```
import numpy as np
import matplotlib.pyplot as plt
```

```
data = np.random.randn(50000) * 20 + 20

def find_anomalies(data):
  anomalies=[]

  data_std = np.std(data)
  data_mean = np.mean(data)
  anomaly_cut = data_std * 3

  low_limit = data_mean - anomaly_cut
  upp_limit = data_mean + anomaly_cut
  print(low_limit,' - ',upp_limit)
  for datum in data:
    if datum > upp_limit or datum < low_limit:
      anomalies.append(datum)

  return anomalies
```

```
anomalies = find_anomalies(data)
len(anomalies)
```

```
-39.69992174203004  -  79.82009753093006
147
```

```
import pandas as pd
stocks = pd.read_csv('http://dicyg.fi-c.unam.mx:8080/lalo/pypcd/presentaciones/st
                     header='infer')
stocks.index = stocks.Date
stocks = stocks.drop(['Date'],axis=1)
stocks.head(3)
```

```
from matplotlib import projections
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt

```

```
fig = plt.figure(figsize=(8,5)).gca(projection='3d')
fig.scatter(stocks.MSFT,stocks.F,stocks.BAC)
fig.set_xlabel('Microsoft')
fig.set_ylabel('Ford')
fig.set_zlabel('Bank of America')
plt.show()
```

```
# Boxplot
data = np.random.randn(50000)*20+20
data = pd.DataFrame(data,columns=['random'])
#data
```

```
import seaborn as sns
sns.boxplot(data=data)
```

```
sns.boxplot(data=stocks)
```

```
  from pandas.core.dtypes.missing import na_value_for_dtype
  fig,ax = plt.subplots(nrows=1,ncols=3,figsize=(11,3))
  sns.distplot(stocks.MSFT, ax=ax[0],color='darkblue')
  sns.distplot(stocks.F, ax=ax[1], color='red')
  sns.distplot(stocks.BAC, ax=ax[2], color='green')
```

```
# DBSCAN
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
from pylab import rcParams
rcParams['figure.figsize'] = 5,4

from sklearn.cluster import DBSCAN
from collections import Counter
```

```
df = pd.read_csv("https://bit.ly/3arouNg")
df.shape
```

```
(6463, 13)
```

```
df.head(1)
```

```
data = df.iloc[:,1:3]
data.head(1)
```

```
model = DBSCAN(eps=0.2, min_samples=20).fit(data)
print(model)
model.get_params()
```

```
DBSCAN(eps=0.2, min_samples=20)
{'algorithm': 'auto',
 'eps': 0.2,
 'leaf_size': 30,
 'metric': 'euclidean',
 'metric_params': None,
 'min_samples': 20,
 'n_jobs': None,
 'p': None}
```

```
print(Counter(model.labels_))
outliers_df = pd.DataFrame(data)
#print(outliers_df[model.labels_==-1])
```

```
Counter({0: 6281, -1: 117, 1: 40, 2: 25})
```

```
fig = plt.figure()
ax = fig.add_axes([0.1,0.1,2,2])
colors = model.labels_

ax.scatter(data.iloc[:,0].values, data.iloc[:,1].values, c=colors, s=100)
```

```
# KrenelPCA "desde cero"
```

```
  import scipy
  from scipy.spatial.distance import pdist, squareform
  from scipy.linalg import eigh
  import numpy as np
  import matplotlib.pyplot as plt
```

```
def rbf_kpca(X, gamma, n_components):
  # distancias cuadradas de cada pareja
  # del conjunto de datos, es MxN-dimensional
  sq_dists = pdist(X, 'sqeuclidean')
  #print(sq_dists.shape)
  # convertirla en matriz
  mat_sq_dists = squareform(sq_dists)
```

```
