```
pip install pandas numpy scikit-learn matplotlib seaborn
```

```
Requirement already satisfied: pandas in /usr/local/lib/python3.11/dist-packages (2.2.2)
Requirement already satisfied: numpy in /usr/local/lib/python3.11/dist-packages (2.0.2)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.11/dist-packages (1.6.1)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.11/dist-packages (3.10.0)
Requirement already satisfied: seaborn in /usr/local/lib/python3.11/dist-packages (0.13.2)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.11/dist-packages (from pandas) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-packages (from pandas) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packages (from pandas) (2025.2)
Requirement already satisfied: scipy>=1.6.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn) (1.16.0)
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn) (1.5.1)
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn) (3.6.0)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (1.3.2)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (4.59.0)
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (1.4.8)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (25.0)
Requirement already satisfied: pillow>=8 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (11.3.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (3.2.3)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (from python-dateutil>=2.8.2->pandas) (1.17.0)
```

```
pip install tensorflow
```

```
Requirement already satisfied: tensorflow in /usr/local/lib/python3.11/dist-packages (2.18.0)
Requirement already satisfied: absl-py>=1.0.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (1.4.0)
Requirement already satisfied: astunparse>=1.6.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (1.6.3)
Requirement already satisfied: flatbuffers>=24.3.25 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (25.2.10)
Requirement already satisfied: gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (0.6
Requirement already satisfied: google-pasta>=0.1.1 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (0.2.0)
Requirement already satisfied: libclang>=13.0.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (18.1.1)
Requirement already satisfied: opt-einsum>=2.3.2 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (3.4.0)
Requirement already satisfied: packaging in /usr/local/lib/python3.11/dist-packages (from tensorflow) (25.0)
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 /usr/local/lib/py
Requirement already satisfied: requests<3,>=2.21.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (2.32.3)
Requirement already satisfied: setuptools in /usr/local/lib/python3.11/dist-packages (from tensorflow) (75.2.0)
Requirement already satisfied: six>=1.12.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (1.17.0)
Requirement already satisfied: termcolor>=1.1.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (3.1.0)
Requirement already satisfied: typing-extensions>=3.6.6 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (4.14.1)
Requirement already satisfied: wrapt>=1.11.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (1.17.2)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (1.74.0)
Requirement already satisfied: tensorboard<2.19,>=2.18 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (2.18.0)
Requirement already satisfied: keras>=3.5.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (3.8.0)
Requirement already satisfied: numpy<2.1.0,>=1.26.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (2.0.2)
Requirement already satisfied: h5py>=3.11.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (3.14.0)
Requirement already satisfied: ml-dtypes<0.5.0,>=0.4.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (0.4.1)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (0
Requirement already satisfied: wheel<1.0,>=0.23.0 in /usr/local/lib/python3.11/dist-packages (from astunparse>=1.6.0->tensorflow) (6
Requirement already satisfied: rich in /usr/local/lib/python3.11/dist-packages (from keras>=3.5.0->tensorflow) (13.9.4)
Requirement already satisfied: namex in /usr/local/lib/python3.11/dist-packages (from keras>=3.5.0->tensorflow) (0.1.0)
Requirement already satisfied: optree in /usr/local/lib/python3.11/dist-packages (from keras>=3.5.0->tensorflow) (0.17.0)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages (from requests<3,>=2.21.0->tensor
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-packages (from requests<3,>=2.21.0->tensorflow) (3.10
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-packages (from requests<3,>=2.21.0->tensorflow)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-packages (from requests<3,>=2.21.0->tensorflow)
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.11/dist-packages (from tensorboard<2.19,>=2.18->tensorflow
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in /usr/local/lib/python3.11/dist-packages (from tensorboard<2
Requirement already satisfied: werkzeug>=1.0.1 in /usr/local/lib/python3.11/dist-packages (from tensorboard<2.19,>=2.18->tensorflow
Requirement already satisfied: MarkupSafe>=2.1.1 in /usr/local/lib/python3.11/dist-packages (from werkzeug>=1.0.1->tensorboard<2.19,
Requirement already satisfied: markdown-it-py>=2.2.0 in /usr/local/lib/python3.11/dist-packages (from rich->keras>=3.5.0->tensorflow
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in /usr/local/lib/python3.11/dist-packages (from rich->keras>=3.5.0->tensorfl
Requirement already satisfied: mdurl~=0.1 in /usr/local/lib/python3.11/dist-packages (from markdown-it-py>=2.2.0->rich->keras>=3.5.6
```

```
from google.colab import files
uploaded = files.upload()  # Choose ecg_sleep_apnea_dataset.csv when prompted
```

