=True).fit(distance_matrix)
labels = cluster.labels_ #Extract the labels of clusters
label_encoder = LabelEncoder()
# results = confusion_matrix(label_encoder.fit_transform(data.iloc[:,-1]),labels)
# print(results)
acc =  metrics.completeness_score(label_encoder.fit_transform(data.iloc[:,-1]),labels)
print(acc)
```

Abalone Data Analysis

The performance is greatly improved.

Method	Completeness Score
K-means	0.502
Hierarchical clustering	0.404
GMM	0.178
Ensemble Clustering	0.429

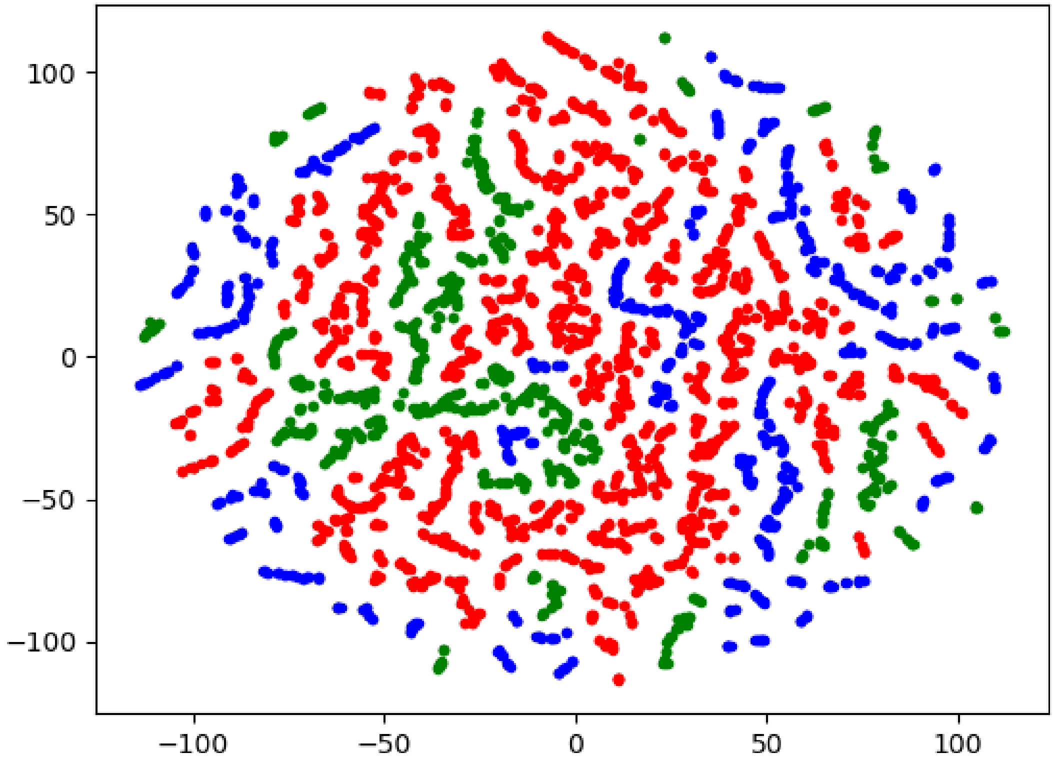
Abalone Data Analysis

- #####Visuzliation
- from sklearn.manifold import TSNE
- X = pd.get_dummies(data.iloc[:,:-1],drop_first=True)
- X_embedded = TSNE(n_components=2,init='random', perplexity=3).fit_transform(X)

