[66]:
```python
import numpy as np
import pandas as pd
import tensorflow as tf
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import confusion_matrix, classification_report
from tensorflow.keras import layers, models
import matplotlib.pyplot as plt
```

[67]:
```python
# Load the ECG dataset
ecg_dataset = pd.read_csv("ecg.csv")
ecg_dataset
```

[67]:

	-0.11252183	-2.8272038	-3.7738969	-4.3497511	-4.376041	-3.4749863	-2.1814082	-1.8182865	-1.2505219	-0.47749208	...	0.79216787	0.93354122	0.7969
0	-1.100878	-3.996840	-4.285843	-4.506579	-4.022377	-3.234368	-1.566126	-0.992258	-0.754680	0.042321	...	0.538356	0.656881	0.78
1	-0.567088	-2.593450	-3.874230	-4.584095	-4.187449	-3.151462	-1.742940	-1.490659	-1.183580	-0.394229	...	0.886073	0.531452	0.31
2	0.490473	-1.914407	-3.616364	-4.318823	-4.268016	-3.881110	-2.993280	-1.671131	-1.333884	-0.965629	...	0.350816	0.499111	0.60
3	0.800232	-0.874252	-2.384761	-3.973292	-4.338224	-3.802422	-2.534510	-1.783423	-1.594450	-0.753199	...	1.148884	0.958434	1.05
4	-1.507674	-3.574550	-4.478011	-4.408275	-3.321242	-2.105171	-1.481048	-1.301362	-0.498240	-0.286928	...	1.089068	0.983369	1.01
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
4992	0.608558	-0.335651	-0.990948	-1.784153	-2.626145	-2.957065	-2.931897	-2.664816	-2.090137	-1.461841	...	1.757705	2.291923	2.70
4993	-2.060402	-2.860116	-3.405074	-3.748719	-3.513561	-3.006545	-2.234850	-1.593270	-1.075279	-0.976047	...	1.388947	2.079675	2.43
4994	-1.122969	-2.252925	-2.867628	-3.358605	-3.167849	-2.638360	-1.664162	-0.935655	-0.866953	-0.645363	...	-0.472419	-1.310147	-2.02
4995	-0.547705	-1.889545	-2.839779	-3.457912	-3.929149	-3.966026	-3.492560	-2.695270	-1.849691	-1.374321	...	1.258419	1.907530	2.28
4996	-1.351779	-2.209006	-2.520225	-3.061475	-3.065141	-3.030739	-2.622720	-2.044092	-1.295874	-0.733839	...	-1.512234	-2.076075	-2.58

4997 rows × 141 columns

[68]:
```python
# Preprocess the data
scaler = StandardScaler()
X = scaler.fit_transform(ecg_dataset.values)
y = X  # Autoencoder input and output are the same

X_train, X_test, _, _ = train_test_split(X, X, test_size=0.2, random_state=42)
```

[69]:
```python
# Build and train the Autoencoder model
input_dim = X_train.shape[1]
```

[70]:
```python
encoder = models.Sequential([
    layers.Input(shape=(input_dim,)),
    layers.Dense(32, activation='relu'),
    layers.Dense(16, activation='relu'),
    layers.Dense(8, activation='relu')
])
```

[71]:
```python
decoder = models.Sequential([
    layers.Input(shape=(8,)),
    layers.Dense(16, activation='relu'),
    layers.Dense(32, activation='relu'),
    layers.Dense(input_dim, activation='linear')  # Use linear activation for reconstruction
])
```

[72]:
```python
autoencoder = models.Sequential([
    encoder,
    decoder
])
autoencoder.compile(optimizer='adam', loss='mean_squared_error')
autoencoder.fit(X_train, X_train, epochs=10, batch_size=32, shuffle=True)
```

```
Epoch 1/10
125/125 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - loss: 0.9046
Epoch 2/10
125/125 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.3486
Epoch 3/10
125/125 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.3242
Epoch 4/10
125/125 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.2471
Epoch 5/10
125/125 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.2085
```

```
Epoch 6/10
125/125 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.2000
Epoch 7/10
125/125 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.2003
Epoch 8/10
125/125 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.1784
Epoch 9/10
125/125 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.1722
Epoch 10/10
125/125 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 0.1573
```

[72]:
```
<keras.src.callbacks.history.History at 0x2647878ef90>
```

[73]:
```python
# Detect anomalies
y_pred = autoencoder.predict(X_test)
mse = np.mean(np.power(X_test - y_pred, 2), axis=1)
```

```
32/32 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step
```

