```
!pip install tensorflow scikit-learn pandas openpyxl matplotlib
```

```
Requirement already satisfied: tensorflow in c:\users\laptop land\anaconda3\lib\site-packages (2.19.0)
Requirement already satisfied: scikit-learn in c:\users\laptop land\anaconda3\lib\site-packages (1.5.1)
Requirement already satisfied: pandas in c:\users\laptop land\anaconda3\lib\site-packages (2.2.2)
Requirement already satisfied: openpyxl in c:\users\laptop land\anaconda3\lib\site-packages (3.1.5)
Requirement already satisfied: matplotlib in c:\users\laptop land\anaconda3\lib\site-packages (3.9.2)
Requirement already satisfied: absl-py>=1.0.0 in c:\users\laptop land\anaconda3\lib\site-packages (from tensorflow) (2.2.2)
Requirement already satisfied: astunparse>=1.6.0 in c:\users\laptop land\anaconda3\lib\site-packages (from tensorflow) (1.6.3)
Requirement already satisfied: flatbuffers>=24.3.25 in c:\users\laptop land\anaconda3\lib\site-packages (from tensorflow) (25.2.10)
Requirement already satisfied: gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 in c:\users\laptop land\anaconda3\lib\site-packages (from tensorflow)
Requirement already satisfied: google-pasta>=0.1.1 in c:\users\laptop land\anaconda3\lib\site-packages (from tensorflow) (0.2.0)
Requirement already satisfied: libclang>=13.0.0 in c:\users\laptop land\anaconda3\lib\site-packages (from tensorflow) (18.1.1)
Requirement already satisfied: opt-einsum>=2.3.2 in c:\users\laptop land\anaconda3\lib\site-packages (from tensorflow) (3.4.0)
Requirement already satisfied: packaging in c:\users\laptop land\anaconda3\lib\site-packages (from tensorflow) (24.1)
Requirement already satisfied: protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<6.0.0dev,>=3.20.3 in c:\users\laptop land\
Requirement already satisfied: requests<3,>=2.21.0 in c:\users\laptop land\anaconda3\lib\site-packages (from tensorflow) (2.32.3)
Requirement already satisfied: setuptools in c:\users\laptop land\anaconda3\lib\site-packages (from tensorflow) (75.1.0)
Requirement already satisfied: six>=1.12.0 in c:\users\laptop land\anaconda3\lib\site-packages (from tensorflow) (1.16.0)
Requirement already satisfied: termcolor>=1.1.0 in c:\users\laptop land\anaconda3\lib\site-packages (from tensorflow) (3.1.0)
Requirement already satisfied: typing-extensions>=3.6.6 in c:\users\laptop land\anaconda3\lib\site-packages (from tensorflow) (4.11.0)
Requirement already satisfied: wrapt>=1.11.0 in c:\users\laptop land\anaconda3\lib\site-packages (from tensorflow) (1.14.1)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in c:\users\laptop land\anaconda3\lib\site-packages (from tensorflow) (1.71.0)
Requirement already satisfied: tensorboard~=2.19.0 in c:\users\laptop land\anaconda3\lib\site-packages (from tensorflow) (2.19.0)
Requirement already satisfied: keras>=3.5.0 in c:\users\laptop land\anaconda3\lib\site-packages (from tensorflow) (3.9.2)
Requirement already satisfied: numpy<2.2.0,>=1.26.0 in c:\users\laptop land\anaconda3\lib\site-packages (from tensorflow) (1.26.4)
Requirement already satisfied: h5py>=3.11.0 in c:\users\laptop land\anaconda3\lib\site-packages (from tensorflow) (3.11.0)
Requirement already satisfied: ml-dtypes<1.0.0,>=0.5.1 in c:\users\laptop land\anaconda3\lib\site-packages (from tensorflow) (0.5.1)
Requirement already satisfied: scipy>=1.6.0 in c:\users\laptop land\anaconda3\lib\site-packages (from scikit-learn) (1.13.1)
Requirement already satisfied: joblib>=1.2.0 in c:\users\laptop land\anaconda3\lib\site-packages (from scikit-learn) (1.4.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in c:\users\laptop land\anaconda3\lib\site-packages (from scikit-learn) (3.5.0)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\laptop land\anaconda3\lib\site-packages (from pandas) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in c:\users\laptop land\anaconda3\lib\site-packages (from pandas) (2024.1)
Requirement already satisfied: tzdata>=2022.7 in c:\users\laptop land\anaconda3\lib\site-packages (from pandas) (2023.3)