- markers = ['b.','g.','r.','c.','m.','y.','k.','bv','gv','rv','cv','mv','yv','kv','b1','g1','r1','c1','m1','y1','k1','b2','g2','r2','c2','m2','y2','k2','b3','g3','r3','c3','m3','y3','k3']
- labels = data.iloc[:,-1]
- ulabels = labels.unique()
- plt.figure();
- for ii in range(ulabels.shape[0]):
-     label = ulabels[ii];
-     plt.plot(X_embedded[labels==label][:,0],X_embedded[labels==label][:,1],markers[ii]);

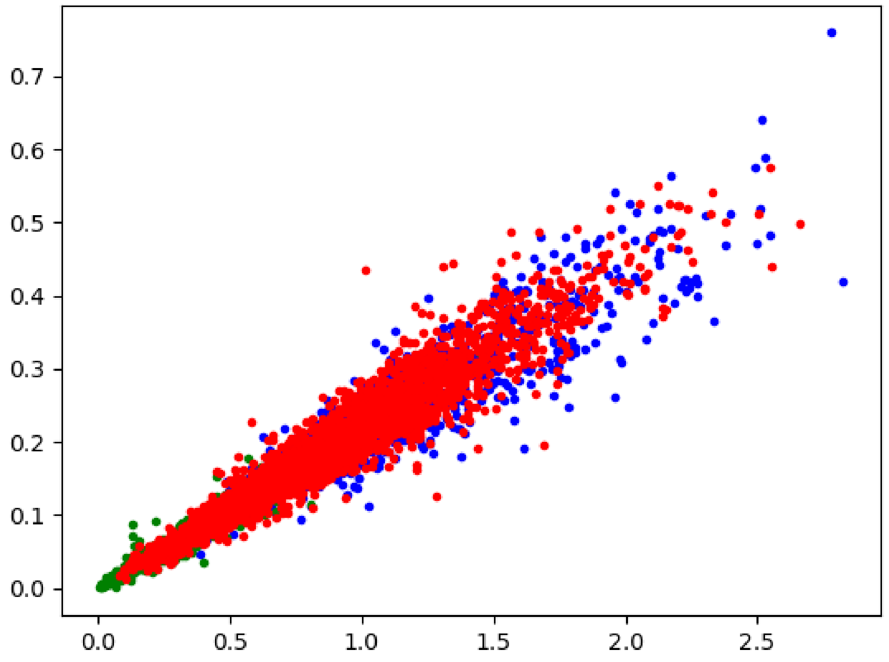
Abalone Data Analysis

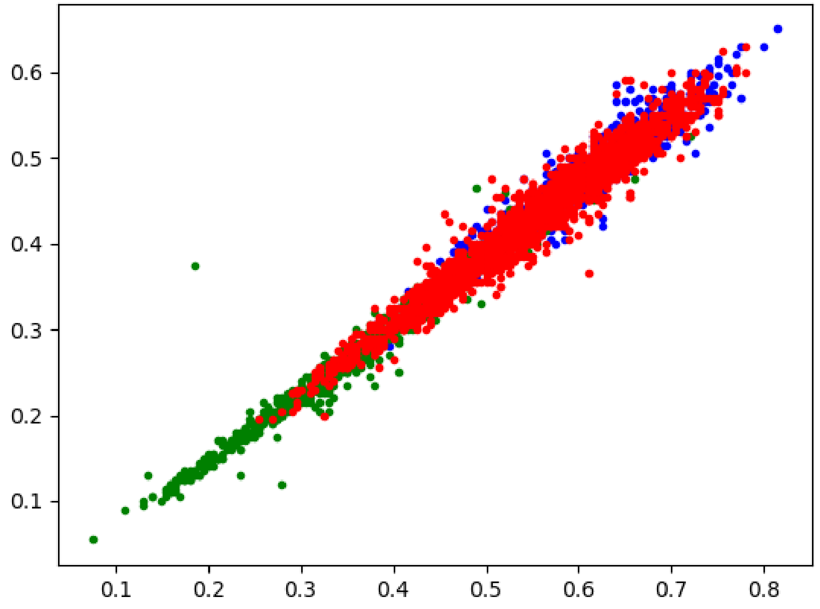
- t-SNE plot again
- The three classes are better separated.



Abalone Data Analysis

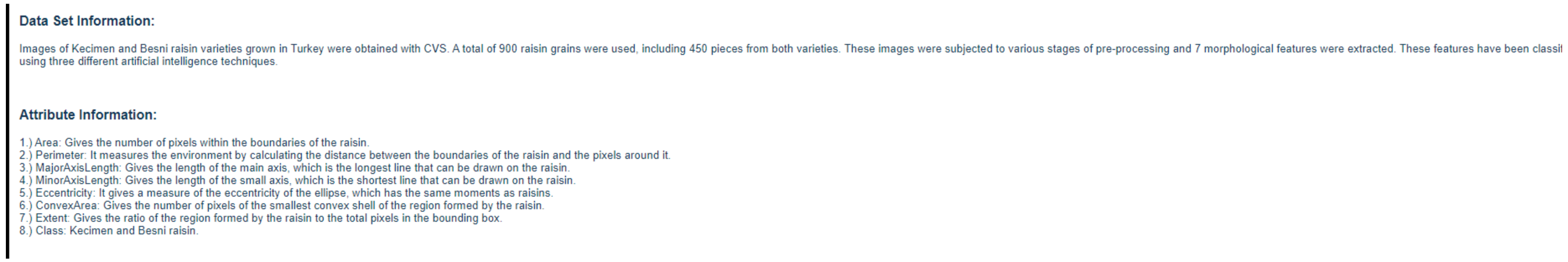
- Why k-means perform the best?
