```
!pip install -q transformers datasets torch seaborn
```

```
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 491.2/491.2 kB 12.9 MB/s eta 0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 363.4/363.4 MB 4.5 MB/s eta 0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 13.8/13.8 MB 77.5 MB/s eta 0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 24.6/24.6 MB 61.0 MB/s eta 0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 883.7/883.7 kB 52.2 MB/s eta 0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 664.8/664.8 MB 2.6 MB/s eta 0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 211.5/211.5 MB 5.6 MB/s eta 0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 56.3/56.3 MB 12.0 MB/s eta 0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 127.9/127.9 MB 7.4 MB/s eta 0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 207.5/207.5 MB 5.5 MB/s eta 0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 21.1/21.1 MB 69.4 MB/s eta 0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 116.3/116.3 kB 8.3 MB/s eta 0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 143.5/143.5 kB 13.5 MB/s eta 0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 194.8/194.8 kB 15.5 MB/s eta 0:00:00
```

```
import pandas as pd
import numpy as np
import torch
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score, precision_score, recall_score, f1_score
import seaborn as sns
import matplotlib.pyplot as plt
from transformers import AutoTokenizer, AutoModel
from tqdm import tqdm
```

```
raw_df = pd.read_csv("/content/merged.csv")
print("Shape of data:", raw_df.shape)
print("Columns:", raw_df.columns.tolist())
print("Label distribution:\n", raw_df["label"].value_counts())
```

```
Shape of data: (1178, 59)
Columns: ['Unnamed: 0', 'net_acc_mean', 'net_acc_std', 'net_acc_min', 'net_acc_max', 'ACC_x_mean', 'ACC_x_std', 'ACC_x_m
Label distribution:
 label
1    628
2    355
0    195
Name: count, dtype: int64
```

Basic Preprocessing

```
possible_drop_cols = ['subject', 'age', 'gender', 'height', 'weight', 'coffee_today_YES', 'smoker_YES']
drop_cols = [col for col in possible_drop_cols if col in raw_df.columns]
df = raw_df.drop(columns=drop_cols)
df = df.interpolate().dropna()
```

```
if df['label'].dtype == object:
    le = LabelEncoder()
    df['label'] = le.fit_transform(df['label'])
```

```
# Step 5: Normalize numeric features (exclude timestamp & label)
features = df.drop(columns=['label'])
scaler = StandardScaler()
X_scaled = scaler.fit_transform(features)
df_scaled = pd.DataFrame(X_scaled, columns=features.columns)
df_scaled['label'] = df['label'].values
```

```
# Step 6: Segment Data into Sliding Windows
WINDOW_SIZE = 60
STEP_SIZE = 30
segments = []
labels_windows = []
```

```
for start in range(0, len(df_scaled) - WINDOW_SIZE, STEP_SIZE):
    end = start + WINDOW_SIZE
    window = df_scaled.iloc[start:end]
    if len(window) == WINDOW_SIZE:
        segment_text = ' '.join(window.drop(columns=['label']).astype(str).values.flatten())
        segments.append(segment_text)
        label = window['label'].mode()[0]
        labels_windows.append(label)
```

```
# Step 7: Load Transformer Model (mock PhysioBERT with 'bert-base-uncased')
tokenizer = AutoTokenizer.from_pretrained('bert-base-uncased')
model = AutoModel.from_pretrained('bert-base-uncased')
model.eval()
```

```
/usr/local/lib/python3.11/dist-packages/huggingface_hub/utils/_auth.py:94: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/settings/tokens),
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models or datasets.
  warnings.warn(
tokenizer_config.json: 100%   48.0/48.0 [00:00<00:00, 1.66kB/s]
config.json: 100%   570/570 [00:00<00:00, 32.6kB/s]
vocab.txt: 100%   232k/232k [00:00<00:00, 3.51MB/s]
tokenizer.json: 100%   466k/466k [00:00<00:00, 9.68MB/s]
model.safetensors: 100%   440M/440M [00:06<00:00, 100MB/s]
BertModel(
  (embeddings): BertEmbeddings(
    (word_embeddings): Embedding(30522, 768, padding_idx=0)
    (position_embeddings): Embedding(512, 768)
    (token_type_embeddings): Embedding(2, 768)
    (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
    (dropout): Dropout(p=0.1, inplace=False)
  )
  (encoder): BertEncoder(
    (layer): ModuleList(
      (0-11): 12 x BertLayer(
        (attention): BertAttention(
          (self): BertSdpaSelfAttention(
            (query): Linear(in_features=768, out_features=768, bias=True)
            (key): Linear(in_features=768, out_features=768, bias=True)
            (value): Linear(in_features=768, out_features=768, bias=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
          (output): BertSelfOutput(
            (dense): Linear(in_features=768, out_features=768, bias=True)
            (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
        )
        (intermediate): BertIntermediate(
          (dense): Linear(in_features=768, out_features=3072, bias=True)
          (intermediate_act_fn): GELUActivation()
        )
        (output): BertOutput(
          (dense): Linear(in_features=3072, out_features=768, bias=True)
          (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
          (dropout): Dropout(p=0.1, inplace=False)
        )
      )
    )
  )
  (pooler): BertPooler(
    (dense): Linear(in_features=768, out_features=768, bias=True)
    (activation): Tanh()
  )
)
```

```
# Step 8: Tokenize & Embed Sequences
embedding_list = []
with torch.no_grad():
    for seg in tqdm(segments, desc="Extracting Embeddings"):
        inputs = tokenizer(seg, return_tensors='pt', truncation=True, padding='max_length', max_length=512)
        outputs = model(**inputs)
        embedding = outputs.last_hidden_state.mean(dim=1).squeeze().numpy()
        embedding_list.append(embedding)
```

```
Extracting Embeddings: 100%|██████████| 38/38 [00:44<00:00,  1.16s/it]
```

```
X_embeddings = np.array(embedding_list)
y = np.array(labels_windows)
```

```
# Step 9: Train-Test Split
X_train, X_test, y_train, y_test = train_test_split(X_embeddings, y, test_size=0.2, random_state=42, stratify=y)
```

```
# Step 10: Train Classifier
```