  #print(mat_sq_dists.shape)
  # Matriz de núcleos
  K = np.exp(-gamma*mat_sq_dists)
  # Centrar la matriz de núcleos
  N = K.shape[0]
  one_n = np.ones((N,N)) / N
  K = K - one_n.dot(K) - K.dot(one_n) + one_n.dot(K).dot(one_n)
  # Eigenpares; scipy.linalg.eigh devuelve los eigenpares ordenados ascendentemen
  eigenvals,eigenvecs = eigh(K)
  eigenvals,eigenvecs = eigenvals[::-1], eigenvecs[:,::-1] #invertir el orden
  # Elegir los k primeros eigenvecs
  alphas = np.column_stack([eigenvecs[:,i] for i in range(n_components)])
  # Eigenvals
  lambdas = np.column_stack([eigenvals[i] for i in range(n_components)])
  return alphas,lambdas
```

```
# Medias lunas
from sklearn.datasets import make_moons
X,y = make_moons(n_samples=100, random_state=123)
plt.scatter(X[y==0,0],X[y==0,1],color='red',marker='^',alpha=0.5)
plt.scatter(X[y==1,0],X[y==1,1],color='blue',marker='o',alpha=0.5)
```

```
from sklearn.decomposition import PCA
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X)
```

```
from IPython.core.pylabtools import figsize
fig,ax = plt.subplots(nrows=1,ncols=2,figsize=(10,4))
ax[0].scatter(X_pca[y==0,0],X_pca[y==0,1],color='red',marker='^',alpha=0.5)
ax[0].scatter(X_pca[y==1,0],X_pca[y==1,1],color='blue',marker='o',alpha=0.5)
ax[1].scatter(X_pca[y==0,0],np.zeros((50,1))+0.02,color='red',marker='^',alpha=0.
ax[1].scatter(X_pca[y==1,0],np.zeros((50,1))-0.02,color='blue',marker='o',alpha=0
ax[1].set_ylim([-1,1])
plt.show()
```

```
X_kpca,l = rbf_kpca(X, gamma=15, n_components=2)
fig,ax = plt.subplots(nrows=1,ncols=2,figsize=(10,4))
ax[0].scatter(X_kpca[y==0,0],X_kpca[y==0,1],color='red',marker='^',alpha=0.5)
ax[0].scatter(X_kpca[y==1,0],X_kpca[y==1,1],color='blue',marker='o',alpha=0.5)
ax[1].scatter(X_kpca[y==0,0],np.zeros((50,1))+0.02,color='red',marker='^',alpha=0
ax[1].scatter(X_kpca[y==1,0],np.zeros((50,1))-0.02,color='blue',marker='o',alpha=
ax[1].set_ylim([-1,1])
plt.show()
```

```
# KPCA de sklearn
from sklearn.decomposition import KernelPCA
kpca = KernelPCA(n_components=2, kernel='rbf', gamma=15)
X_kpca = kpca.fit_transform(X)
fig,ax = plt.subplots(nrows=1,ncols=2,figsize=(10,4))
ax[0].scatter(X_kpca[y==0,0],X_kpca[y==0,1],color='red',marker='^',alpha=0.5)
ax[0].scatter(X_kpca[y==1,0],X_kpca[y==1,1],color='blue',marker='o',alpha=0.5)
ax[1].scatter(X_kpca[y==0,0],np.zeros((50,1))+0.02,color='red',marker='^',alpha=0
ax[1].scatter(X_kpca[y==1,0],np.zeros((50,1))-0.02,color='blue',marker='o',alpha=
ax[1].set_ylim([-1,1])
plt.show()
```

```
from sklearn.decomposition import KernelPCA
kpca = KernelPCA(n_components=2, kernel='rbf', gamma=5)
X_kpca = kpca.fit_transform(X)
fig,ax = plt.subplots(nrows=1,ncols=2,figsize=(10,4))
ax[0].scatter(X_kpca[y==0,0],X_kpca[y==0,1],color='red',marker='^',alpha=0.5)
ax[0].scatter(X_kpca[y==1,0],X_kpca[y==1,1],color='blue',marker='o',alpha=0.5)
ax[1].scatter(X_kpca[y==0,0],np.zeros((50,1))+0.02,color='red',marker='^',alpha=0
ax[1].scatter(X_kpca[y==1,0],np.zeros((50,1))-0.02,color='blue',marker='o',alpha=
