Outlier Detection

Name: Soumyadeep Ganguly

Reg No: 24MDT0082

In [65]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.neighbors import NearestNeighbors
from scipy.stats import chi2, zscore
```

In [66]:
```
np.random.seed(42)

n_samples = 300

mean = [0,0]
cov = [[1, 0.5], [0.5, 1]]
```

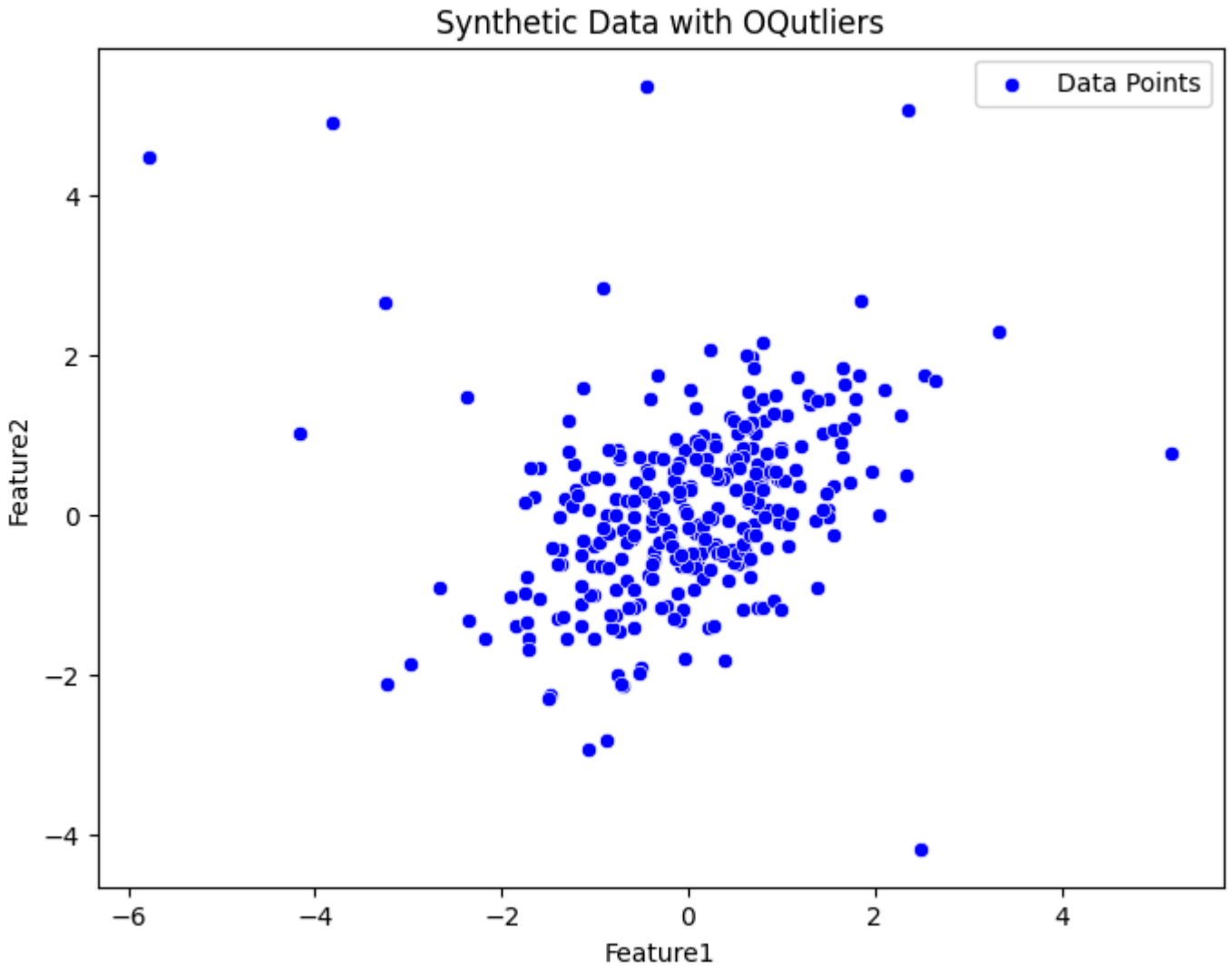
In [67]:
```
data_normal = np.random.multivariate_normal(mean, cov, n_samples)

n_outliers = 15
data_outliers = np.random.uniform(low=-6, high=6, size=(n_outliers, 2))

data_all = np.vstack([data_normal, data_outliers])
df = pd.DataFrame(data_all, columns=[ 'Feature1', 'Feature2'])