Requirement already satisfied: et-xmlfile in c:\users\laptop land\anaconda3\lib\site-packages (from openpyxl) (1.1.0)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\laptop land\anaconda3\lib\site-packages (from matplotlib) (1.2.0)
Requirement already satisfied: cycler>=0.10 in c:\users\laptop land\anaconda3\lib\site-packages (from matplotlib) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\laptop land\anaconda3\lib\site-packages (from matplotlib) (4.51.0)
Requirement already satisfied: kiwisolver>=1.3.1 in c:\users\laptop land\anaconda3\lib\site-packages (from matplotlib) (1.4.4)
Requirement already satisfied: pillow>=8 in c:\users\laptop land\anaconda3\lib\site-packages (from matplotlib) (10.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\laptop land\anaconda3\lib\site-packages (from matplotlib) (3.1.2)
Requirement already satisfied: wheel<1.0,>=0.23.0 in c:\users\laptop land\anaconda3\lib\site-packages (from astunparse>=1.6.0->tensorflc
Requirement already satisfied: rich in c:\users\laptop land\anaconda3\lib\site-packages (from keras>=3.5.0->tensorflow) (13.7.1)
Requirement already satisfied: namex in c:\users\laptop land\anaconda3\lib\site-packages (from keras>=3.5.0->tensorflow) (0.0.9)
Requirement already satisfied: optree in c:\users\laptop land\anaconda3\lib\site-packages (from keras>=3.5.0->tensorflow) (0.15.0)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\laptop land\anaconda3\lib\site-packages (from requests<3,>=2.21.0->t
Requirement already satisfied: idna<4,>=2.5 in c:\users\laptop land\anaconda3\lib\site-packages (from requests<3,>=2.21.0->tensorflow) (
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\laptop land\anaconda3\lib\site-packages (from requests<3,>=2.21.0->tensorf
Requirement already satisfied: certifi>=2017.4.17 in c:\users\laptop land\anaconda3\lib\site-packages (from requests<3,>=2.21.0->tensorf
Requirement already satisfied: markdown>=2.6.8 in c:\users\laptop land\anaconda3\lib\site-packages (from tensorboard~=2.19.0->tensorflow
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in c:\users\laptop land\anaconda3\lib\site-packages (from tensorboa
Requirement already satisfied: werkzeug>=1.0.1 in c:\users\laptop land\anaconda3\lib\site-packages (from tensorboard~=2.19.0->tensorflow
Requirement already satisfied: MarkupSafe>=2.1.1 in c:\users\laptop land\anaconda3\lib\site-packages (from werkzeug>=1.0.1->tensorboard~
Requirement already satisfied: markdown-it-py>=2.2.0 in c:\users\laptop land\anaconda3\lib\site-packages (from rich->keras>=3.5.0->tenso
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in c:\users\laptop land\anaconda3\lib\site-packages (from rich->keras>=3.5.0->ten
Requirement already satisfied: mdurl~=0.1 in c:\users\laptop land\anaconda3\lib\site-packages (from markdown-it-py>=2.2.0->rich->keras>=
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

import tensorflow as tf
from tensorflow.keras.models import Sequential, Model
from tensorflow.keras.layers import (
    Conv1D, MaxPooling1D, Flatten, Dense, Input, Dropout,
    LayerNormalization, MultiHeadAttention, GlobalAveragePooling1D
)
from tensorflow.keras.optimizers import Adam
```

```
df_train = pd.read_excel('Patient_Training_Dataset.xlsx')
df_test = pd.read_csv('data_device_4_variables_3O2Y83P1SO.csv')

# Auto-label test data
def classify(row):
```

```
    if (row['temperature'] < 35.2 or row['temperature'] > 40.1 or
        row['heart_rate'] < 60 or row['heart_rate'] > 100 or
        row['spo2'] < 93):
        return 1
    return 0

df_test['label'] = df_test.apply(classify, axis=1)



# Step 3: Updated to match train size
sequence_length = 20  # was 10
features = ['temperature', 'heart_rate', 'spo2']
X_test, y_test = [], []

for i in range(len(df_test) - sequence_length + 1):
    temp_seq = df_test.iloc[i:i+sequence_length][features].values
    ecg_seq = df_test.iloc[i:i+sequence_length]['ecg'].values
    label = df_test.iloc[i+sequence_length-1]['label']
    sequence = np.hstack((temp_seq, ecg_seq.reshape(-1, 1)))
    X_test.append(sequence)
    y_test.append(label)

X_test = np.array(X_test)
