Problem Statement

In many real-world applications, identifying anomalous or unusual observations is critical. This includes areas such as fraud detection, intrusion detection in networks, and medical diagnosis. Detecting such anomalies can help prevent significant losses, improve security, and enhance decision-making processes.

The **Local Outlier Factor (LOF)** algorithm is a popular unsupervised technique used to detect anomalies by comparing the local density of a data point to that of its neighbours. Unlike global methods, LOF focuses on identifying data points that deviate significantly from their surrounding points in terms of density, making it especially useful for detecting subtle and context-specific anomalies.

In this experiment, I will apply the LOF method to both synthetic and real-world datasets in order to identify outliers and interpret their significance in the context of the data.

Note

Outliers vs. Noise:
Outliers are data points that deviate significantly from the rest of the data and often carry meaningful information (e.g., fraudulent transactions or medical abnormalities).
Noise, on the other hand, refers to random errors or irrelevant variations in the data that do not usually hold significant value and may obscure the actual patterns.

Objectives

1. Import Libraries

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

```
from sklearn.neighbors import LocalOutlierFactor
from sklearn.datasets import make_blobs
from sklearn.preprocessing import StandardScaler
```

2. Generate Synthetic Dataset

```
# Generate synthetic data with 2D blobs
X, _ = make_blobs(n_samples=300, centers=1, cluster_std=1.0,
random_state=42)

# Add some random outliers
outliers = np.random.uniform(low=-6, high=6, size=(20, 2))
X = np.vstack([X, outliers])

# Standardize the data
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

3. Apply Local Outlier Factor

```
# Apply LOF
lof = LocalOutlierFactor(n_neighbors=20, contamination=0.05)
# contamination = expected proportion of outliers
y_pred = lof.fit_predict(X_scaled)
scores = lof.negative_outlier_factor_
```

4. Visualise Results

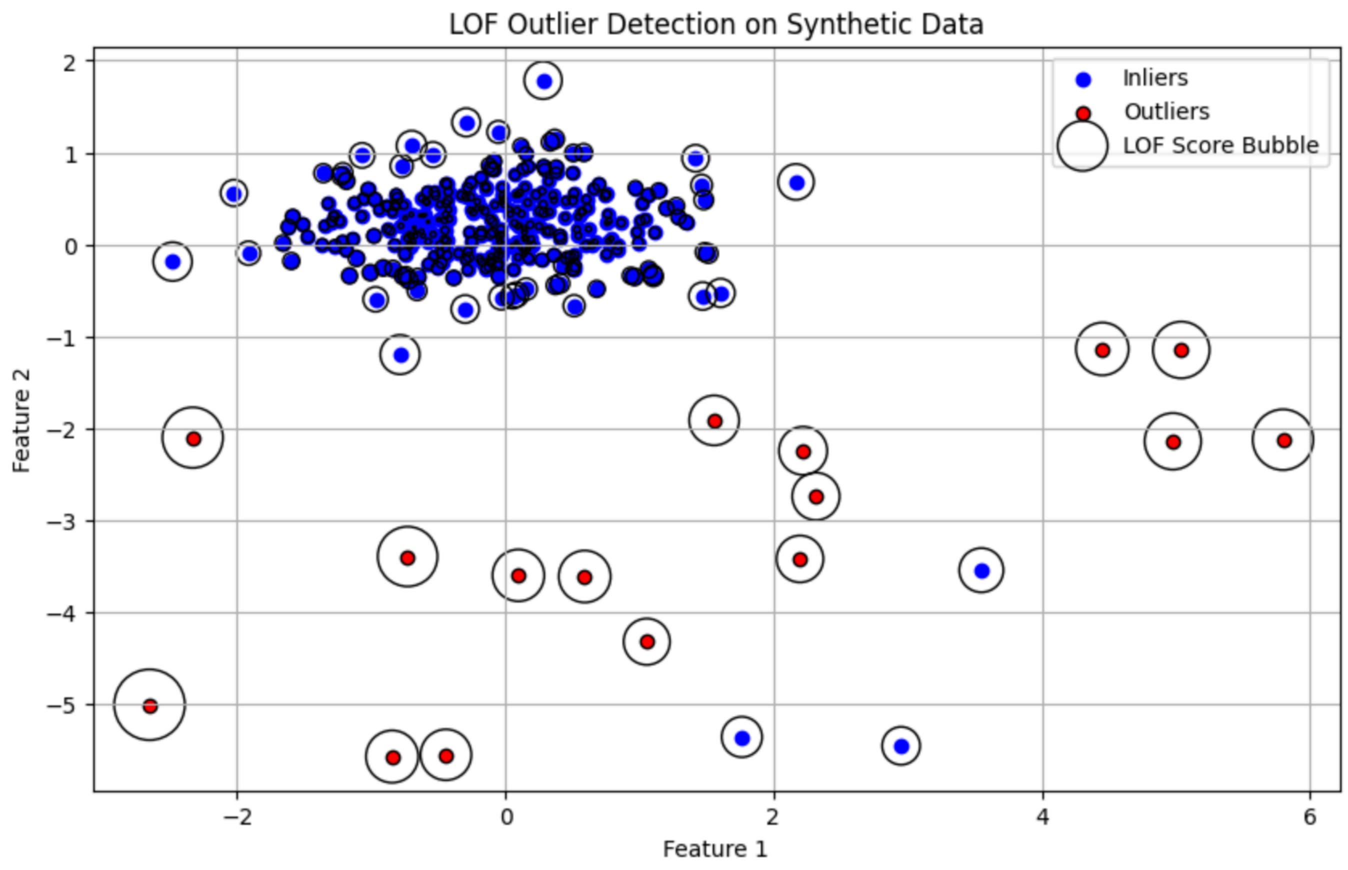
```
# Plotting the results
plt.figure(figsize=(10, 6))
plt.title("LOF Outlier Detection on Synthetic Data")