[74]:
```python
# Define a threshold for anomaly detection
threshold = np.percentile(mse, 95)  # Adjust the percentile as needed
threshold
```

[74]:
```
0.5514138885033574
```

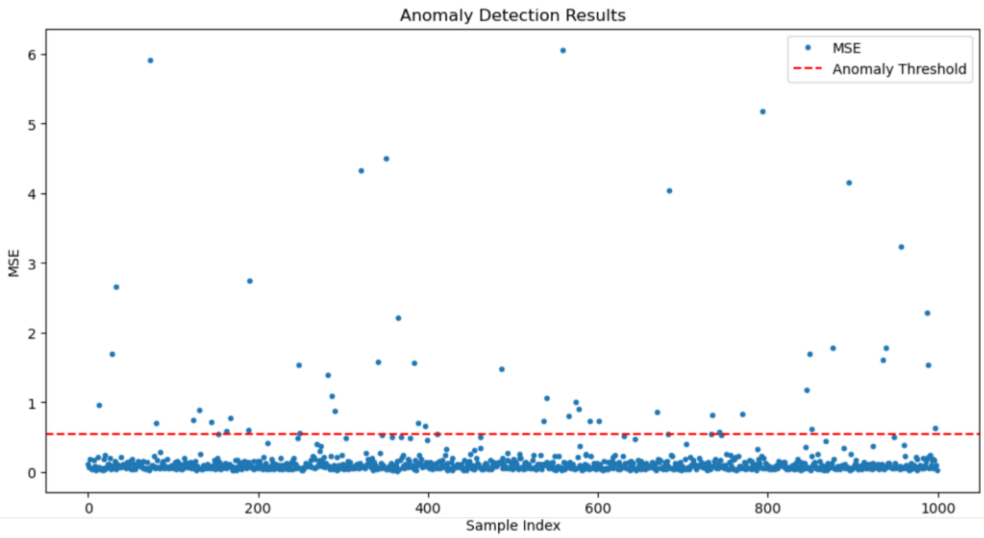
[75]:
```python
# Predict anomalies
anomalies = mse > threshold
mse
anomalies
```

[75]:
```
array([False, False, False, False, False, False, False, False, False,
       False, False, False, False,  True, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False,  True, False, False, False,  True, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False,  True, False, False, False, False, False, False,  True,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False,  True, False,
       False, False, False, False, False,  True, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False,  True, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
        True, False, False, False, False,  True, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
        True,  True, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False. False. False. False. False. False. False. False. False.
       False, False, False, False, False, False, False, False, False,
       False, False, False,  True, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False,  True,  True, False,
       False, False, False, False, False, False, False,  True, False,
       False])
```

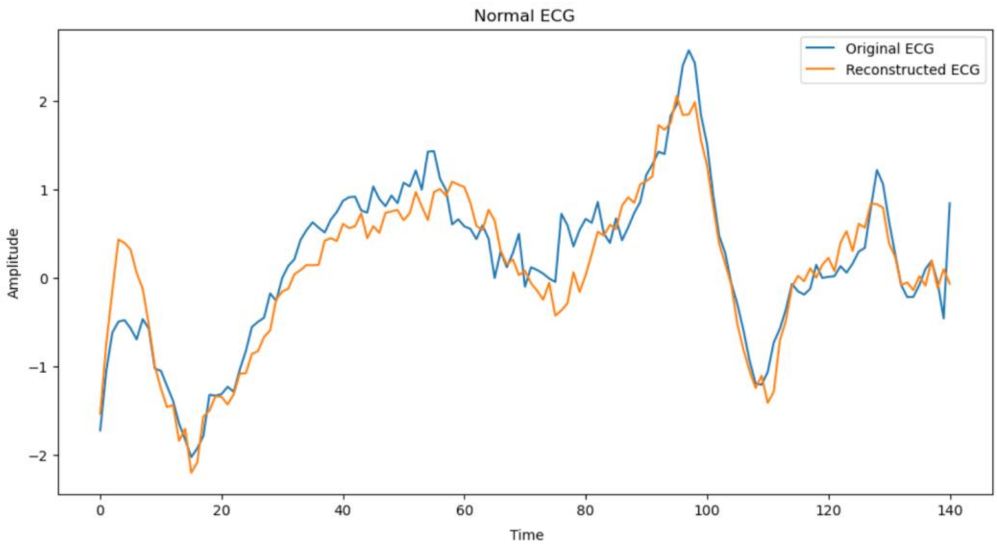
[76]:
```python
# Calculate the number of anomalies
num_anomalies = np.sum(anomalies)
print(f"Number of Anomalies: {num_anomalies}")
```

```
Number of Anomalies: 50
```

