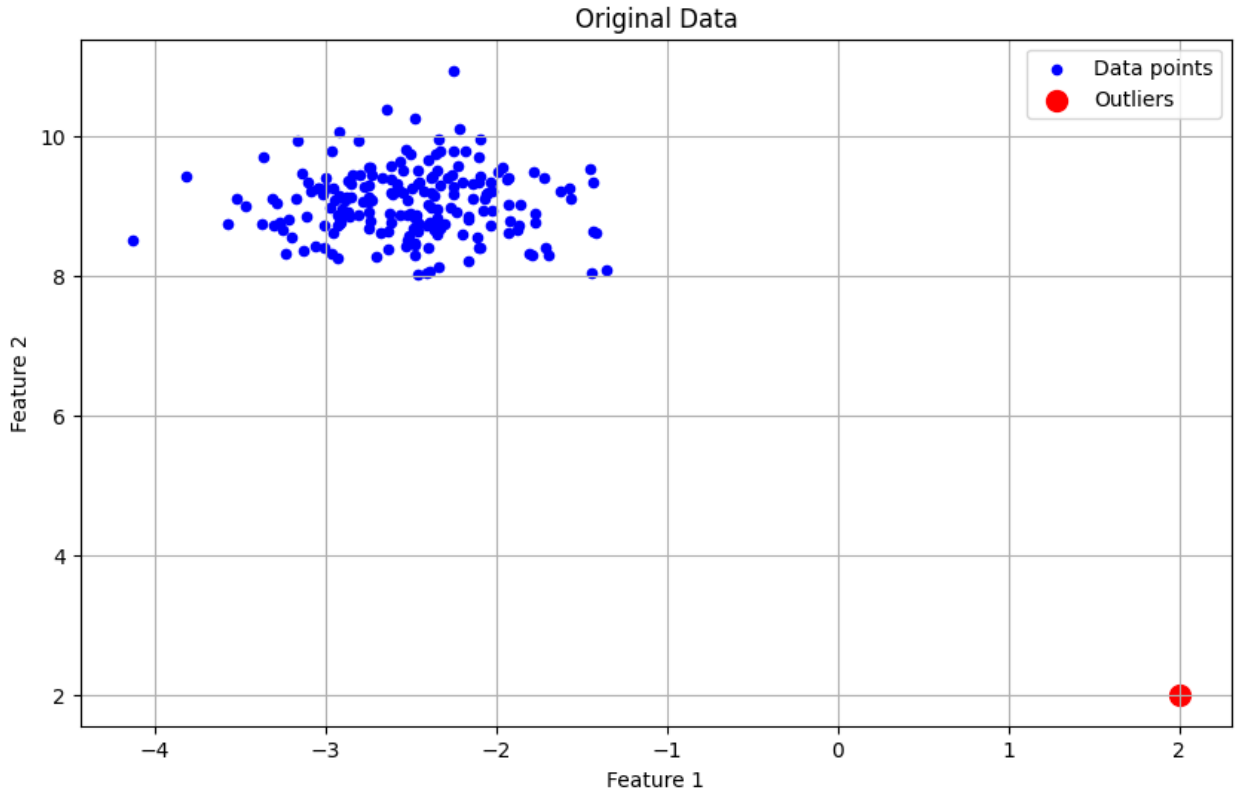
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import make_blobs
from sklearn.neighbors import LocalOutlierFactor

# Generate synthetic data with outliers
X, _ = make_blobs(n_samples=200, centers=1, cluster_std=0.5,
random_state=42)
outliers = np.array([[2, 2]])  # Introduce an outlier
X = np.append(X, outliers, axis=0)

# Visualize the data
plt.figure(figsize=(10, 6))
plt.scatter(X[:, 0], X[:, 1], color='b', s=20, label='Data points')
plt.scatter(outliers[:, 0], outliers[:, 1], color='r', s=100,
label='Outliers')
plt.title('Original Data')
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.legend()
plt.grid(True)
plt.show()
```




```
# Density-based outlier detection (Local Outlier Factor)
lof = LocalOutlierFactor(n_neighbors=20, contamination=0.1)  # Adjust
```

```
contamination based on expected outlier ratio
lof_scores = -lof.fit_predict(X)

# Visualize density-based outlier detection
plt.figure(figsize=(10, 6))
plt.scatter(X[:, 0], X[:, 1], s=20, color='b', label='Data points')
plt.scatter(outliers[:, 0], outliers[:, 1], color='r', s=100,
label='Outliers')
plt.scatter(X[:, 0], X[:, 1], s=1000 * lof_scores, edgecolors='r',
facecolors='none', label='Outlier scores')
plt.title('Density-based Outlier Detection (Local Outlier Factor)')
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.legend()
plt.grid(True)
plt.show()
```

```
/usr/local/lib/python3.10/dist-packages/matplotlib/collections.py:963:
RuntimeWarning: invalid value encountered in sqrt
  scale = np.sqrt(self._sizes) * dpi / 72.0 * self._factor
```