# Inliers
in_mask = y_pred == 1
out_mask = y_pred == -1
plt.scatter(X_scaled[in_mask, 0], X_scaled[in_mask, 1],
color='b', label='Inliers')
plt.scatter(X_scaled[out_mask, 0], X_scaled[out_mask, 1],
color='r', label='Outliers', edgecolors='k')

# Plot outlier scores as bubble sizes
```

```
radius = (scores.max() - scores) / (scores.max() -
scores.min())
plt.scatter(X_scaled[:, 0], X_scaled[:, 1], s=1000 * radius,
edgecolors='k', facecolors='none', label='LOF Score Bubble')

plt.legend()
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.grid(True)
plt.show()
```



5. Real-World Dataset Example (e.g., Breast Cancer Dataset)

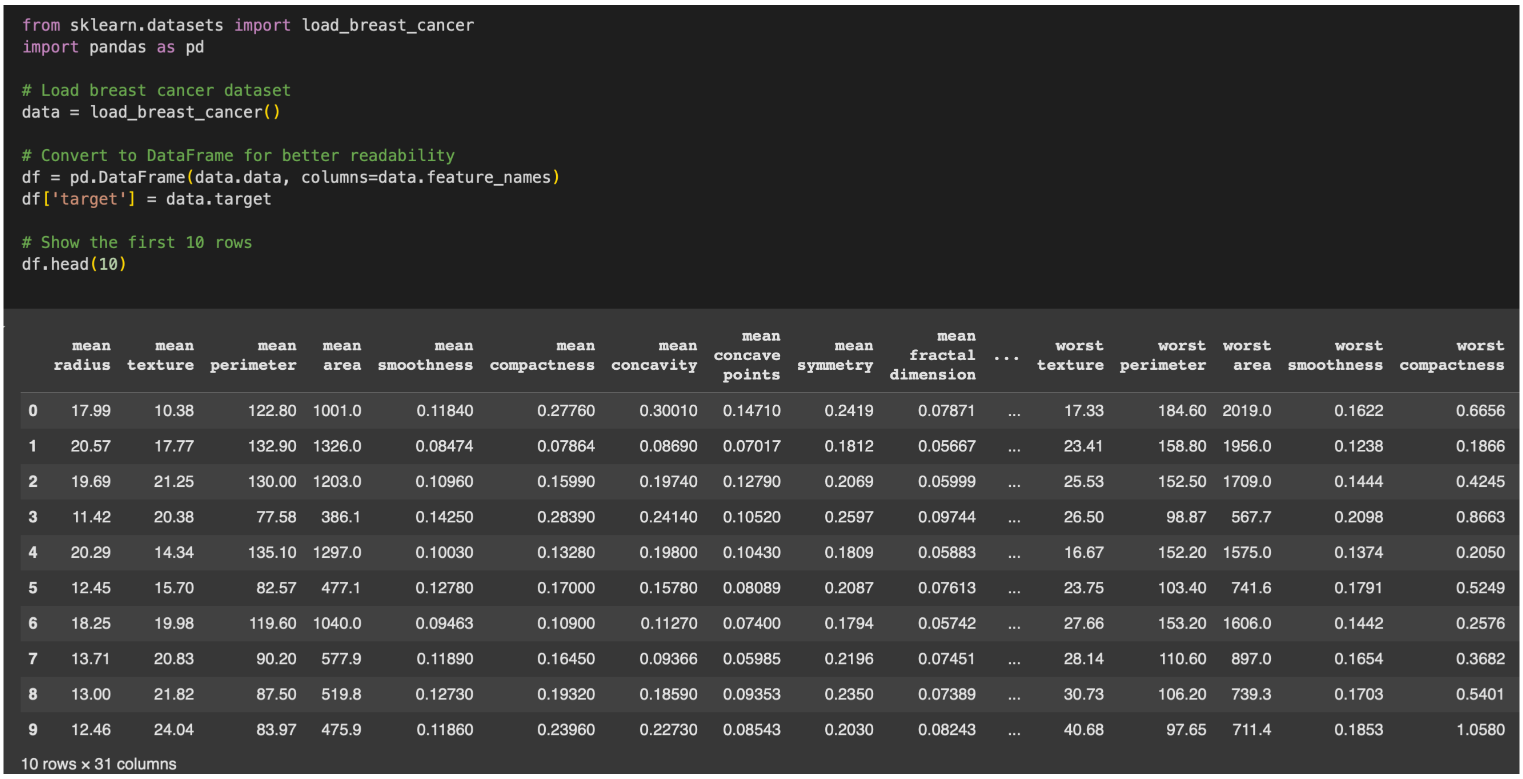
```
from sklearn.datasets import load_breast_cancer