y_test = np.array(y_test)



X_train_seq, y_train = [], []

for _, row in df_train.iterrows():
    temp = row['temperature']
    hr = row['heartRate']
    spo2 = row['spo2']
    ecg_values = row[[f'ecg_{i}' for i in range(200)]].values.reshape(-1, 20)

    for ecg_chunk in ecg_values:
        chunk = np.column_stack((
            np.full((20, 1), temp),
            np.full((20, 1), hr),
            np.full((20, 1), spo2),
            ecg_chunk.reshape(-1, 1)
        ))
        X_train_seq.append(chunk)
        y_train.append(row['label'])

X_train = np.array(X_train_seq)
y_train = np.array(y_train)

# Normalize and split
X_train_small, _, y_train_small, _ = train_test_split(
    X_train, y_train, test_size=0.8, stratify=y_train, random_state=42
)

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train_small.reshape(-1, 4)).reshape(X_train_small.shape)
X_test_scaled = scaler.transform(X_test.reshape(-1, 4)).reshape(X_test.shape)



cnn_model = Sequential()
cnn_model.add(Input(shape=(X_train_scaled.shape[1], X_train_scaled.shape[2])))
cnn_model.add(Conv1D(64, 3, activation='relu'))
cnn_model.add(MaxPooling1D(2))
cnn_model.add(Flatten())
cnn_model.add(Dense(64, activation='relu'))
cnn_model.add(Dense(1, activation='sigmoid'))

cnn_model.compile(optimizer=Adam(0.001), loss='binary_crossentropy', metrics=['accuracy'])
history_cnn = cnn_model.fit(X_train_scaled, y_train_small, epochs=10, batch_size=16, validation_split=0.2)
cnn_accuracy = cnn_model.evaluate(X_test_scaled, y_test, verbose=0)[1]
```

```
Epoch 1/10
50/50 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step - accuracy: 0.4964 - loss: 0.7084 - val_accuracy: 0.5200 - val_loss: 0.6943
Epoch 2/10
50/50 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.5393 - loss: 0.6818 - val_accuracy: 0.5150 - val_loss: 0.7043
Epoch 3/10
```

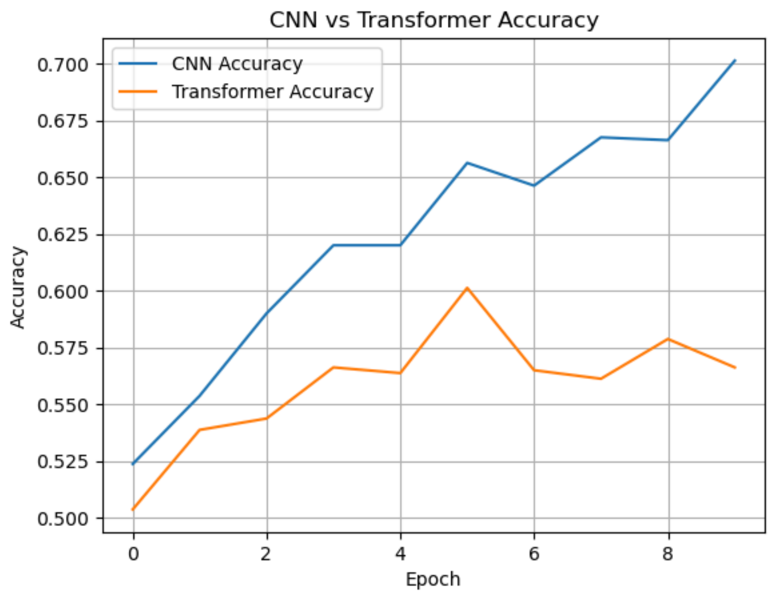
```
50/50 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.6187 - loss: 0.6521 - val_accuracy: 0.5500 - val_loss: 0.7127
Epoch 4/10
50/50 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.6228 - loss: 0.6455 - val_accuracy: 0.5600 - val_loss: 0.6884
Epoch 5/10
50/50 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.6520 - loss: 0.6232 - val_accuracy: 0.5450 - val_loss: 0.7101
Epoch 6/10
50/50 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.6464 - loss: 0.6285 - val_accuracy: 0.5550 - val_loss: 0.7149
Epoch 7/10
50/50 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.6708 - loss: 0.6052 - val_accuracy: 0.5350 - val_loss: 0.7024
Epoch 8/10
50/50 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.6664 - loss: 0.6071 - val_accuracy: 0.5550 - val_loss: 0.7052
Epoch 9/10
50/50 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.7048 - loss: 0.5739 - val_accuracy: 0.5600 - val_loss: 0.7234
Epoch 10/10
50/50 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.7214 - loss: 0.5652 - val_accuracy: 0.5200 - val_loss: 0.7130
```

```python
#  Step 6: Transformer Model with Input Projection

class TransformerBlock(tf.keras.layers.Layer):
    def __init__(self, embed_dim, num_heads):
        super().__init__()
        self.att = MultiHeadAttention(num_heads=num_heads, key_dim=embed_dim)
        self.ffn = Sequential([
            Dense(embed_dim, activation="relu"),
            Dense(embed_dim),
        ])
        self.norm1 = LayerNormalization()
        self.norm2 = LayerNormalization()

    def call(self, inputs):
        attn_output = self.att(inputs, inputs)
        out1 = self.norm1(inputs + attn_output)
        ffn_output = self.ffn(out1)
        return self.norm2(out1 + ffn_output)

#  Project input to match embed_dim
embed_dim = 64
inputs = Input(shape=(X_train_scaled.shape[1], X_train_scaled.shape[2]))  # (20, 4)
x = Dense(embed_dim)(inputs)  # Project input to (20, 64)
x = TransformerBlock(embed_dim=embed_dim, num_heads=2)(x)
x = GlobalAveragePooling1D()(x)
x = Dropout(0.1)(x)
x = Dense(64, activation='relu')(x)
x = Dropout(0.1)(x)
outputs = Dense(1, activation='sigmoid')(x)

transformer_model = Model(inputs, outputs)