# Plot the data to visualize the generated points and the injected outliers
plt.figure(figsize=(8, 6)) # Set the size of the plot
sns.scatterplot(x='Feature1', y='Feature2', data=df, color='blue', label='Data Points')
# Create a scatter plot
```

```
plt.title('Synthetic Data with OQutliers') # Title of the plot
plt.xlabel('Feature1') # Label for the x-axis
plt.ylabel('Feature2') # Label for the y-axis
plt.legend() # Display legend on the plot
plt.show() # Render the plot
```



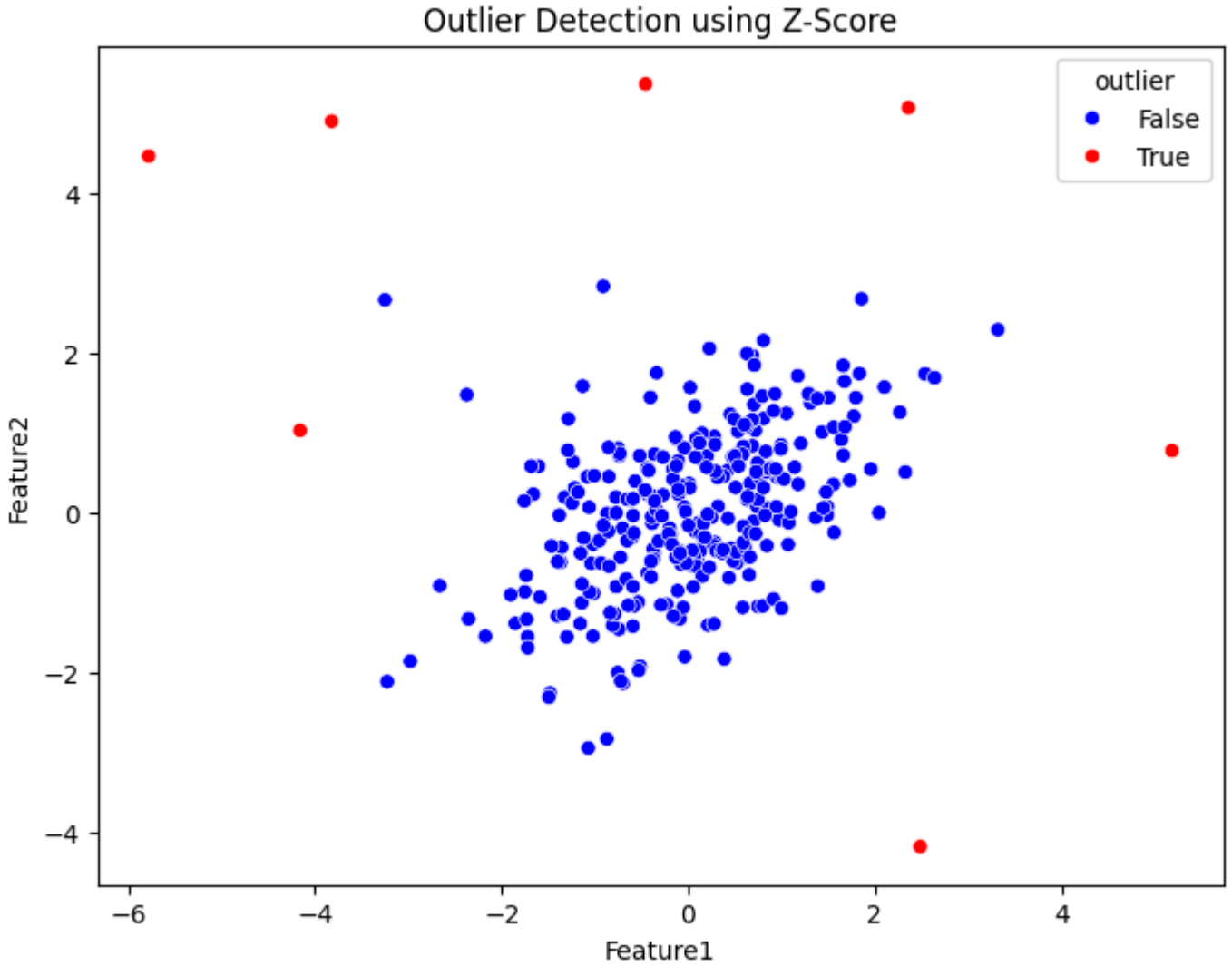
Z Score Method

In [68]:
```
df_z = df.copy()
df_z['Z Feature1'] = zscore(df[ 'Feature1'])
df_z['Z Feature2'] = zscore(df['Feature2'])
```

In [69]:
```
threshold = 3
df_z['Outlier_z'] = (df_z['Z Feature1'].abs() > threshold) | (df_z['Z Feature2'].abs() > threshold)
```

In [70]:
```
# Plot the results to visualize outliers detected by the Z-score method.
plt.figure(figsize=(8, 6))
sns.scatterplot(x='Feature1', y='Feature2', data=df_z, hue='Outlier_z',
palette={False: 'blue', True: 'red'})
plt.title('Outlier Detection using Z-Score')
plt.xlabel('Feature1')
plt.ylabel('Feature2')
plt.legend(title='outlier')
plt.show()