ax[1].set_ylim([-1,1])
plt.show()
```

```
from sklearn.decomposition import KernelPCA
kpca = KernelPCA(n_components=2, kernel='rbf', gamma=10)
X_kpca = kpca.fit_transform(X)
fig,ax = plt.subplots(nrows=1,ncols=2,figsize=(10,4))
ax[0].scatter(X_kpca[y==0,0],X_kpca[y==0,1],color='red',marker='^',alpha=0.5)
ax[0].scatter(X_kpca[y==1,0],X_kpca[y==1,1],color='blue',marker='o',alpha=0.5)
ax[1].scatter(X_kpca[y==0,0],np.zeros((50,1))+0.02,color='red',marker='^',alpha=0
ax[1].scatter(X_kpca[y==1,0],np.zeros((50,1))-0.02,color='blue',marker='o',alpha=
ax[1].set_ylim([-1,1])
plt.show()
```

```
from sklearn.decomposition import KernelPCA
kpca = KernelPCA(n_components=2, kernel='rbf', gamma=20)
X_kpca = kpca.fit_transform(X)
fig,ax = plt.subplots(nrows=1,ncols=2,figsize=(10,4))
ax[0].scatter(X_kpca[y==0,0],X_kpca[y==0,1],color='red',marker='^',alpha=0.5)
ax[0].scatter(X_kpca[y==1,0],X_kpca[y==1,1],color='blue',marker='o',alpha=0.5)
ax[1].scatter(X_kpca[y==0,0],np.zeros((50,1))+0.02,color='red',marker='^',alpha=0
ax[1].scatter(X_kpca[y==1,0],np.zeros((50,1))-0.02,color='blue',marker='o',alpha=
ax[1].set_ylim([-1,1])
plt.show()
```

```
from sklearn.datasets import make_circles
X,y = make_circles(n_samples=1000, random_state=123, noise=0.1, factor=0.2)
plt.scatter(X[y==0,0],X[y==0,1],color='red',marker='^',alpha=0.5)
plt.scatter(X[y==1,0],X[y==1,1],color='blue',marker='o',alpha=0.5)
plt.show()
```

```
# Con PCA
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X)
fig,ax = plt.subplots(nrows=1,ncols=2,figsize=(10,4))
ax[0].scatter(X_pca[y==0,0],X_pca[y==0,1],color='red',marker='^',alpha=0.5)
ax[0].scatter(X_pca[y==1,0],X_pca[y==1,1],color='blue',marker='o',alpha=0.5)
ax[1].scatter(X_pca[y==0,0],np.zeros((500,1))+0.02,color='red',marker='^',alpha=0
ax[1].scatter(X_pca[y==1,0],np.zeros((500,1))-0.02,color='blue',marker='o',alpha=
ax[1].set_ylim([-1,1])
plt.show()
```

```
X_kpca,l = rbf_kpca(X, gamma=15, n_components=2)
fig,ax = plt.subplots(nrows=1,ncols=2,figsize=(10,4))
ax[0].scatter(X_kpca[y==0,0],X_kpca[y==0,1],color='red',marker='^',alpha=0.5)
ax[0].scatter(X_kpca[y==1,0],X_kpca[y==1,1],color='blue',marker='o',alpha=0.5)
ax[1].scatter(X_kpca[y==0,0],np.zeros((500,1))+0.02,color='red',marker='^',alpha=
ax[1].scatter(X_kpca[y==1,0],np.zeros((500,1))-0.02,color='blue',marker='o',alpha
ax[1].set_ylim([-1,1])
plt.show()
```

```
from sklearn.decomposition import KernelPCA
kpca = KernelPCA(n_components=2, kernel='rbf', gamma=15)
X_kpca = kpca.fit_transform(X)
fig,ax = plt.subplots(nrows=1,ncols=2,figsize=(10,4))
ax[0].scatter(X_kpca[y==0,0],X_kpca[y==0,1],color='red',marker='^',alpha=0.5)
ax[0].scatter(X_kpca[y==1,0],X_kpca[y==1,1],color='blue',marker='o',alpha=0.5)
ax[1].scatter(X_kpca[y==0,0],np.zeros((500,1))+0.02,color='red',marker='^',alpha=
ax[1].scatter(X_kpca[y==1,0],np.zeros((500,1))-0.02,color='blue',marker='o',alpha
ax[1].set_ylim([-1,1])
plt.show()
```

```
1
```

```
1
```

```
1
```

```
1
```

```
1
```

```
1
```

0 s completado a las 12:11