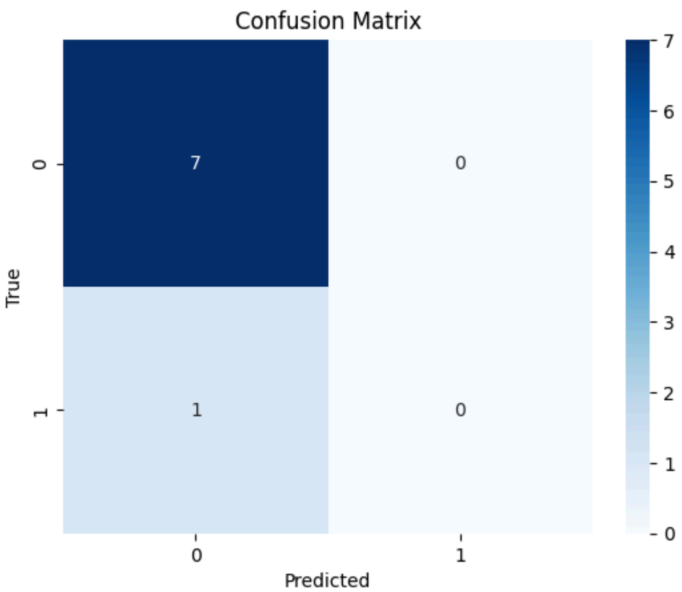
```
# Step 10: Train classifier
from sklearn.ensemble import RandomForestClassifier
clf = RandomForestClassifier(n_estimators=100, random_state=42)
clf.fit(X_train, y_train)
```

```
▾ RandomForestClassifier
RandomForestClassifier(random_state=42)
```

```
# Step 11: Evaluate
y_pred = clf.predict(X_test)
print("\nClassification Report:\n", classification_report(y_test, y_pred))
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Precision:", precision_score(y_test, y_pred, average='weighted', zero_division=0))
print("Recall:", recall_score(y_test, y_pred, average='weighted'))
print("F1 Score:", f1_score(y_test, y_pred, average='weighted'))
```

```
Classification Report:
               precision    recall  f1-score   support

           1       0.88      1.00      0.93         7
           2       0.00      0.00      0.00         1

    accuracy                           0.88         8
   macro avg       0.44      0.50      0.47         8
weighted avg       0.77      0.88      0.82         8

Accuracy: 0.875
Precision: 0.765625
Recall: 0.875
F1 Score: 0.8166666666666667
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is il
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is il
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is il
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