transformer_model.compile(optimizer=Adam(0.001), loss='binary_crossentropy', metrics=['accuracy'])
history_transformer = transformer_model.fit(X_train_scaled, y_train_small, epochs=10, batch_size=16, validation_split=0.2)
transformer_accuracy = transformer_model.evaluate(X_test_scaled, y_test, verbose=0)[1]
```

```
Epoch 1/10
50/50 ━━━━━━━━━━━━━━━━━━━━ 3s 12ms/step - accuracy: 0.4791 - loss: 0.8147 - val_accuracy: 0.5900 - val_loss: 0.7105
Epoch 2/10
50/50 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.5371 - loss: 0.7310 - val_accuracy: 0.5400 - val_loss: 0.6830
Epoch 3/10
50/50 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.5473 - loss: 0.6779 - val_accuracy: 0.4950 - val_loss: 0.7174
Epoch 4/10
50/50 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.5531 - loss: 0.6920 - val_accuracy: 0.5300 - val_loss: 0.6959
Epoch 5/10
50/50 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.5570 - loss: 0.6829 - val_accuracy: 0.5750 - val_loss: 0.6609
Epoch 6/10
50/50 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.6345 - loss: 0.6553 - val_accuracy: 0.5750 - val_loss: 0.6784
Epoch 7/10
50/50 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - accuracy: 0.5819 - loss: 0.6687 - val_accuracy: 0.5550 - val_loss: 0.6857
Epoch 8/10
50/50 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.5432 - loss: 0.6847 - val_accuracy: 0.5100 - val_loss: 0.6900
Epoch 9/10
50/50 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - accuracy: 0.5591 - loss: 0.6727 - val_accuracy: 0.6050 - val_loss: 0.6642
Epoch 10/10
50/50 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.5526 - loss: 0.6745 - val_accuracy: 0.5650 - val_loss: 0.6751
```

```python
#  Plot accuracy comparison
plt.plot(history_cnn.history['accuracy'], label='CNN Accuracy')
plt.plot(history_transformer.history['accuracy'], label='Transformer Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.title('CNN vs Transformer Accuracy')
```

```
plt.legend()
plt.grid(True)
plt.show()

#  Final Accuracy Output
print(f"CNN Accuracy: {cnn_accuracy * 100:.2f}%")
print(f"Transformer Accuracy: {transformer_accuracy * 100:.2f}%")
```




```
CNN Accuracy: 37.50%
Transformer Accuracy: 12.50%
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
from sklearn.metrics import classification_report, confusion_matrix

# Predictions (rounded)
y_pred_cnn = (cnn_model.predict(X_test_scaled) > 0.5).astype(int)
y_pred_transformer = (transformer_model.predict(X_test_scaled) > 0.5).astype(int)

print("CNN Classification Report:\n", classification_report(y_test, y_pred_cnn))
print("Transformer Classification Report:\n", classification_report(y_test, y_pred_transformer))
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 54ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 140ms/step
CNN Classification Report:
               precision    recall  f1-score   support

           0       0.10      0.50      0.17         4
           1       0.83      0.36      0.50        28

    accuracy                           0.38        32
   macro avg       0.47      0.43      0.33        32
weighted avg       0.74      0.38      0.46        32

Transformer Classification Report:
               precision    recall  f1-score   support

           0       0.12      1.00      0.22         4
           1       0.00      0.00      0.00        28

    accuracy                           0.12        32
   macro avg       0.06      0.50      0.11        32
weighted avg       0.02      0.12      0.03        32

C:\Users\Laptop Land\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1531: UndefinedMetricWarning: Precision is ill-defin
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
C:\Users\Laptop Land\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1531: UndefinedMetricWarning: Precision is ill-defin
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
C:\Users\Laptop Land\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1531: UndefinedMetricWarning: Precision is ill-defin
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

Start coding or generate with AI.