Choose files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
import pandas as pd
df = pd.read_csv('ecg_sleep_apnea_dataset.csv')
df.head()
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
/tmp/ipython-input-736266576.py in <cell line: 0>()
      1 import pandas as pd
----> 2 df = pd.read_csv('ecg_sleep_apnea_dataset.csv')
      3 df.head()

                              4 frames
/usr/local/lib/python3.11/dist-packages/pandas/io/common.py in get_handle(path_or_buf, mode, encoding, compression, memory_map,
is_text, errors, storage_options)
    871         if ioargs.encoding and "b" not in ioargs.mode:
    872             # Encoding
--> 873             handle = open(
    874                 handle,
    875                 ioargs.mode,

FileNotFoundError: [Errno 2] No such file or directory: 'ecg_sleep_apnea_dataset.csv'
```

```
from google.colab import files
uploaded = files.upload()  # Upload your ecg_sleep_apnea_dataset.csv.zip
```

Choose files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
import zipfile
import os

# Extract zip file
with zipfile.ZipFile("ecg_sleep_apnea_dataset.csv.zip", 'r') as zip_ref:
    zip_ref.extractall(".")

# Now check what files were extracted
print(os.listdir("."))
```

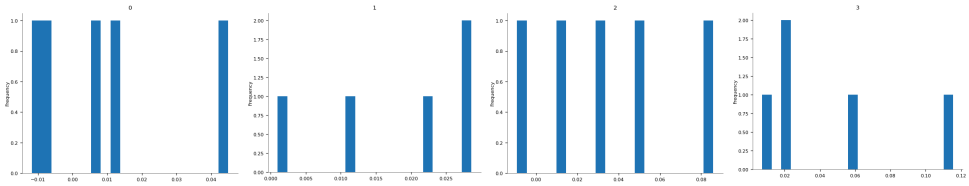
```
['.config', 'ecg_sleep_apnea_dataset.csv', 'ecg_sleep_apnea_dataset.csv.zip', 'ecg_sleep_apnea_dataset.csv (1).zip', 'sample_data']
```

```
import pandas as pd

df = pd.read_csv("ecg_sleep_apnea_dataset.csv")
df.head()
```

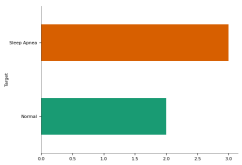
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | ... | 2491 | 2492 | 24 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0.012976 | 0.022008 | 0.049401 | 0.007309 | 0.080700 | 0.084465 | 0.053040 | 0.110527 | 0.147975 | 0.158048 | ... | -0.086520 | -0.076036 | -0.0730 |
| **1** | 0.007665 | 0.011024 | 0.014001 | 0.115614 | 0.045236 | 0.053854 | 0.144119 | 0.093378 | 0.178889 | 0.133928 | ... | -0.590053 | -0.585818 | -0.5045 |
| **2** | 0.044957 | 0.028612 | 0.085881 | 0.018910 | 0.078694 | 0.103297 | 0.046348 | 0.148435 | 0.148251 | 0.140560 | ... | -0.663029 | -0.596072 | -0.5015 |
| **3** | -0.011676 | 0.027831 | 0.029627 | 0.021658 | 0.068194 | 0.075705 | 0.095863 | 0.123471 | 0.155526 | 0.150709 | ... | -0.085069 | -0.117560 | -0.1082 |
| **4** | -0.008188 | 0.001010 | -0.009165 | 0.061274 | 0.087704 | 0.055419 | 0.120823 | 0.107706 | 0.133526 | 0.166235 | ... | -0.114268 | -0.061457 | -0.0972 |