print('Z-Score method detected outliers:', df_z['Outlier_z'].sum())
```




```
Z-Score method detected outliers: 7
```

IQR Method

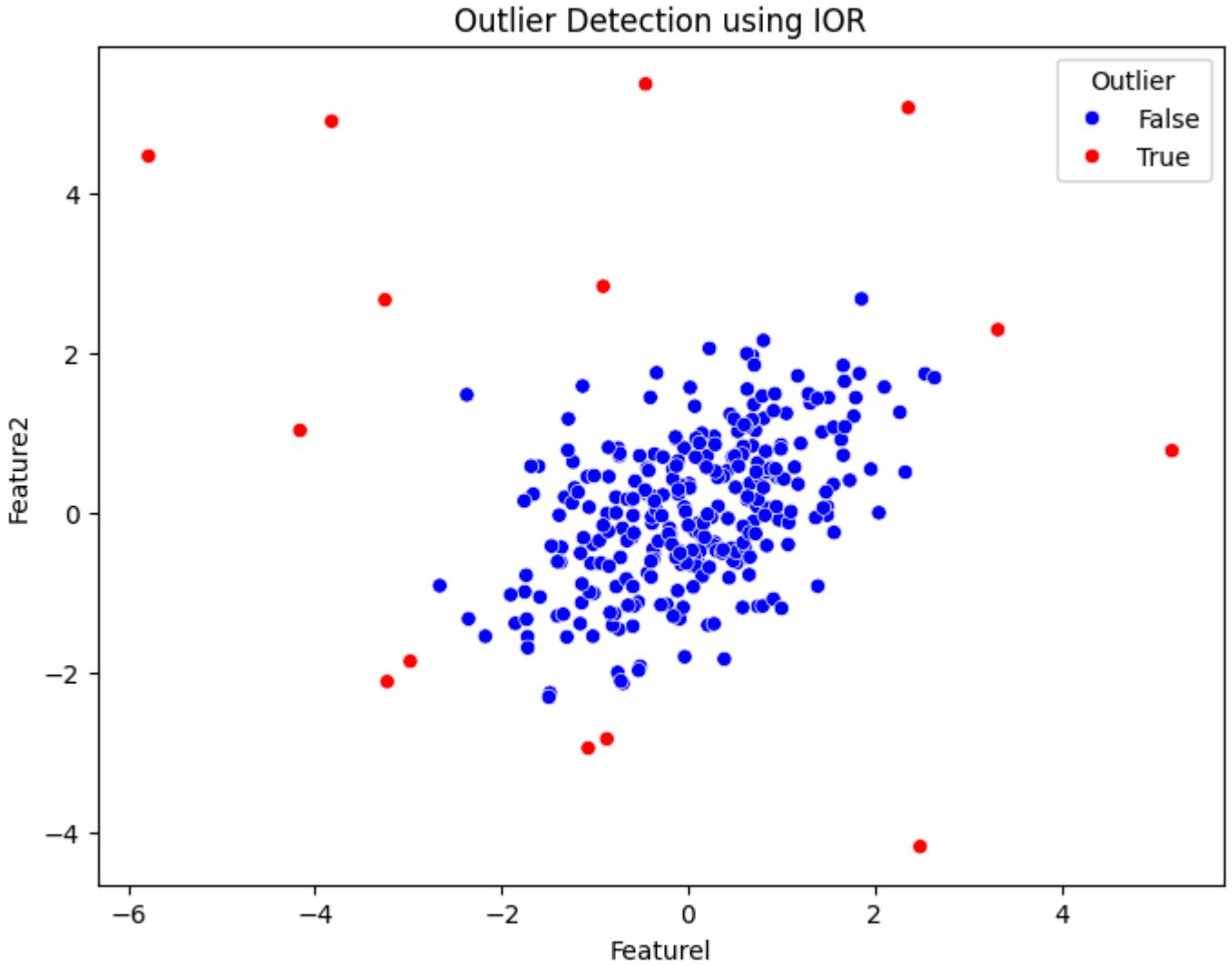
In [71]:
```
Q1 = df.quantile(0.25)
Q3 = df.quantile(0.75)
```