- The three classes have spherical shapes.
- Green and red classes are better separated.
- The results can be better if the red and blue classes are combined.





Raisin Data Analysis

- Downloaded from https://archive.ics.uci.edu/ml/datasets/Raisin+Dataset



Data Set Information:

Images of Kecimen and Besni raisin varieties grown in Turkey were obtained with CVS. A total of 900 raisin grains were used, including 450 pieces from both varieties. These images were subjected to various stages of pre-processing and 7 morphological features were extracted. These features have been classi
using three different artificial intelligence techniques.

Attribute Information:

1.) Area: Gives the number of pixels within the boundaries of the raisin.
2.) Perimeter: It measures the environment by calculating the distance between the boundaries of the raisin and the pixels around it.
3.) MajorAxisLength: Gives the length of the main axis, which is the longest line that can be drawn on the raisin.
4.) MinorAxisLength: Gives the length of the small axis, which is the shortest line that can be drawn on the raisin.
5.) Eccentricity: It gives a measure of the eccentricity of the ellipse, which has the same moments as raisins.
6.) ConvexArea: Gives the number of pixels of the smallest convex shell of the region formed by the raisin.
7.) Extent: Gives the ratio of the region formed by the raisin to the total pixels in the bounding box.
8.) Class: Kecimen and Besni raisin.

Raisin Data Analysis

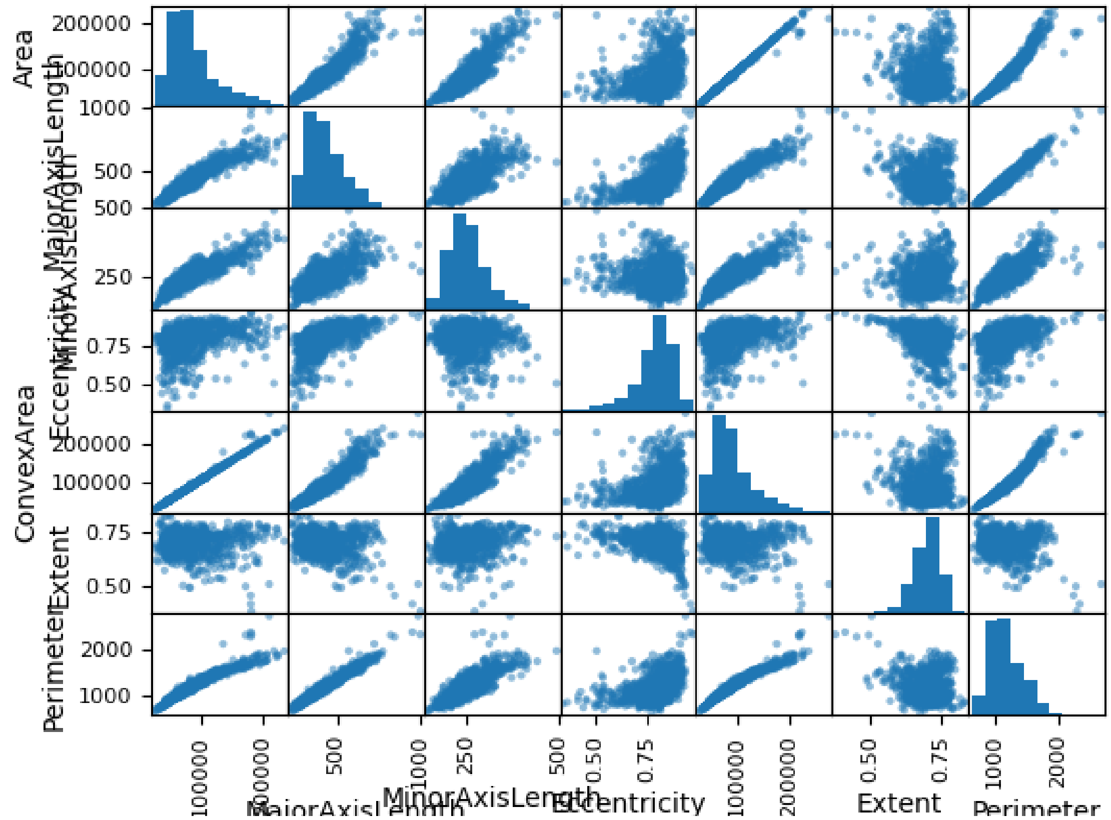
- Apply K-means, Hierarchical clustering, GMM and Ensemble Clustering to the data

Method	Completeness Score
K-means	0.302
Hierarchical clustering	0.297
GMM	0.371
Ensemble Clustering	0.121

- GMM performed the best.
- The performance of ensemble clustering is the worst.

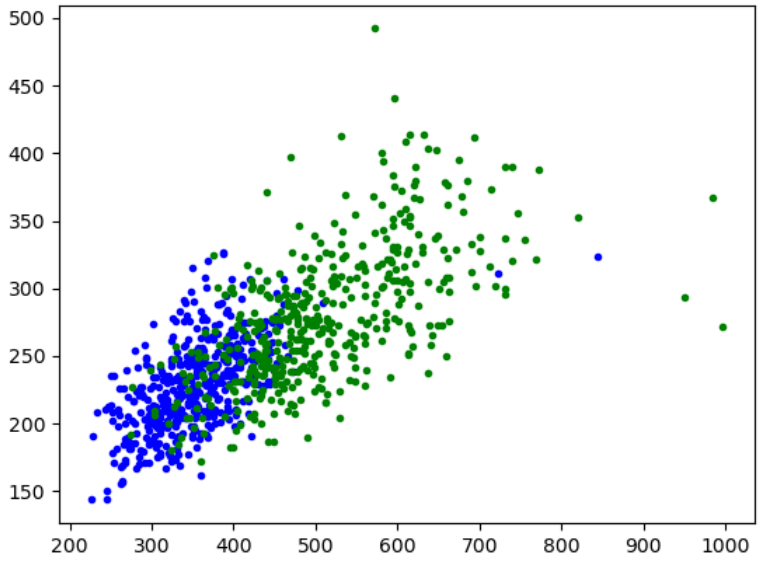
Raisin Data Analysis

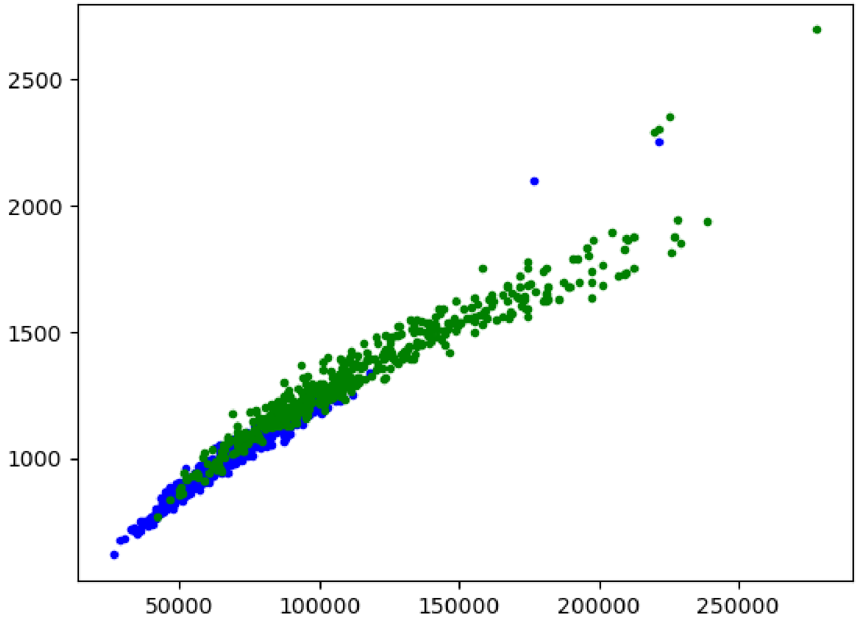
- Visualization (Scatter matrix plot)
- No well-separated clusters



Raisin Data Analysis

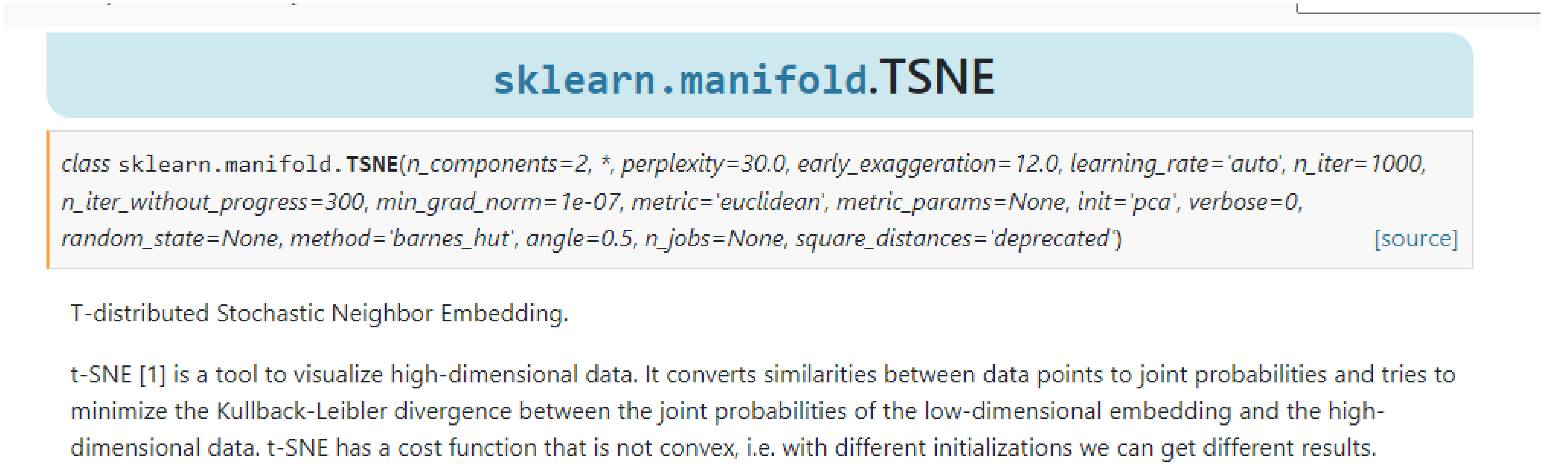
- Samples from different classes are separated well in some dimensions.
- The classes have elliptical shapes. Therefore, GMM is the best.





Raisin Data Analysis

- Visualization using t-SNE.



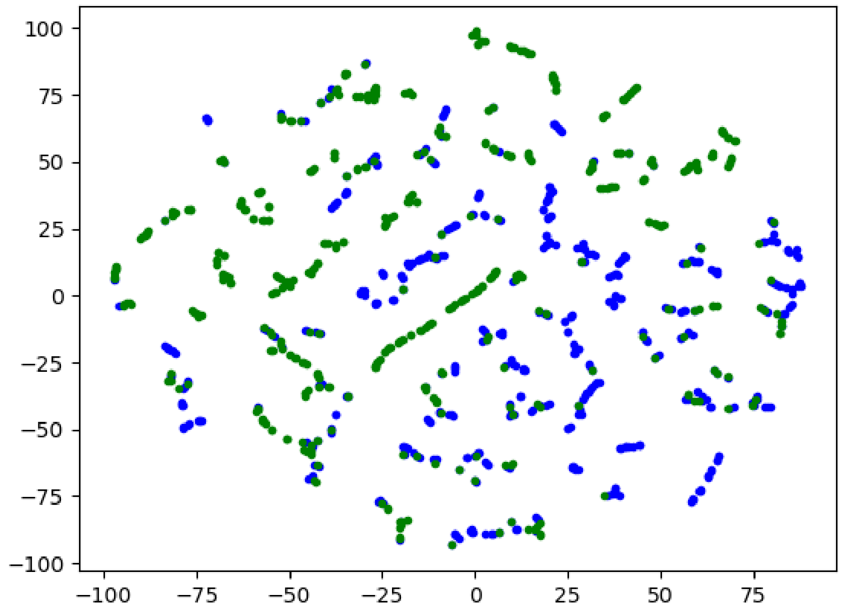
sklearn.manifold.TSNE

class `sklearn.manifold.`**`TSNE`**(*n_components=2, *, perplexity=30.0, early_exaggeration=12.0, learning_rate='auto', n_iter=1000, n_iter_without_progress=300, min_grad_norm=1e-07, metric='euclidean', metric_params=None, init='pca', verbose=0, random_state=None, method='barnes_hut', angle=0.5, n_jobs=None, square_distances='deprecated'*) [source]

T-distributed Stochastic Neighbor Embedding.

t-SNE [1] is a tool to visualize high-dimensional data. It converts similarities between data points to joint probabilities and tries to minimize the Kullback-Leibler divergence between the joint probabilities of the low-dimensional embedding and the high-dimensional data. t-SNE has a cost function that is not convex, i.e. with different initializations we can get different results.

- tSNE didn't provide much information to this dataset.



Parameter Tuning for Clustering Methods

The parameter(s) of clustering methods can be tunned by silhouette analysis.

Steps:

- Apply the clustering method with different parameter settings (e.g. K-means with different random seeds, GMM with different settings).
- Then, for each clustering results, find the silhouette coefficient.
- The one with the optimal silhouette coefficient is the best setting.

Note: As silhouette coefficient is different from class labels, there is no guarantee that the setting can get the best accuracy.