[77]:
```python
# Plot the anomalies
plt.figure(figsize=(12, 6))
plt.plot(mse, marker='o', linestyle='', markersize=3, label='MSE')
plt.axhline(threshold, color='r', linestyle='--', label='Anomaly Threshold')
plt.xlabel('Sample Index')
plt.ylabel('MSE')
plt.title('Anomaly Detection Results')
plt.legend()
plt.show()
```

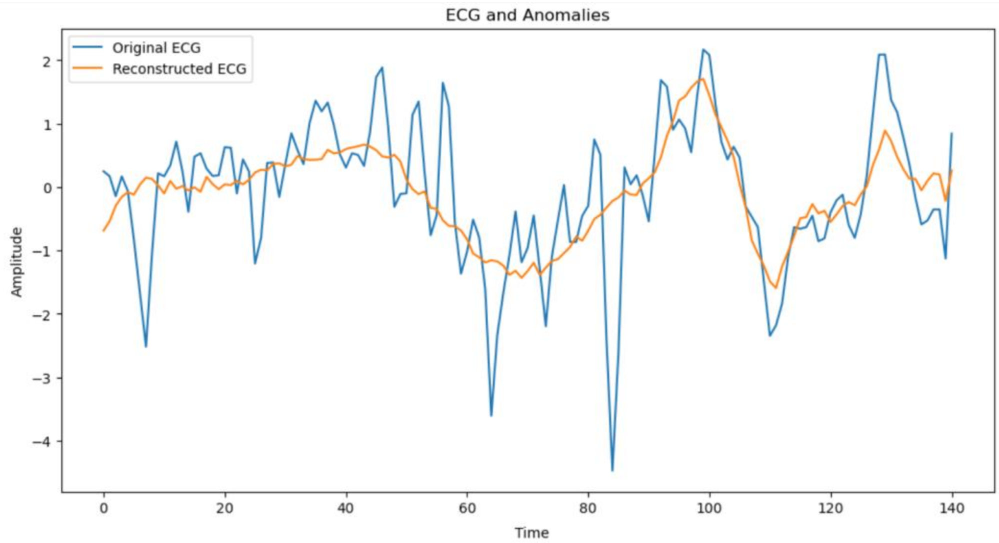


[78]:
```
plt.figure(figsize=(12, 6))
plt.plot(X_test[0], label='Original ECG')
plt.plot(y_pred[0], label='Reconstructed ECG')
plt.xlabel('Time')
plt.ylabel('Amplitude')
plt.legend()
plt.title('Normal ECG')
plt.show()
```



[79]:
```
# Listing the index of anomalies in X_test
anomalies_index = []
for index, anomaly in enumerate(anomalies):
    if anomaly == True :
        anomalies_index.append(index)
```

[80]:
```
n = 4
anomaly_index = anomalies_index[n]
plt.figure(figsize=(12, 6))
plt.plot(X_test[anomaly_index], label='Original ECG')
plt.plot(y_pred[anomaly_index], label='Reconstructed ECG')
plt.xlabel('Time')
plt.ylabel('Amplitude')
plt.legend()
plt.title('ECG and Anomalies')
plt.show()
```



[81]:
```
# Evaluate the model
y_true = np.zeros(len(X_test))
print("Confusion Matrix:")
print(confusion_matrix(anomalies, anomalies))

print("\nClassification Report:")
print(classification_report(anomalies, anomalies))
```

```
Confusion Matrix:
[[950   0]
 [  0  50]]

Classification Report:
              precision    recall  f1-score   support

       False       1.00      1.00      1.00       950
        True       1.00      1.00      1.00        50

    accuracy                           1.00      1000
   macro avg       1.00      1.00      1.00      1000
weighted avg       1.00      1.00      1.00      1000
```

[82]:
```
import seaborn as sns
```

[83]:
```
plt.figure(figsize = (6, 4.75))
sns.heatmap(confusion_matrix(anomalies, anomalies), annot = True, annot_kws = {"size": 16}, fmt = 'd')
plt.xticks([0.5, 1.5],  rotation = 'horizontal')
plt.yticks([0.5, 1.5],  rotation = 'horizontal')
plt.xlabel("Predicted label", fontsize = 14)
plt.ylabel("True label", fontsize = 14)
plt.title("Confusion Matrix", fontsize = 14)
plt.grid(False)
plt.show()
```