5 rows × 2501 columns

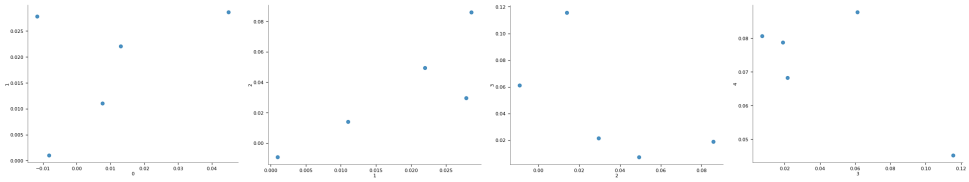
**Distributions**



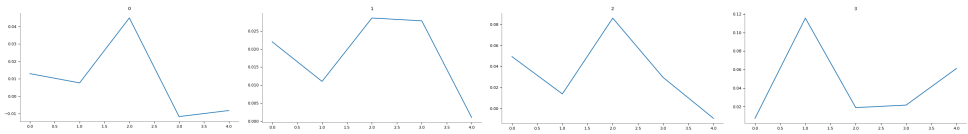
**Categorical distributions**



**2-d distributions**

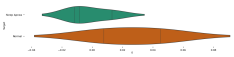


**Values**

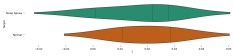


**Faceted distributions**

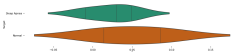
```
<string>:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `
```



```
<string>:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `
```



```
<string>:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `
```



```
<string>:5: FutureWarning:
```

```
print(df.info())
print(df.isnull().sum())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2660 entries, 0 to 2659
Columns: 2501 entries, 0 to Target
dtypes: float64(2500), object(1)
memory usage: 50.8+ MB
None
0         0
1         0
2         0
3         0
4         0
         ..
2496      0
2497      0
2498      0
```

```
2499      0
Target    0
Length: 2501, dtype: int64
```

```
df = df.dropna()  # or use df.fillna(method='ffill')
```

```
X = df.drop(['Target'], axis=1)
y = df['Target']
```

```
# Features and labels
X = df.drop(['Target'], axis=1)
y = df['Target']

# Convert target to integer by mapping string labels to numerical values
y = y.map({'Normal': 0, 'Sleep Apnea': 1})
```

```
print(df.columns.tolist())
```

```
['0', '1', '2', '3', '4', '5', '6', '7', '8', '9', '10', '11', '12', '13', '14', '15', '16', '17', '18', '19', '20', '21', '22', '23
```

```
# Apply feature engineering to all numerical columns
# Drop the 'Target' column from df before applying rolling window functions
df_processed = df.drop(['Target'], axis=1)

for col in df_processed.columns:
    df[f'{col}_mean'] = df_processed[col].rolling(window=100).mean()
    df[f'{col}_std'] = df_processed[col].rolling(window=100).std()
    df[f'{col}_rms'] = (df_processed[col]**2).rolling(window=100).mean()**0.5

# Drop original columns and rows with NaN values created by rolling window
# Keep the original 'Target' column from the original df
original_target = df['Target']
df = df.drop(df_processed.columns, axis=1)

# Combine the processed features with the original target column
df = pd.concat([df, original_target], axis=1)

df = df.dropna()

# Update X and y after feature engineering and dropping NaNs
X = df.drop(['Target'], axis=1)
y = df['Target']
```

```
Streaming output truncated to the last 5000 lines.
/tmp/ipython-input-19442847.py:8: PerformanceWarning: DataFrame is highly fragmented.  This is usually the result of calling `fram
  df[f'{col}_rms'] = (df_processed[col]**2).rolling(window=100).mean()**0.5
/tmp/ipython-input-19442847.py:6: PerformanceWarning: DataFrame is highly fragmented.  This is usually the result of calling `fram
  df[f'{col}_mean'] = df_processed[col].rolling(window=100).mean()
/tmp/ipython-input-19442847.py:7: PerformanceWarning: DataFrame is highly fragmented.  This is usually the result of calling `fram
  df[f'{col}_std'] = df_processed[col].rolling(window=100).std()
/tmp/ipython-input-19442847.py:8: PerformanceWarning: DataFrame is highly fragmented.  This is usually the result of calling `fram
  df[f'{col}_rms'] = (df_processed[col]**2).rolling(window=100).mean()**0.5
/tmp/ipython-input-19442847.py:6: PerformanceWarning: DataFrame is highly fragmented.  This is usually the result of calling `fram
  df[f'{col}_mean'] = df_processed[col].rolling(window=100).mean()
/tmp/ipython-input-19442847.py:7: PerformanceWarning: DataFrame is highly fragmented.  This is usually the result of calling `fram
  df[f'{col}_std'] = df_processed[col].rolling(window=100).std()
/tmp/ipython-input-19442847.py:8: PerformanceWarning: DataFrame is highly fragmented.  This is usually the result of calling `fram
  df[f'{col}_rms'] = (df_processed[col]**2).rolling(window=100).mean()**0.5
/tmp/ipython-input-19442847.py:6: PerformanceWarning: DataFrame is highly fragmented.  This is usually the result of calling `fram
  df[f'{col}_mean'] = df_processed[col].rolling(window=100).mean()
/tmp/ipython-input-19442847.py:7: PerformanceWarning: DataFrame is highly fragmented.  This is usually the result of calling `fram
  df[f'{col}_std'] = df_processed[col].rolling(window=100).std()
/tmp/ipython-input-19442847.py:8: PerformanceWarning: DataFrame is highly fragmented.  This is usually the result of calling `fram
  df[f'{col}_rms'] = (df_processed[col]**2).rolling(window=100).mean()**0.5
/tmp/ipython-input-19442847.py:6: PerformanceWarning: DataFrame is highly fragmented.  This is usually the result of calling `fram
  df[f'{col}_mean'] = df_processed[col].rolling(window=100).mean()
/tmp/ipython-input-19442847.py:7: PerformanceWarning: DataFrame is highly fragmented.  This is usually the result of calling `fram
  df[f'{col}_std'] = df_processed[col].rolling(window=100).std()
/tmp/ipython-input-19442847.py:8: PerformanceWarning: DataFrame is highly fragmented.  This is usually the result of calling `fram
  df[f'{col}_rms'] = (df_processed[col]**2).rolling(window=100).mean()**0.5
/tmp/ipython-input-19442847.py:6: PerformanceWarning: DataFrame is highly fragmented.  This is usually the result of calling `fram
  df[f'{col}_mean'] = df_processed[col].rolling(window=100).mean()
/tmp/ipython-input-19442847.py:7: PerformanceWarning: DataFrame is highly fragmented.  This is usually the result of calling `fram
  df[f'{col}_std'] = df_processed[col].rolling(window=100).std()