```
IQR =Q3 - Q1

lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

outlier_iqr = ((df < lower_bound) | (df > upper_bound)).any(axis=1)
df['outlier IQR'] = outlier_iqr


# Plot the results to visualize outliers detected by the IOR method.
plt. figure(figsize=(8, 6))
sns.scatterplot (x='Feature1', y='Feature2', data=df, hue='outlier IQR', palette={False: 'blue', True:'red'})
plt.title('Outlier Detection using IOR')
plt.xlabel('Feature1')
plt.ylabel('Feature2')
plt.legend(title='Outlier')
plt.show()
```



In [72]:
```
print('IQR method detected outliers:', df['outlier IQR'].sum())
```

```
IQR method detected outliers: 14
```

KNN

In [73]:
```
k = 5

nbrs = NearestNeighbors(n_neighbors=k+1)
nbrs.fit(df[['Feature1', 'Feature2']])
```

Out[73]:
```
NearestNeighbors
NearestNeighbors(n_neighbors=6)
```

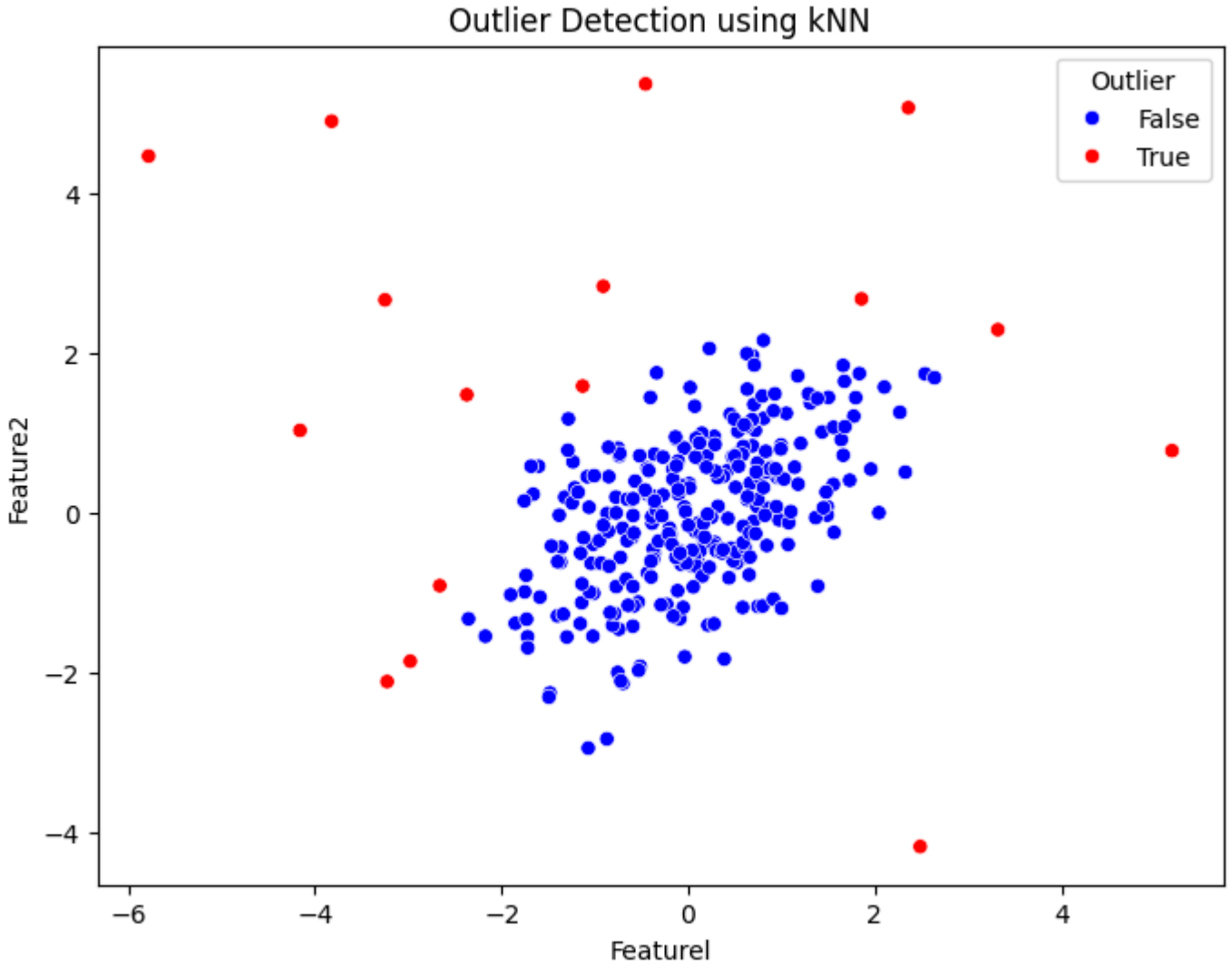
In [74]:
```
distances, indices = nbrs.kneighbors(df[['Feature1', 'Feature2']])

avg_distance = distances[:, 1:].mean(axis=1)
```

In [75]:
```
df['Avg_KIN_Distance'] = avg_distance
```

In [76]:
```