```
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues")
plt.xlabel("Predicted")
plt.ylabel("True")
plt.title("Confusion Matrix")
plt.show()
```



```
# Step 12: Save Segment Strings for Visualization
segments_df = pd.DataFrame({
    "window_text": segments,
    "label": labels_windows
})
segments_df.to_csv("physiobert_input_strings.csv", index=False)
print("Saved segment strings to physiobert_input_strings.csv")

# Optionally, download the file in Colab
from google.colab import files
files.download("physiobert_input_strings.csv")
```

```
Saved segment strings to physiobert_input_strings.csv
```

Optimize Your Model for Better Performance

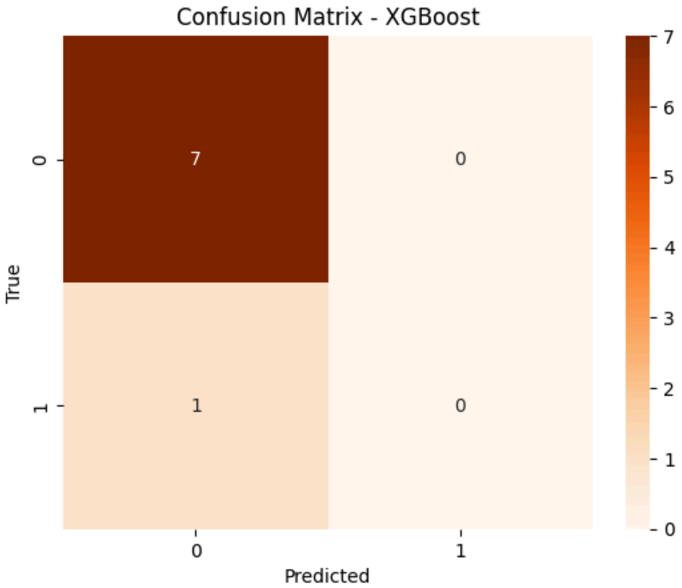
Instead of RandomForestClassifier, you can use a gradient boosting model like XGBoost or LightGBM, which often performs better.

```
import xgboost as xgb

# Copy original train-test splits into new variables
X_train_xgb = X_train.copy()
X_test_xgb = X_test.copy()
y_train_xgb = y_train.copy()
y_test_xgb = y_test.copy()

# Map labels to start from 0 without affecting original
label_map = {label: idx for idx, label in enumerate(sorted(np.unique(y_train_xgb)))}
y_train_xgb = np.array([label_map[val] for val in y_train_xgb])
y_test_xgb = np.array([label_map[val] for val in y_test_xgb])

# Initialize and train XGBoost
xgb_clf = xgb.XGBClassifier(n_estimators=200, learning_rate=0.05, max_depth=6, random_state=42)
xgb_clf.fit(X_train_xgb, y_train_xgb)
```

```
XGBClassifier
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, device=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_types=None,
              gamma=None, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=0.05, max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=6, max_leaves=None,
              min_child_weight=None, missing=nan, monotone_constraints=None,
              multi_strategy=None, n_estimators=200, n_jobs=None,
              num_parallel_tree=None, random_state=42, ...)
```

```
y_pred_xgb = xgb_clf.predict(X_test_xgb)

print("\nClassification Report (XGBoost):\n", classification_report(y_test_xgb, y_pred_xgb))
print("Accuracy:", accuracy_score(y_test_xgb, y_pred_xgb))
print("Precision:", precision_score(y_test_xgb, y_pred_xgb, average='weighted', zero_division=0))
print("Recall:", recall_score(y_test_xgb, y_pred_xgb, average='weighted'))
print("F1 Score:", f1_score(y_test_xgb, y_pred_xgb, average='weighted'))

# Confusion Matrix
cm_xgb = confusion_matrix(y_test_xgb, y_pred_xgb)
sns.heatmap(cm_xgb, annot=True, fmt="d", cmap="Oranges")
plt.xlabel("Predicted")
plt.ylabel("True")
plt.title("Confusion Matrix - XGBoost")
plt.show()
```

```
Classification Report (XGBoost):
              precision    recall  f1-score   support

           0       0.88      1.00      0.93         7
           1       0.00      0.00      0.00         1

    accuracy                           0.88         8
   macro avg       0.44      0.50      0.47         8
weighted avg       0.77      0.88      0.82         8

Accuracy: 0.875
Precision: 0.765625
Recall: 0.875
F1 Score: 0.8166666666666667
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is il
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is il
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is il
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```



Start coding or generate with AI.

Apply SMOTE to Balance Classes Adjust k_neighbors Based on Minority Class Size and --- Apply SMOTE with Dynamic k_neighbors

```
from collections import Counter
from imblearn.over_sampling import SMOTE

# Check class distribution
class_counts = Counter(y_train_xgb)
print("Class distribution before SMOTE:", class_counts)

# Get smallest class size
min_class_samples = min(class_counts.values())

# Set k_neighbors < min_class_samples
k_val = min(min_class_samples - 1, 5)

# Apply SMOTE with adjusted k
sm = SMOTE(random_state=42, k_neighbors=k_val)
X_train_xgb_sm, y_train_xgb_sm = sm.fit_resample(X_train_xgb, y_train_xgb)

print("Class distribution after SMOTE:", Counter(y_train_xgb_sm))
```

```
Class distribution before SMOTE: Counter({0: 26, 1: 4})
Class distribution after SMOTE: Counter({0: 26, 1: 26})
```

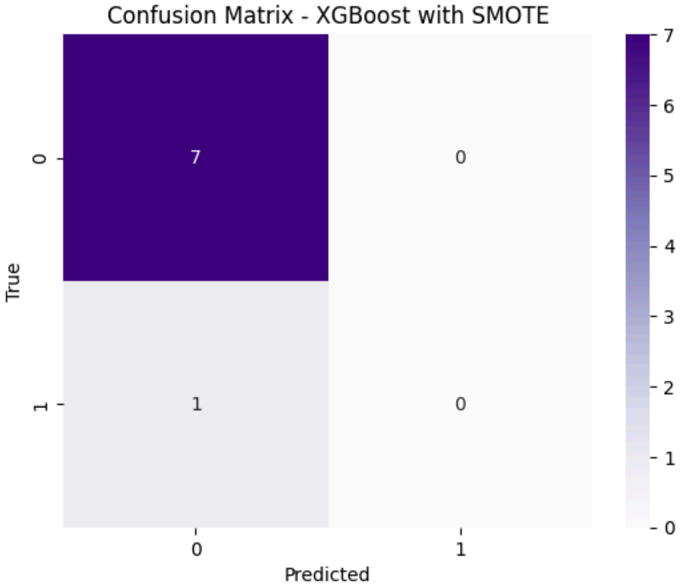
Train XGBoost on SMOTE Data

```
xgb_clf_sm = xgb.XGBClassifier(n_estimators=200, learning_rate=0.05, max_depth=6, random_state=42)
xgb_clf_sm.fit(X_train_xgb_sm, y_train_xgb_sm)
```

```
XGBClassifier
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, device=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_types=None,
              gamma=None, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=0.05, max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=6, max_leaves=None,
              min_child_weight=None, missing=nan, monotone_constraints=None,
              multi_strategy=None, n_estimators=200, n_jobs=None,
              num_parallel_tree=None, random_state=42, ...)
```

```
y_pred_xgb_sm = xgb_clf_sm.predict(X_test_xgb)

print("\nClassification Report (XGBoost + SMOTE):\n", classification_report(y_test_xgb, y_pred_xgb_sm))
print("Accuracy:", accuracy_score(y_test_xgb, y_pred_xgb_sm))
print("Precision:", precision_score(y_test_xgb, y_pred_xgb_sm, average='weighted', zero_division=0))
print("Recall:", recall_score(y_test_xgb, y_pred_xgb_sm, average='weighted'))
print("F1 Score:", f1_score(y_test_xgb, y_pred_xgb_sm, average='weighted'))

# Confusion Matrix
cm_xgb_sm = confusion_matrix(y_test_xgb, y_pred_xgb_sm)
sns.heatmap(cm_xgb_sm, annot=True, fmt="d", cmap="Purples")
plt.xlabel("Predicted")
plt.ylabel("True")
plt.title("Confusion Matrix - XGBoost with SMOTE")
plt.show()
```

```
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is il
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is il
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is il
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))

Classification Report (XGBoost + SMOTE):
               precision    recall  f1-score   support

           0       0.88      1.00      0.93         7
           1       0.00      0.00      0.00         1

    accuracy                           0.88         8
   macro avg       0.44      0.50      0.47         8
weighted avg       0.77      0.88      0.82         8

Accuracy: 0.875
Precision: 0.765625
Recall: 0.875
F1 Score: 0.8166666666666667
```



:Branch off and Try TimeGPT Embeddings

Let’s say you keep your BERT-based X_embeddings as X_bert_embed.

You can now add a TimeGPT-based embedding path:

Optionally: Use an existing time-series transformer from tsai:

```
!pip install tsai
```

```
Requirement already satisfied: nvidia-cuda-cupti-cu12==12.4.127 in /usr/local/lib/python3.11/dist-packages (from torch<2
Requirement already satisfied: nvidia-cudnn-cu12==9.1.0.70 in /usr/local/lib/python3.11/dist-packages (from torch<2.6,>=
Requirement already satisfied: nvidia-cublas-cu12==12.4.5.8 in /usr/local/lib/python3.11/dist-packages (from torch<2.6,>
Requirement already satisfied: nvidia-cufft-cu12==11.2.1.3 in /usr/local/lib/python3.11/dist-packages (from torch<2.6,>=
Requirement already satisfied: nvidia-curand-cu12==10.3.5.147 in /usr/local/lib/python3.11/dist-packages (from torch<2.6
Requirement already satisfied: nvidia-cusolver-cu12==11.6.1.9 in /usr/local/lib/python3.11/dist-packages (from torch<2.6
Requirement already satisfied: nvidia-cusparse-cu12==12.3.1.170 in /usr/local/lib/python3.11/dist-packages (from torch<2
Requirement already satisfied: nvidia-nccl-cu12==2.21.5 in /usr/local/lib/python3.11/dist-packages (from torch<2.6,>=1.1
Requirement already satisfied: nvidia-nvtx-cu12==12.4.127 in /usr/local/lib/python3.11/dist-packages (from torch<2.6,>=1
Requirement already satisfied: nvidia-nvjitlink-cu12==12.4.127 in /usr/local/lib/python3.11/dist-packages (from torch<2.
Requirement already satisfied: triton==3.1.0 in /usr/local/lib/python3.11/dist-packages (from torch<2.6,>=1.10->tsai) (3
Requirement already satisfied: sympy==1.13.1 in /usr/local/lib/python3.11/dist-packages (from torch<2.6,>=1.10->tsai) (1
Requirement already satisfied: mpmath<1.4,>=1.1.0 in /usr/local/lib/python3.11/dist-packages (from sympy==1.13.1->torch<
Requirement already satisfied: llvmlite<0.44,>=0.43.0dev0 in /usr/local/lib/python3.11/dist-packages (from numba>=0.55.2
Requirement already satisfied: spacy-legacy<3.1.0,>=3.0.11 in /usr/local/lib/python3.11/dist-packages (from spacy<4->fas
Requirement already satisfied: spacy-loggers<2.0.0,>=1.0.0 in /usr/local/lib/python3.11/dist-packages (from spacy<4->fas
Requirement already satisfied: murmurhash<1.1.0,>=0.28.0 in /usr/local/lib/python3.11/dist-packages (from spacy<4->fasta
Requirement already satisfied: cymem<2.1.0,>=2.0.2 in /usr/local/lib/python3.11/dist-packages (from spacy<4->fastai>=2.7
Requirement already satisfied: preshed<3.1.0,>=3.0.2 in /usr/local/lib/python3.11/dist-packages (from spacy<4->fastai>=2
Requirement already satisfied: thinc<8.3.0,>=8.2.2 in /usr/local/lib/python3.11/dist-packages (from spacy<4->fastai>=2.7
Requirement already satisfied: wasabi<1.2.0,>=0.9.1 in /usr/local/lib/python3.11/dist-packages (from spacy<4->fastai>=2.
Requirement already satisfied: srsly<3.0.0,>=2.4.3 in /usr/local/lib/python3.11/dist-packages (from spacy<4->fastai>=2.7
Requirement already satisfied: catalogue<2.1.0,>=2.0.6 in /usr/local/lib/python3.11/dist-packages (from spacy<4->fastai>
Requirement already satisfied: weasel<0.5.0,>=0.1.0 in /usr/local/lib/python3.11/dist-packages (from spacy<4->fastai>=2.
Requirement already satisfied: typer<1.0.0,>=0.3.0 in /usr/local/lib/python3.11/dist-packages (from spacy<4->fastai>=2.7
Requirement already satisfied: tqdm<5.0.0,>=4.38.0 in /usr/local/lib/python3.11/dist-packages (from spacy<4->fastai>=2.7
Requirement already satisfied: pydantic!=1.8,!=1.8.1,<3.0.0,>=1.7.4 in /usr/local/lib/python3.11/dist-packages (from spa
Requirement already satisfied: setuptools in /usr/local/lib/python3.11/dist-packages (from spacy<4->fastai>=2.7.18->tsai
Requirement already satisfied: langcodes<4.0.0,>=3.2.0 in /usr/local/lib/python3.11/dist-packages (from spacy<4->fastai>
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages (from requests->fasta
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-packages (from requests->fastai>=2.7.18->t
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-packages (from requests->fastai>=2.7
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-packages (from requests->fastai>=2.7
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.11/dist-packages (from jinja2->torch<2.6,>=1.10
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib->fastai>=2.7
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.11/dist-packages (from matplotlib->fastai>=2.7.18-
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.11/dist-packages (from matplotlib->fastai>=2.
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib->fastai>=2.
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib->fastai>=2.7
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.11/dist-packages (from matplotlib->fastai>
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-packages (from pandas->fastai>=2.7.18->tsa
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packages (from pandas->fastai>=2.7.18->t
Requirement already satisfied: language-data>=1.2 in /usr/local/lib/python3.11/dist-packages (from langcodes<4.0.0,>=3.2
Requirement already satisfied: annotated-types>=0.6.0 in /usr/local/lib/python3.11/dist-packages (from pydantic!=1.8,!=1
Requirement already satisfied: pydantic-core==2.27.2 in /usr/local/lib/python3.11/dist-packages (from pydantic!=1.8,!=1.
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (from python-dateutil>=2.7->matplotli
Requirement already satisfied: blis<0.8.0,>=0.7.8 in /usr/local/lib/python3.11/dist-packages (from thinc<8.3.0,>=8.2.2->
Requirement already satisfied: confection<1.0.0,>=0.0.1 in /usr/local/lib/python3.11/dist-packages (from thinc<8.3.0,>=8
Requirement already satisfied: click>=8.0.0 in /usr/local/lib/python3.11/dist-packages (from typer<1.0.0,>=0.3.0->spacy<
Requirement already satisfied: shellingham>=1.3.0 in /usr/local/lib/python3.11/dist-packages (from typer<1.0.0,>=0.3.0->
Requirement already satisfied: rich>=10.11.0 in /usr/local/lib/python3.11/dist-packages (from typer<1.0.0,>=0.3.0->spacy
Requirement already satisfied: cloudpathlib<1.0.0,>=0.7.0 in /usr/local/lib/python3.11/dist-packages (from weasel<0.5.0,
Requirement already satisfied: smart-open<8.0.0,>=5.2.1 in /usr/local/lib/python3.11/dist-packages (from weasel<0.5.0,>=
Requirement already satisfied: marisa-trie>=1.1.0 in /usr/local/lib/python3.11/dist-packages (from language-data>=1.2->l
Requirement already satisfied: markdown-it-py>=2.2.0 in /usr/local/lib/python3.11/dist-packages (from rich>=10.11.0->typ
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in /usr/local/lib/python3.11/dist-packages (from rich>=10.11.0->t
Requirement already satisfied: wrapt in /usr/local/lib/python3.11/dist-packages (from smart-open<8.0.0,>=5.2.1->weasel<0
Requirement already satisfied: mdurl~=0.1 in /usr/local/lib/python3.11/dist-packages (from markdown-it-py>=2.2.0->rich>=
```

Recreate df_scaled from Uploaded Data

```
import pandas as pd
from sklearn.preprocessing import StandardScaler, LabelEncoder

# Load your dataset (make sure merged.csv is uploaded again)
raw_df = pd.read_csv("merged.csv")

# Drop irrelevant columns if they exist
possible_drop_cols = ['subject', 'age', 'gender', 'height', 'weight', 'coffee_today_YES', 'smoker_YES']
drop_cols = [col for col in possible_drop_cols if col in raw_df.columns]
df_temp = raw_df.drop(columns=drop_cols)

# Fill missing values
df_temp = df_temp.interpolate().dropna()

# Encode label if it's not numeric
if df_temp['label'].dtype == object:
```

```
    le = LabelEncoder()
    df_temp['label'] = le.fit_transform(df_temp['label'])

# Scale all features except label
features = df_temp.drop(columns=['label'])
scaler = StandardScaler()
X_scaled = scaler.fit_transform(features)

# Final scaled dataframe
df_scaled = pd.DataFrame(X_scaled, columns=features.columns)
df_scaled['label'] = df_temp['label'].values



X_array = df_scaled.drop(columns=['label']).values
y_array = df_scaled['label'].values



print(df_scaled.head())
print(df_scaled['label'].value_counts())
```

```
   Unnamed: 0  net_acc_mean  net_acc_std  net_acc_min  net_acc_max  \
0   -1.730581      0.037111     1.690034    -1.299795     2.343844
1   -1.727640     -0.797006     3.284458    -1.299795     1.460136
2   -1.724700     -0.917309     1.985042    -1.093426     0.499585
3   -1.721759      0.510472    -0.135361    -0.267948    -0.268857
4   -1.718818     -0.074758     0.187103    -0.267948    -0.384123

   ACC_x_mean  ACC_x_std  ACC_x_min  ACC_x_max  ACC_y_mean  ...  TEMP_min  \
0    0.626491   1.702861  -0.591976   1.874421    0.626491  ...  1.919647
1    0.203971   3.985211  -0.997753   1.383929    0.203971  ...  1.892392
2    0.325166   1.676815   0.199289   0.850785    0.325166  ...  1.831070
3    0.798112  -0.175366   0.523910   0.424270    0.798112  ...  1.831070
4    0.604257   0.106307   0.523910   0.360293    0.604257  ...  1.865138

   TEMP_max  BVP_peak_freq  TEMP_slope  gender_ female  gender_ male  \
0  1.897464       0.181195   -0.254462       -0.504506      0.504506
1  1.917844      -0.822014   -1.275585       -0.504506      0.504506
2  1.836324      -1.269800   -1.157422       -0.504506      0.504506
3  1.822737       0.294767    0.148210       -0.504506      0.504506
4  1.863497       0.518123    0.752063       -0.504506      0.504506

   sport_today_YES  smoker_NO  feel_ill_today_YES  label
0        -0.392136   0.266288           -0.266288      1
1        -0.392136   0.266288           -0.266288      1
2        -0.392136   0.266288           -0.266288      1
3        -0.392136   0.266288           -0.266288      1
4        -0.392136   0.266288           -0.266288      1

[5 rows x 53 columns]
label
1    628
2    355
0    195
Name: count, dtype: int64
```

Use Time Series Transformer via tsai