/tmp/ipython-input-19442847.py:8: PerformanceWarning: DataFrame is highly fragmented.  This is usually the result of calling `fram
  df[f'{col}_rms'] = (df_processed[col]**2).rolling(window=100).mean()**0.5
/tmp/ipython-input-19442847.py:6: PerformanceWarning: DataFrame is highly fragmented.  This is usually the result of calling `fram
  df[f'{col}_mean'] = df_processed[col].rolling(window=100).mean()
/tmp/ipython-input-19442847.py:7: PerformanceWarning: DataFrame is highly fragmented.  This is usually the result of calling `fram
```

```
  df[f'{col}_std'] = df_processed[col].rolling(window=100).std()
/tmp/ipython-input-19442847.py:8: PerformanceWarning: DataFrame is highly fragmented.  This is usually the result of calling `fram
  df[f'{col}_rms'] = (df_processed[col]**2).rolling(window=100).mean()**0.5
/tmp/ipython-input-19442847.py:6: PerformanceWarning: DataFrame is highly fragmented.  This is usually the result of calling `fram
  df[f'{col}_mean'] = df_processed[col].rolling(window=100).mean()
/tmp/ipython-input-19442847.py:7: PerformanceWarning: DataFrame is highly fragmented.  This is usually the result of calling `fram
  df[f'{col}_std'] = df_processed[col].rolling(window=100).std()
/tmp/ipython-input-19442847.py:8: PerformanceWarning: DataFrame is highly fragmented.  This is usually the result of calling `fram
  df[f'{col}_rms'] = (df_processed[col]**2).rolling(window=100).mean()**0.5
/tmp/ipython-input-19442847.py:6: PerformanceWarning: DataFrame is highly fragmented.  This is usually the result of calling `fram
  df[f'{col}_mean'] = df_processed[col].rolling(window=100).mean()
/tmp/ipython-input-19442847.py:7: PerformanceWarning: DataFrame is highly fragmented.  This is usually the result of calling `fram
  df[f'{col}_std'] = df_processed[col].rolling(window=100).std()
/tmp/ipython-input-19442847.py:8: PerformanceWarning: DataFrame is highly fragmented.  This is usually the result of calling `fram
  df[f'{col}_rms'] = (df_processed[col]**2).rolling(window=100).mean()**0.5
/tmp/ipython-input-19442847.py:6: PerformanceWarning: DataFrame is highly fragmented.  This is usually the result of calling `fram
  df[f'{col}_mean'] = df_processed[col].rolling(window=100).mean()
/tmp/ipython-input-19442847.py:7: PerformanceWarning: DataFrame is highly fragmented.  This is usually the result of calling `fram
  df[f'{col}_std'] = df_processed[col].rolling(window=100).std()
/tmp/ipython-input-19442847.py:8: PerformanceWarning: DataFrame is highly fragmented.  This is usually the result of calling `fram
  df[f'{col}_rms'] = (df_processed[col]**2).rolling(window=100).mean()**0.5
```

```python
from sklearn.model_selection import train_test_split

X = df.drop(['Target'], axis=1)
y = df['Target']

# Convert target to integer by mapping string labels to numerical values
y = y.map({'Normal': 0, 'Sleep Apnea': 1})


X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)
```

```python
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report

model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

y_pred = model.predict(X_test)
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

      Normal       0.52      0.55      0.54       245
 Sleep Apnea       0.57      0.54      0.55       268

    accuracy                           0.54       513
   macro avg       0.54      0.54      0.54       513
weighted avg       0.55      0.54      0.54       513
```

```python
from sklearn.preprocessing import StandardScaler

# Ensure target is integer
y_train = y_train.astype('int')
y_test = y_test.astype('int')

# Normalize features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Deep Learning model
from tensorflow.keras import layers, models

model = models.Sequential([
    layers.Input(shape=(X_train_scaled.shape[1],)),
    layers.Dense(128, activation='relu'),
    layers.Dropout(0.3),
    layers.Dense(64, activation='relu'),
    layers.Dense(1, activation='sigmoid')
])

model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
model.fit(X_train_scaled, y_train, epochs=10, batch_size=32, validation_split=0.2)
```

```
Epoch 1/10
52/52 ━━━━━━━━━━━━━━━━━━━━ 3s 18ms/step - accuracy: 0.4868 - loss: 1.6767 - val_accuracy: 0.4878 - val_loss: 1.0876
```

```
Epoch 2/10
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - accuracy: 0.5267 - loss: 1.3922 - val_accuracy: 0.4780 - val_loss: 1.3728
Epoch 3/10
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - accuracy: 0.5338 - loss: 1.3162 - val_accuracy: 0.4805 - val_loss: 1.2130
Epoch 4/10
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - accuracy: 0.4924 - loss: 1.2206 - val_accuracy: 0.5098 - val_loss: 0.8825
Epoch 5/10
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - accuracy: 0.5208 - loss: 1.1455 - val_accuracy: 0.5000 - val_loss: 0.8810
Epoch 6/10
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - accuracy: 0.5238 - loss: 0.9828 - val_accuracy: 0.5122 - val_loss: 1.1007
Epoch 7/10
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - accuracy: 0.5515 - loss: 0.9622 - val_accuracy: 0.5122 - val_loss: 0.8890
Epoch 8/10
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - accuracy: 0.5787 - loss: 0.8847 - val_accuracy: 0.4512 - val_loss: 0.8391
Epoch 9/10
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - accuracy: 0.5772 - loss: 0.8050 - val_accuracy: 0.5122 - val_loss: 0.8024
Epoch 10/10
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 17ms/step - accuracy: 0.5465 - loss: 0.7966 - val_accuracy: 0.4902 - val_loss: 0.7769
<keras.src.callbacks.history.History at 0x7888bacc9090>
```

Double-click (or enter) to edit

```
loss, accuracy = model.evaluate(X_test_scaled, y_test)
print(f"Test accuracy: {accuracy:.2f}")
```

```
17/17 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.4889 - loss: 0.7876
Test accuracy: 0.51
```

```
# For ML
import joblib
joblib.dump(model, 'apnea_rf_model.pkl')

# For Deep Learning
model.save('apnea_nn_model.h5')
```

```
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is c
```

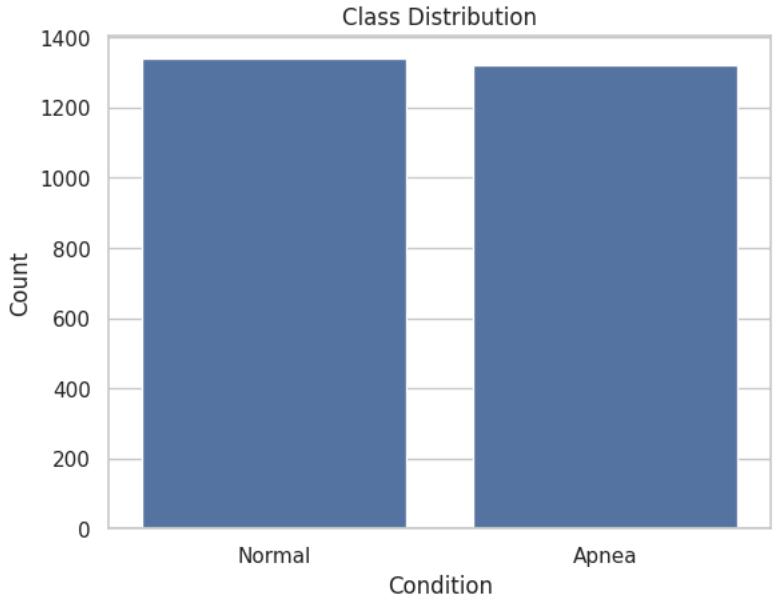
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Optional: make plots pretty
sns.set(style="whitegrid")
```

```
print(type(df))
print(df.head())
```

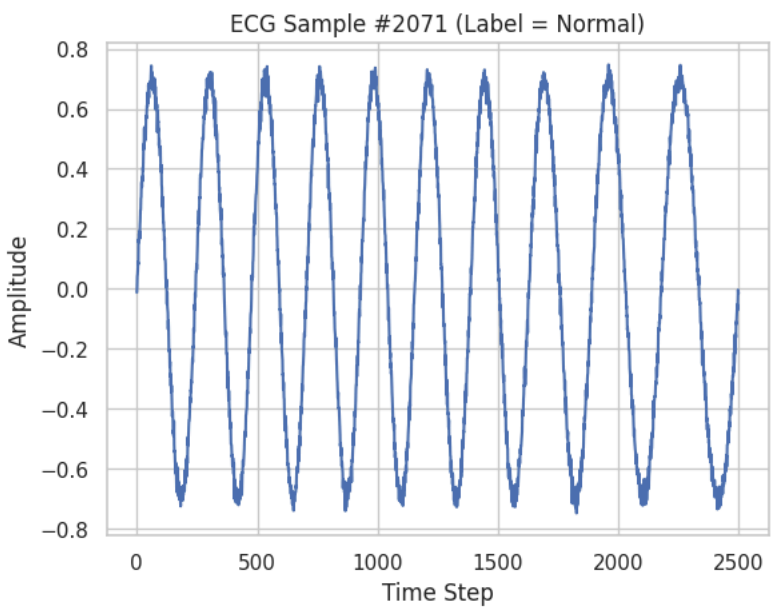
```
<class 'pandas.core.frame.DataFrame'>
           Target  0_mean_mean  0_mean_std  0_mean_rms  0_std_mean  0_std_std  \
198  Sleep Apnea    -0.002379    0.003358    0.004102    0.055500   0.010272
199  Sleep Apnea    -0.002314    0.003304    0.004020    0.055201   0.010295
200  Sleep Apnea    -0.002244    0.003247    0.003933    0.054904   0.010307
201  Sleep Apnea    -0.002180    0.003191    0.003852    0.054605   0.010310
202       Normal    -0.002116    0.003127    0.003762    0.054308   0.010307

     0_std_rms  0_rms_mean  0_rms_std  0_rms_rms  ...  2499_mean_mean  \
198   0.056433    0.055351   0.010346   0.056300  ...        0.001953
199   0.056143    0.055048   0.010365   0.056006  ...        0.002005
200   0.055853    0.054748   0.010372   0.055712  ...        0.002053
201   0.055560    0.054447   0.010371   0.055416  ...        0.002093
202   0.055267    0.054146   0.010363   0.055119  ...        0.002130

     2499_mean_std  2499_mean_rms  2499_std_mean  2499_std_std  2499_std_rms  \
198       0.002588       0.003232       0.037779      0.005067      0.038114
199       0.002565       0.003245       0.037752      0.005052      0.038085
200       0.002537       0.003254       0.037726      0.005037      0.038058
201       0.002512       0.003260       0.037703      0.005023      0.038033
202       0.002486       0.003264       0.037680      0.005009      0.038009

     2499_rms_mean  2499_rms_std  2499_rms_rms       Target
198       0.037731      0.005028      0.038061  Sleep Apnea
199       0.037704      0.005013      0.038033  Sleep Apnea
200       0.037680      0.005000      0.038007  Sleep Apnea
201       0.037657      0.004986      0.037982  Sleep Apnea
202       0.037635      0.004973      0.037959       Normal