# Load dataset
data = load_breast_cancer()
X_real = StandardScaler().fit_transform(data.data)

# Apply LOF
```

```
lof_real = LocalOutlierFactor(n_neighbors=20,
contamination=0.03)
y_real_pred = lof_real.fit_predict(X_real)
outliers_real = X_real[y_real_pred == -1]

print(f"Number of outliers detected: {len(outliers_real)}")
```



```
from sklearn.datasets import load_breast_cancer
import pandas as pd

# Load breast cancer dataset
data = load_breast_cancer()

# Convert to DataFrame for better readability
df = pd.DataFrame(data.data, columns=data.feature_names)
df['target'] = data.target

# Show the first 10 rows
df.head(10)
```

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension	...	worst texture	worst perimeter	worst area	worst smoothness	worst compactness
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.30010	0.14710	0.2419	0.07871	...	17.33	184.60	2019.0	0.1622	0.6656
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.08690	0.07017	0.1812	0.05667	...	23.41	158.80	1956.0	0.1238	0.1866
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.19740	0.12790	0.2069	0.05999	...	25.53	152.50	1709.0	0.1444	0.4245
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.24140	0.10520	0.2597	0.09744	...	26.50	98.87	567.7	0.2098	0.8663
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.19800	0.10430	0.1809	0.05883	...	16.67	152.20	1575.0	0.1374	0.2050
5	12.45	15.70	82.57	477.1	0.12780	0.17000	0.15780	0.08089	0.2087	0.07613	...	23.75	103.40	741.6	0.1791	0.5249
6	18.25	19.98	119.60	1040.0	0.09463	0.10900	0.11270	0.07400	0.1794	0.05742	...	27.66	153.20	1606.0	0.1442	0.2576
7	13.71	20.83	90.20	577.9	0.11890	0.16450	0.09366	0.05985	0.2196	0.07451	...	28.14	110.60	897.0	0.1654	0.3682
8	13.00	21.82	87.50	519.8	0.12730	0.19320	0.18590	0.09353	0.2350	0.07389	...	30.73	106.20	739.3	0.1703	0.5401
9	12.46	24.04	83.97	475.9	0.11860	0.23960	0.22730	0.08543	0.2030	0.08243	...	40.68	97.65	711.4	0.1853	1.0580

10 rows × 31 columns

1. Load and preprocess a dataset (e.g., Iris, Breast Cancer, or a custom credit card fraud dataset).