# Set a threshold for outlier detection based on the 95th percentile of the average distances.
threshold_knn = np.percentile(avg_distance, 95) # Flag points as outliers if their average distance exceeds the threshold.
df['Outlier_KNN'] = df['Avg_KIN_Distance'] > threshold_knn
```

In [77]:
```
plt.figure(figsize=(8, 6))
sns.scatterplot(x='Feature1', y='Feature2', data=df, hue='Outlier_KNN', palette={False:'blue', True:'red'})
plt.title('Outlier Detection using kNN')
plt.xlabel('Feature1')
plt.ylabel('Feature2')
plt.legend(title='Outlier')
plt.show()
```



In [78]:
```
print( 'Kl nethod detected outliers:', df['Outlier_KNN'].sum())
```

```
Kl nethod detected outliers: 16
```

Mahalanobis Distance

In [79]:
```
def mahalanobis_distance(x=None, data=None, cov_inv=None):
    if cov_inv is None:
        cov = np.cov(data.T)
        cov_inv = np.linalg.inv(cov)

    x_minus_mu = x - np.mean(data, ax1s=0)
    left_term = np.dot(x_minus_mu, cov_inv)
    mahal = np.dot(left_term, x_minus_mu.T)

    return mahal.diagonal() if mahal.ndim > 0 else mahal

cov_matrix = np.cov(df[['Feature1', 'Feature2']].values.T)
cov_inv = np. linalg.inv(cov_matrix)