[5 rows x 22502 columns]
```

```
df = pd.read_csv('ecg_sleep_apnea_dataset.csv')  # Or your correct path
```

```
import seaborn as sns
import matplotlib.pyplot as plt

sns.countplot(x='Target', data=df)
plt.title('Class Distribution')
plt.xticks([0, 1], ['Normal', 'Apnea'])
plt.ylabel('Count')
plt.xlabel('Condition')
plt.show()
```

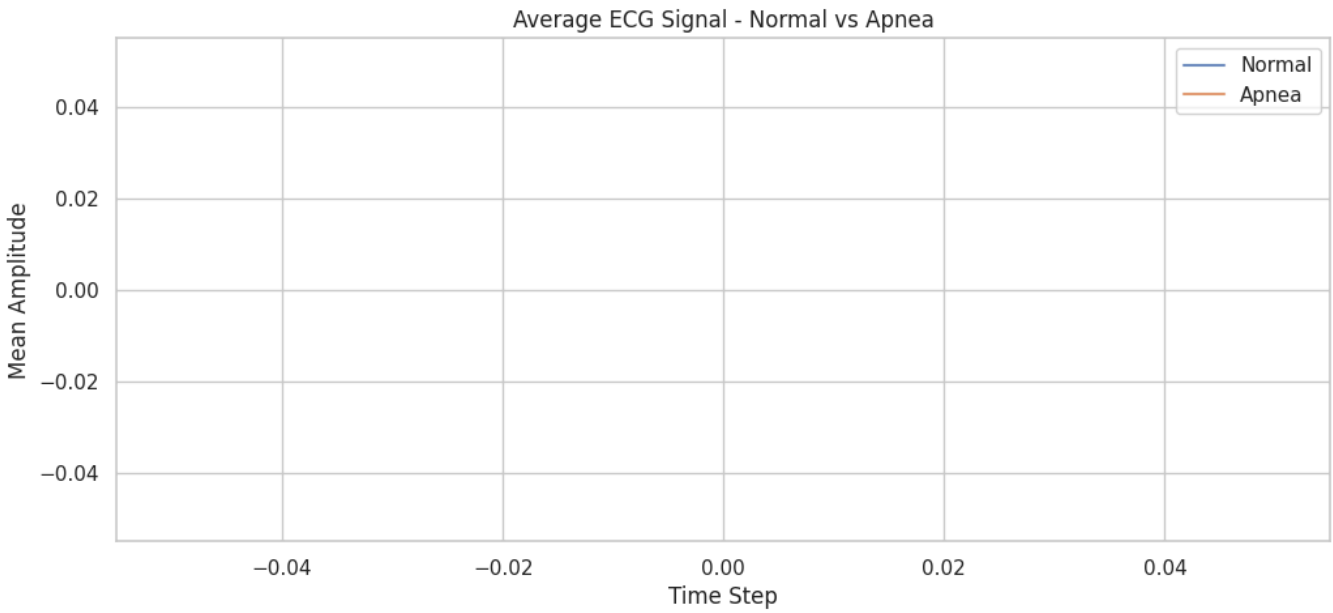



```
# Pick a random row and drop the 'Target'
sample_index = np.random.randint(0, len(df))
ecg_signal = df.drop('Target', axis=1).iloc[sample_index].astype(float)