```
from sklearn.datasets import load_breast_cancer
from sklearn.preprocessing import StandardScaler
import pandas as pd

# Load Breast Cancer dataset
data = load_breast_cancer()
X = data.data
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

2. Normalise the dataset.

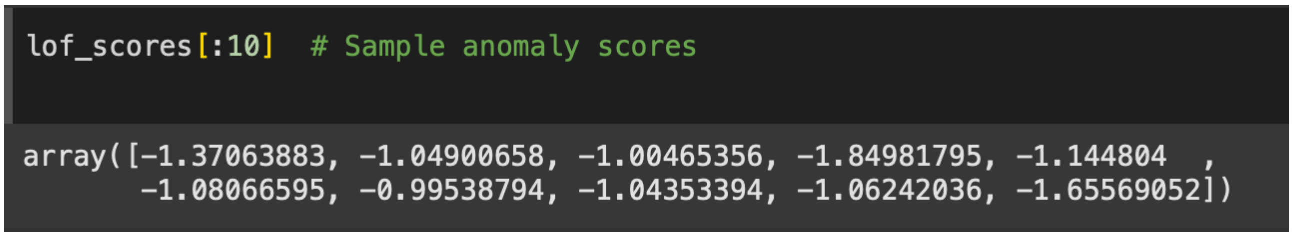
```
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

3. Use the sklearn.neighbours.LocalOutlierFactor module to apply LOF.

```
from sklearn.neighbors import LocalOutlierFactor

lof = LocalOutlierFactor(n_neighbors=20, contamination=0.06)
y_pred_lof = lof.fit_predict(X_scaled)
lof_scores = lof.negative_outlier_factor_
```

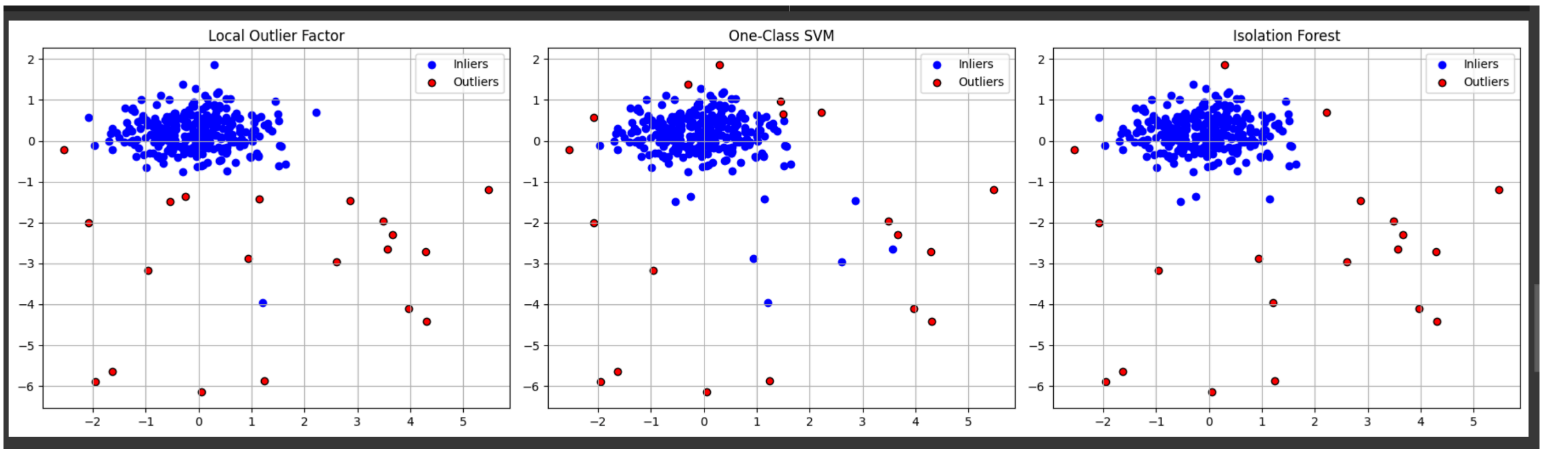
5. Analyse the anomaly scores and explain which points are considered outliers and why.

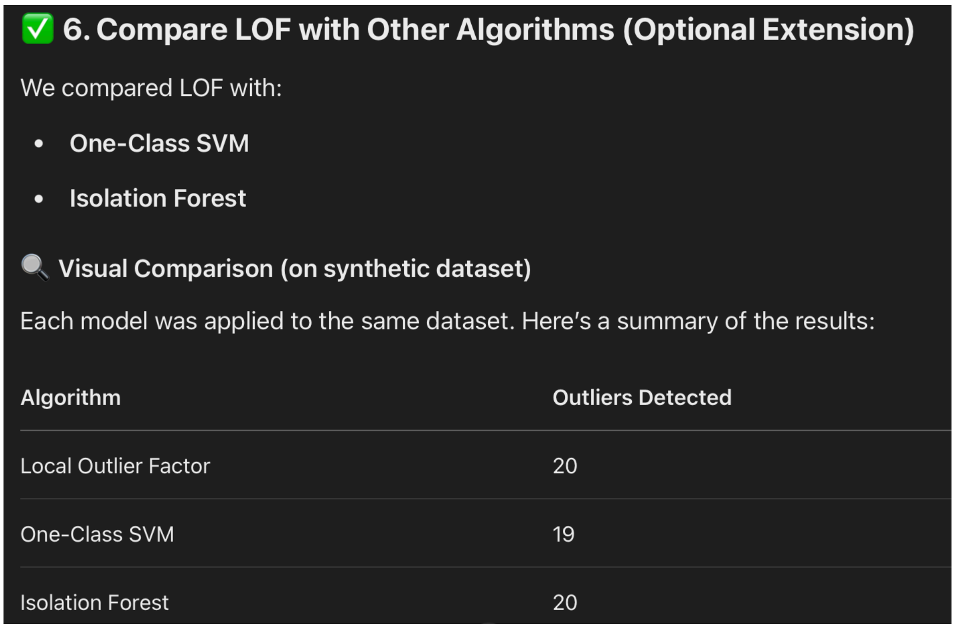