m_dist = [] # Calculate the mean of the features to center the data.
mean_d = df[['Feature1', 'Feature2']].mean().values
```

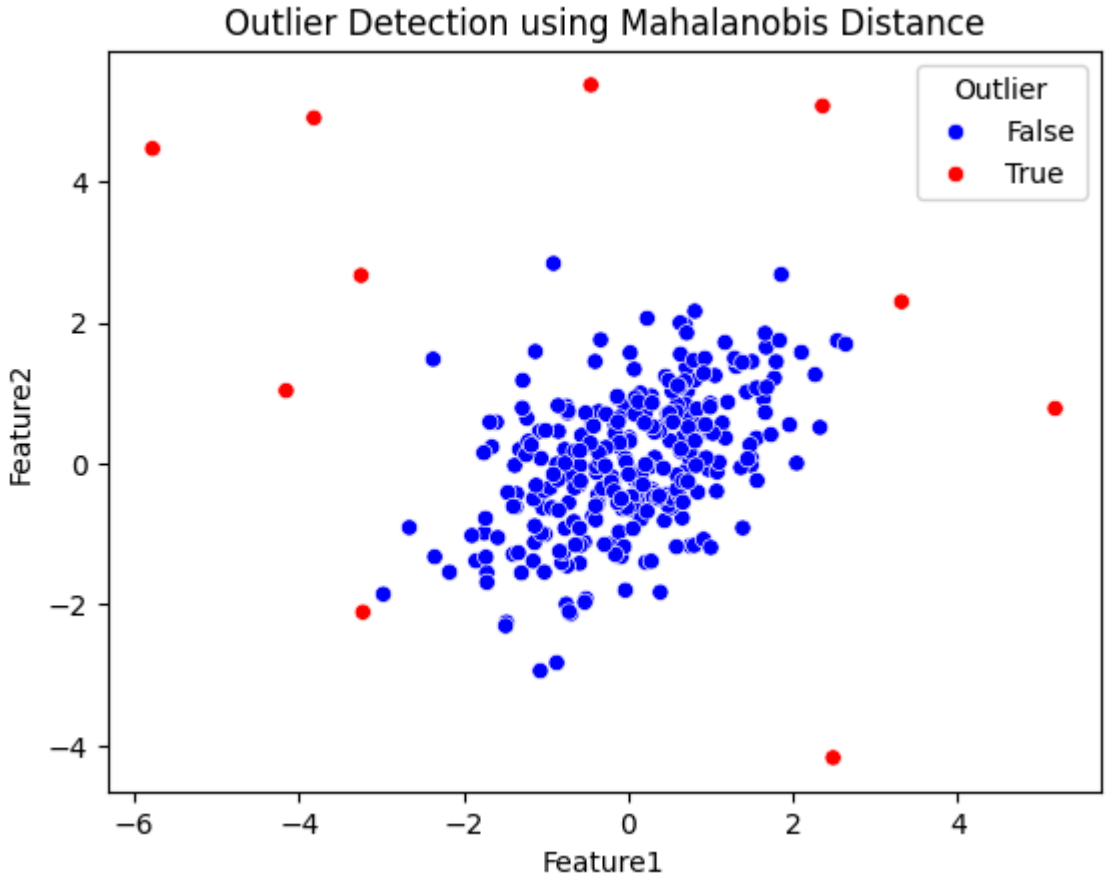
In [80]:
```
for i, row in df[['Feature1', 'Feature2']].iterrows():
    diff = row.values - mean_d

    md = np.sqrt(np.dot(np.dot(diff.T, cov_inv), diff))
    m_dist.append(md)
```

In [81]:
```
df['Mahalanobis'] = m_dist

dof = 2
alpha = 0.99
threshold_maha = np.sqrt(chi2.ppf(alpha, dof))
df['Outlier Mahalanobis'] = df['Mahalanobis'] > threshold_maha
```

In [82]:
```
plt.Figure(figsize=(8, 6))
sns.scatterplot (x='Feature1', y='Feature2', data=df, hue='Outlier Mahalanobis', palette={False:'blue', True:'red'})
plt.title('Outlier Detection using Mahalanobis Distance')
plt.xlabel('Feature1')
plt.ylabel('Feature2')
plt.legend (title="Outlier")
plt.show()
```



In [83]:
```
print('Mahalanobis method detected outliers:', df['Outlier Mahalanobis'].sum())
```

```
Mahalanobis method detected outliers: 10
```

In [ ]: