Here is the README file in the requested format:

Unsupervised Anomaly Detection using Isolation Forest

Overview

This project implements **Anomaly Detection using Isolation Forest**, an unsupervised learning algorithm that detects anomalies or outliers by isolating observations through recursive partitioning. The goal is to identify anomalies in the dataset by using the **Isolation Forest** technique.

Key Features

1. **Data Loading**:
    - The dataset is loaded using **pandas**, which allows for efficient handling and manipulation of the data.
2. **Feature Selection**:
    - Relevant features for anomaly detection are selected, which are critical in determining whether data points are anomalies.
3. **Isolation Forest**:
    - **Isolation Forest** is used to detect anomalies. It works by recursively partitioning the data and isolating the anomalies, which are few and different from the rest of the data.
4. **Anomaly Detection**:
    - The algorithm identifies anomalies and assigns them a label of `-1`, while normal points are labeled as `1`.
5. **Visualization**:
    - A scatter plot visualizes the detected anomalies, where normal points are marked in blue and anomalies in red.

How It Works

1. **Data Loading**:
    - The dataset is loaded using **pandas** `read_csv` function, and specific features are selected for analysis.
2. **Anomaly Detection**:
    - The **Isolation Forest** model is initialized and trained using the dataset.
    - The model predicts anomalies in the dataset based on its inherent structure.
3. **Visualization**:
    - The results are visualized using **Matplotlib**, where anomalies are plotted separately from normal points for clarity.

Code Walkthrough

1. **Data Loading and Feature Selection**:

```
data = pd.read_csv('your_dataset.csv')
X = data[['feature1', 'feature2']]  # Replace with relevant feature columns
```

2. **Isolation Forest Initialization and Fitting**:

```
iso_forest = IsolationForest(n_estimators=100, contamination=0.05, random_state=42
data['anomaly'] = iso_forest.fit_predict(X)
```

3. **Anomaly Detection**:

   - Anomalies are identified and stored:

```
anomalies = data[data['anomaly'] == -1]
print("Number of anomalies detected:", len(anomalies))
```

4. **Visualization**:

```
plt.scatter(X['feature1'], X['feature2'], label='Normal', c='blue', s=20)
plt.scatter(anomalies['feature1'], anomalies['feature2'], label='Anomaly', c='red',
plt.title('Isolation Forest Anomaly Detection')
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.legend()
plt.show()
```

Advantages

- **Efficient and Scalable**: Works well with high-dimensional data and large datasets.
- **No Assumptions About Data**: The model doesn't make assumptions about the data distribution, making it suitable for complex real-world datasets.
- **Fast**: Performs anomaly detection efficiently, even for large datasets.

Future Work

- **Extend to Multivariate Data**: Adapt the model to work with high-dimensional or multivariate datasets.
- **Parameter Tuning**: Experiment with different values for hyperparameters like `n_estimators` and `contamination` to optimize performance.
- **Comparison with Other Algorithms**: Compare the performance of Isolation Forest with other anomaly detection algorithms, like One-Class SVM or DBSCAN.

References

- [Isolation Forest - scikit-learn documentation]
- [Anomaly Detection using Isolation Forest - GeeksforGeeks]