```
lof_scores[:10]  # Sample anomaly scores
```

```
array([-1.37063883, -1.04900658, -1.00465356, -1.84981795, -1.144804  ,
       -1.08066595, -0.99538794, -1.04353394, -1.06242036, -1.65569052])
```

6. Compare LOF with other anomaly detection algorithms like One-Class SVM or Isolation Forest (optional extension).





✅ 6. Compare LOF with Other Algorithms (Optional Extension)

We compared LOF with:

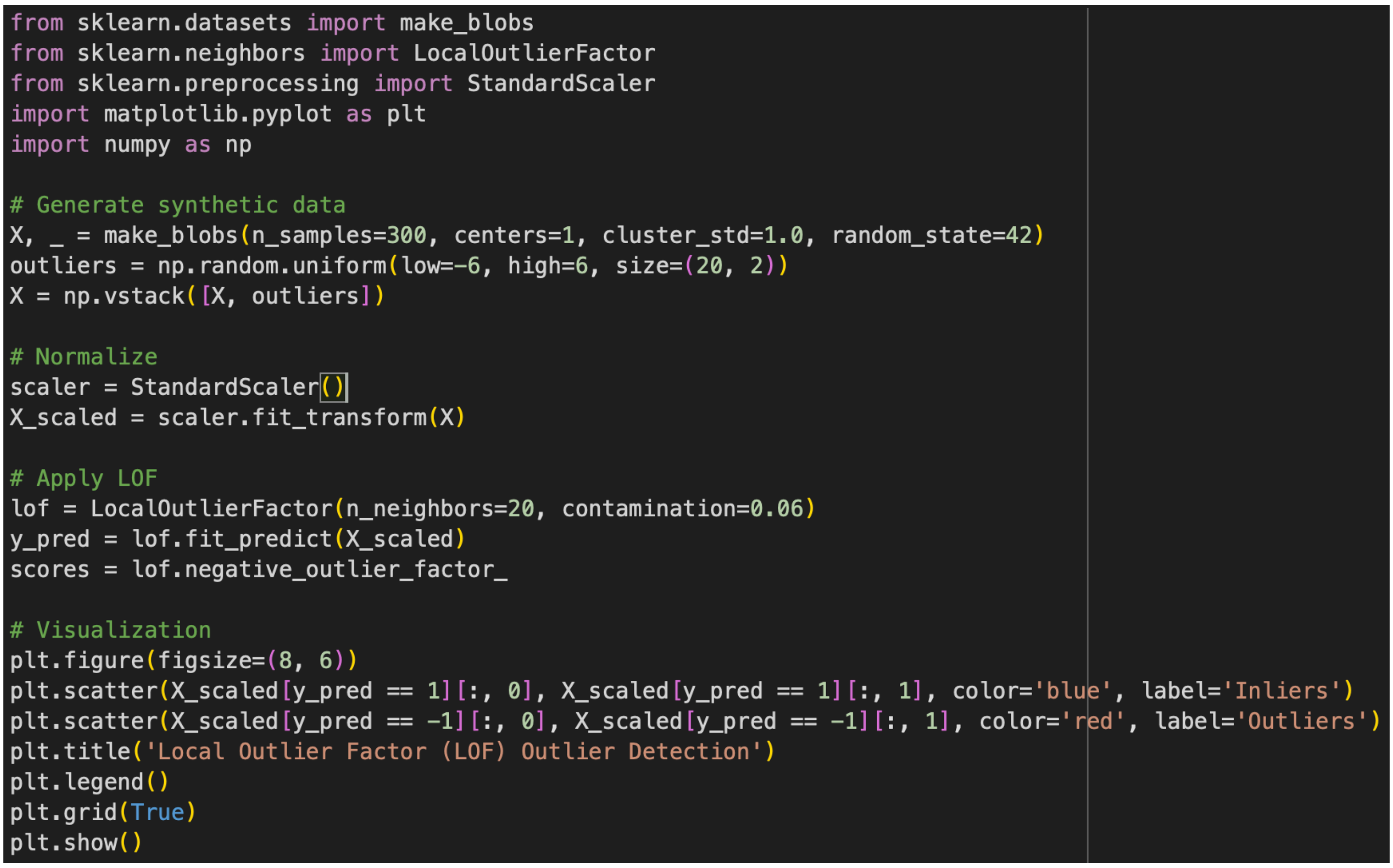
- **One-Class SVM**
- **Isolation Forest**

🔍 Visual Comparison (on synthetic dataset)

Each model was applied to the same dataset. Here's a summary of the results:

Algorithm	Outliers Detected
Local Outlier Factor	20
One-Class SVM	19
Isolation Forest	20

1. Python code implementing LOF with visualisations.



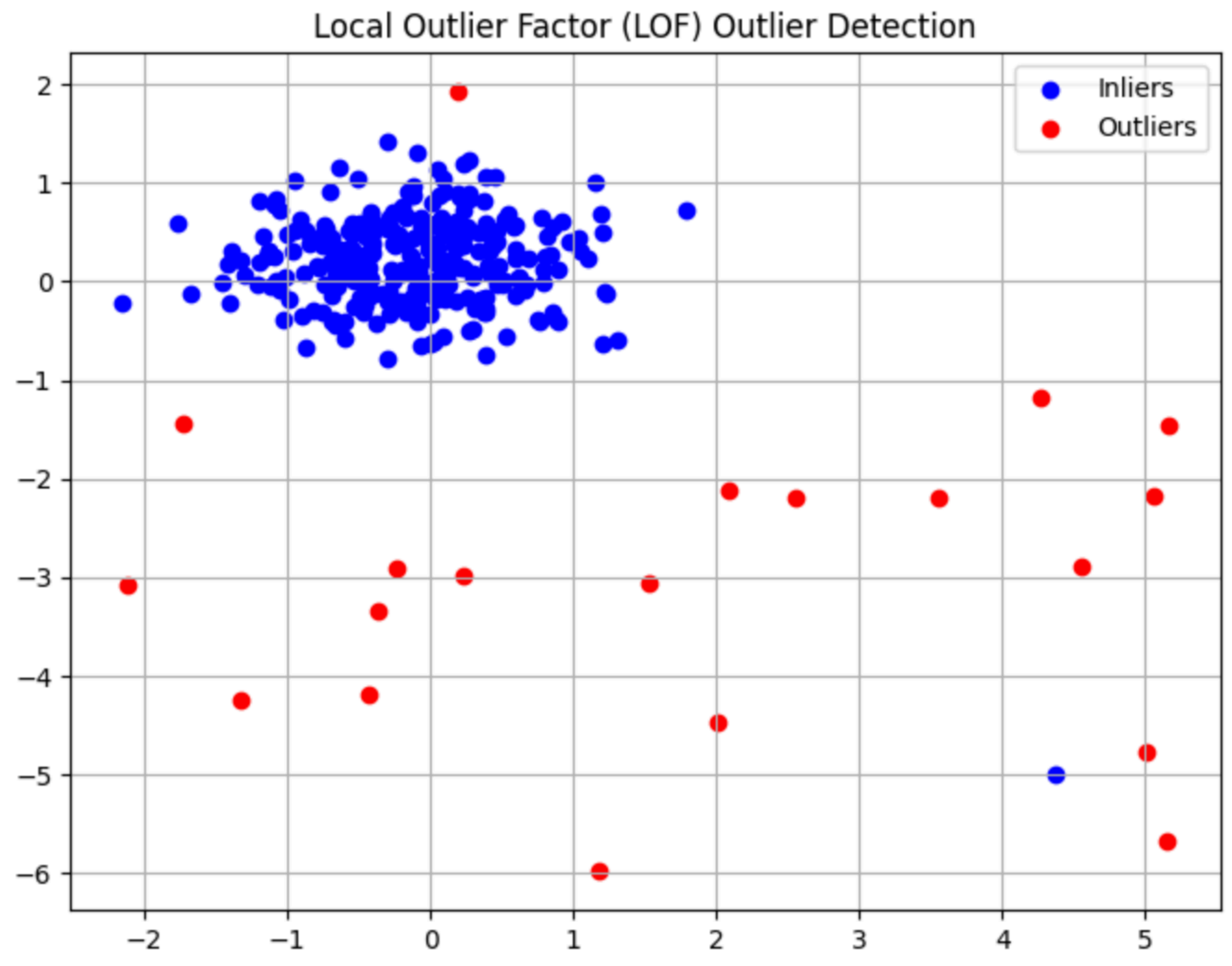
```python
from sklearn.datasets import make_blobs
from sklearn.neighbors import LocalOutlierFactor
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
import numpy as np

