Non-Gaussian statistical with exponential data

import numpy as np

import matplotlib.pyplot as plt

from scipy.stats import expon

from sklearn.mixture import GaussianMixture

from sklearn.metrics import f1\_score, confusion\_matrix

import seaborn as sns

# Generate non-Gaussian exponential synthetic data

np.random.seed(42)

data1 = expon.rvs(scale=2, size=500, random\_state=42)

data2 = expon.rvs(scale=5, size=500, random\_state=42)

data = np.column\_stack((data1, data2))

# Split the data into features and labels

X = data

y = np.concatenate([np.zeros((500, 1)), np.ones((500, 1))])

# Reshape y to have a shape of (500,2)

y = y.reshape((500, 2))

# Fit a Gaussian Mixture Model

gmm = GaussianMixture(n\_components=2, random\_state=42)

gmm.fit(X, y)

# Predict labels

y\_pred = gmm.predict(X)

# Calculate F1-score

f1 = f1\_score(y[:, 0], y\_pred)

print(f"F1-score: {f1:.2f}")

# Plot the data with predicted labels

plt.figure(figsize=(8, 6))

plt.scatter(X[:, 0], X[:, 1], c=y\_pred, cmap='viridis', alpha=0.5)

plt.title("Non-Gaussian Exponential Synthetic Data")

plt.xlabel("Feature 1")

plt.ylabel("Feature 2")

plt.show()

# Plot the confusion matrix

cm = confusion\_matrix(y[:, 0], y\_pred)

plt.figure(figsize=(8, 6))

sns.heatmap(cm, annot=True, cmap='Blues', fmt='g')

plt.title("Confusion Matrix")

plt.xlabel("Predicted Labels")

plt.ylabel("True Labels")

plt.show()

Svm exopentinal

import numpy as np

import matplotlib.pyplot as plt

from sklearn.model\_selection import train\_test\_split

from sklearn.svm import SVC

from sklearn.metrics import confusion\_matrix, f1\_score

# Step 1: Generate synthetic data

np.random.seed(42)

# Generate 1000 samples from exponential distribution

data = np.random.exponential(scale=1.0, size=(1000, 2))

# Generate labels, e.g., if sum of the features is greater than 1, label as 1, else 0

labels = (np.sum(data, axis=1) > 1).astype(int)

# Step 2: Split data into train and test sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(data, labels, test\_size=0.2, random\_state=42)

# Step 3: Train SVM model

svm\_model = SVC(kernel='linear', C=1.0, random\_state=42)

svm\_model.fit(X\_train, y\_train)

# Step 4: Evaluate model

y\_pred = svm\_model.predict(X\_test)

f1 = f1\_score(y\_test, y\_pred)

print("F1 Score:", f1)

# Plot confusion matrix

def plot\_confusion\_matrix(y\_true, y\_pred, classes, title='Confusion matrix', cmap=plt.cm.Blues):

cm = confusion\_matrix(y\_true, y\_pred)

plt.imshow(cm, interpolation='nearest', cmap=cmap)

plt.title(title)

plt.colorbar()

tick\_marks = np.arange(len(classes))

plt.xticks(tick\_marks, classes, rotation=45)

plt.yticks(tick\_marks, classes)

fmt = 'd'

thresh = cm.max() / 2.

for i in range(cm.shape[0]):

for j in range(cm.shape[1]):

plt.text(j, i, format(cm[i, j], fmt),

horizontalalignment="center",

color="white" if cm[i, j] > thresh else "black")

plt.ylabel('True label')

plt.xlabel('Predicted label')

plt.tight\_layout()

# Plot non-normalized confusion matrix

plt.figure()

plot\_confusion\_matrix(y\_test, y\_pred, classes=['Class 0', 'Class 1'], title='Confusion matrix, without normalization')

plt.show()

Latent Non-Gaussian Models exponential

import numpy as np

from sklearn.mixture import GaussianMixture

from sklearn.metrics import confusion\_matrix, f1\_score

import matplotlib.pyplot as plt

import seaborn as sns

# Generating synthetic non-Gaussian exponential data

num\_samples = 1000

mean = 5

scale = 2

# Generate non-Gaussian exponential data

data = np.random.exponential(scale=scale, size=num\_samples) + mean

# Train Gaussian Mixture Model

gmm = GaussianMixture(n\_components=2, random\_state=42)

gmm.fit(data.reshape(-1, 1))

# Predictions

predictions = gmm.predict(data.reshape(-1, 1))

# True labels (for synthetic data, assuming we know the distribution)

true\_labels = np.zeros\_like(predictions)

true\_labels[data > mean] = 1

# Calculate F1 Score

f1 = f1\_score(true\_labels, predictions)

# Plot Confusion Matrix

cm = confusion\_matrix(true\_labels, predictions)

plt.figure(figsize=(8, 6))

sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", cbar=False,

xticklabels=["Class 0", "Class 1"], yticklabels=["Class 0", "Class 1"])

plt.xlabel('Predicted')

plt.ylabel('True')

plt.title(f'Confusion Matrix (F1 Score: {f1:.2f})')

plt.show()

Latent Non-Gaussian Models with synthetic data

import numpy as np

import matplotlib.pyplot as plt

from sklearn.datasets import make\_blobs

from sklearn.model\_selection import train\_test\_split

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import confusion\_matrix, f1\_score

# Generate synthetic non-Gaussian data

X, y = make\_blobs(n\_samples=1000, centers=3, random\_state=42)

# Split data into train and test sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Train a classifier (Random Forest, for example)

clf = RandomForestClassifier(random\_state=42)

clf.fit(X\_train, y\_train)

# Make predictions

y\_pred = clf.predict(X\_test)

# Calculate F1 score

f1 = f1\_score(y\_test, y\_pred, average='weighted')

# Print F1 score

print("F1 Score:", f1)

# Print confusion matrix

conf\_matrix = confusion\_matrix(y\_test, y\_pred)

print("Confusion Matrix:")

print(conf\_matrix)

# Plotting the data

plt.figure(figsize=(8, 6))

plt.scatter(X[:, 0], X[:, 1], c=y, cmap='viridis', s=50, alpha=0.6)

plt.title('Synthetic Non-Gaussian Data')

plt.xlabel('Feature 1')

plt.ylabel('Feature 2')

plt.colorbar(label='Class')

plt.show()

**Non-Gaussian statistical models**

**import numpy as np**

**import matplotlib.pyplot as plt**

**from sklearn.mixture import GaussianMixture**

**from sklearn.metrics import f1\_score, confusion\_matrix**

**import seaborn as sns**

**# Generate non-Gaussian synthetic data**

**np.random.seed(42)**

**data = np.random.uniform(-5, 5, size=(1000, 2))**

**data[:500, 0] += 5 # Shift the first 500 points to the right**

**data[500:, 1] += 5 # Shift the remaining 500 points upwards**

**# Split the data into features and labels**

**X = data**

**y = np.concatenate([np.zeros(500), np.ones(500)])**

**# Fit a Gaussian Mixture Model**

**gmm = GaussianMixture(n\_components=2, random\_state=42)**

**gmm.fit(X)**

**# Predict labels**

**y\_pred = gmm.predict(X)**

**# Calculate F1-score**

**f1 = f1\_score(y, y\_pred)**

**print(f"F1-score: {f1:.2f}")**

**# Plot the data with predicted labels**

**plt.figure(figsize=(8, 6))**

**plt.scatter(X[:, 0], X[:, 1], c=y\_pred, cmap='viridis', alpha=0.5)**

**plt.title("Non-Gaussian Synthetic Data")**

**plt.xlabel("Feature 1")**

**plt.ylabel("Feature 2")**

**plt.show()**

**# Plot the confusion matrix**

**cm = confusion\_matrix(y, y\_pred)**

**plt.figure(figsize=(8, 6))**

**sns.heatmap(cm, annot=True, cmap='Blues', fmt='g')**

**plt.title("Confusion Matrix")**

**plt.xlabel("Predicted Labels")**

**plt.ylabel("True Labels")**

**plt.show()**