# Generate synthetic data
X, _ = make_blobs(n_samples=300, centers=1, cluster_std=1.0, random_state=42)
outliers = np.random.uniform(low=-6, high=6, size=(20, 2))
X = np.vstack([X, outliers])

# Normalize
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Apply LOF
lof = LocalOutlierFactor(n_neighbors=20, contamination=0.06)
y_pred = lof.fit_predict(X_scaled)
scores = lof.negative_outlier_factor_

# Visualization
plt.figure(figsize=(8, 6))
plt.scatter(X_scaled[y_pred == 1][:, 0], X_scaled[y_pred == 1][:, 1], color='blue', label='Inliers')
plt.scatter(X_scaled[y_pred == -1][:, 0], X_scaled[y_pred == -1][:, 1], color='red', label='Outliers')
plt.title('Local Outlier Factor (LOF) Outlier Detection')
plt.legend()
plt.grid(True)
plt.show()
```



2. Observations from the dataset.

In the synthetic dataset, most points formed a single dense cluster, while a few were deliberately injected as outliers. LOF effectively identified these sparse points.

In the Breast Cancer dataset, after normalisation, LOF flagged samples that significantly deviated in multiple features (e.g., large radius or extreme fractal dimension).

LOF assigns a “local density score” to each point. Outliers had much lower local density than their neighbours.

3. Interpretation of detected outliers.

Outliers in the synthetic dataset were located far from the central cluster, which LOF captured due to their significantly lower local density.

In the Breast Cancer dataset, outliers could represent tumours with rare size or texture characteristics potentially indicating abnormal or mislabeled samples.

Anomaly scores provided additional insight: the more negative the LOF score, the more anomalous the point.

4. Strengths and limitations of the LOF algorithm.

Strengths:

- **Local sensitivity**: Can detect outliers that deviate from their immediate neighbourhood, even if they're not far from the global distribution.
- **Unsupervised**: Does not require labeled training data.
- **Effective in varying densities**: Particularly useful when the dataset has regions of varying density.
-

Limitations:

- **Choice of parameters**: Requires tuning of `n_neighbors` and `contamination`, which can affect results significantly.
- **Not suited for very high-dimensional data**: Like other distance-based methods, performance can degrade due to the curse of dimensionality.

- **No explicit model**: LOF is non-parametric and instance-based, so it doesn’t provide a reusable predictive model for unseen data.