plt.plot(ecg_signal.values)
plt.title(f"ECG Sample #{sample_index} (Label = {df['Target'].iloc[sample_index]})")
plt.xlabel('Time Step')
plt.ylabel('Amplitude')
plt.grid(True)
plt.show()
```

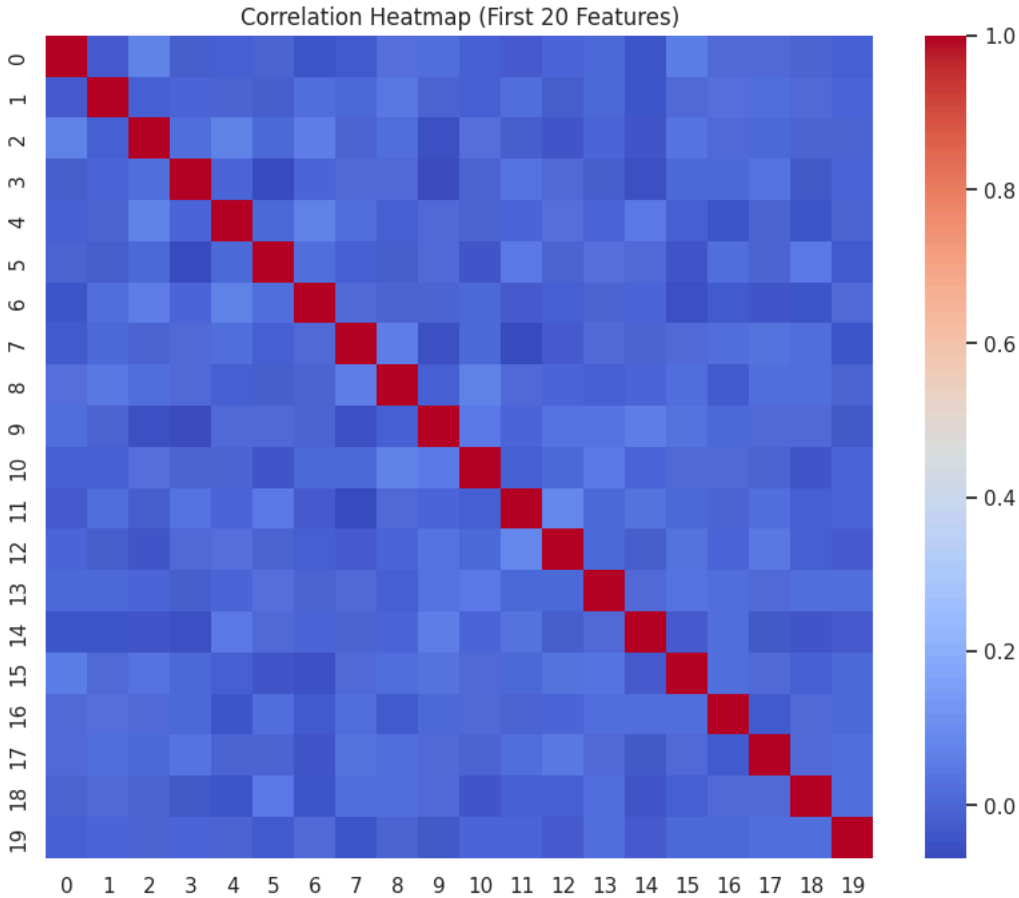



```
# Convert all signal values to float
normal = df[df['Target'] == 0].drop('Target', axis=1).astype(float)
apnea = df[df['Target'] == 1].drop('Target', axis=1).astype(float)
```

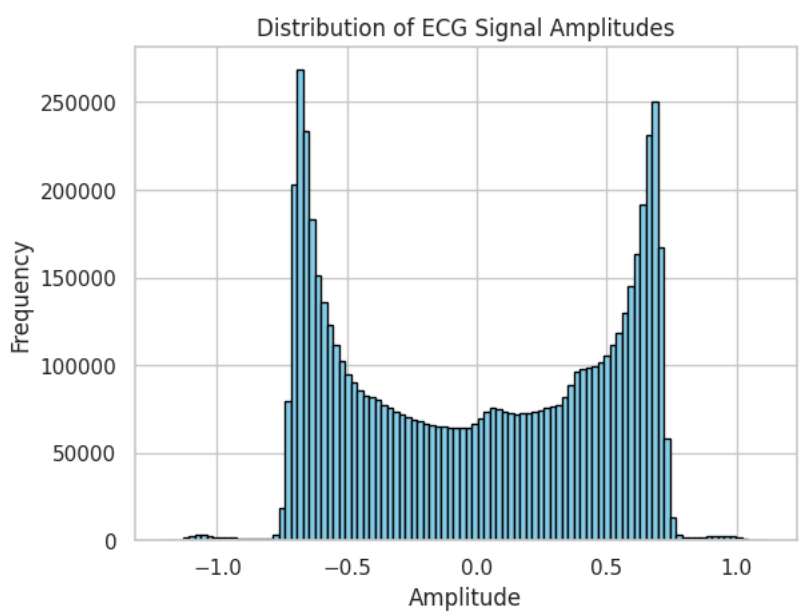
```
mean_normal = normal.mean()
mean_apnea = apnea.mean()

plt.figure(figsize=(12, 5))
plt.plot(mean_normal.values, label='Normal', alpha=0.8)
plt.plot(mean_apnea.values, label='Apnea', alpha=0.8)
plt.title('Average ECG Signal - Normal vs Apnea')
plt.xlabel('Time Step')
plt.ylabel('Mean Amplitude')
plt.legend()
plt.grid(True)
plt.show()
```




```
subset = df.iloc[:, :20]  # Use first 20 ECG features to avoid memory issues
plt.figure(figsize=(10, 8))
sns.heatmap(subset.corr(), cmap='coolwarm', annot=False)
plt.title("Correlation Heatmap (First 20 Features)")
plt.show()
```

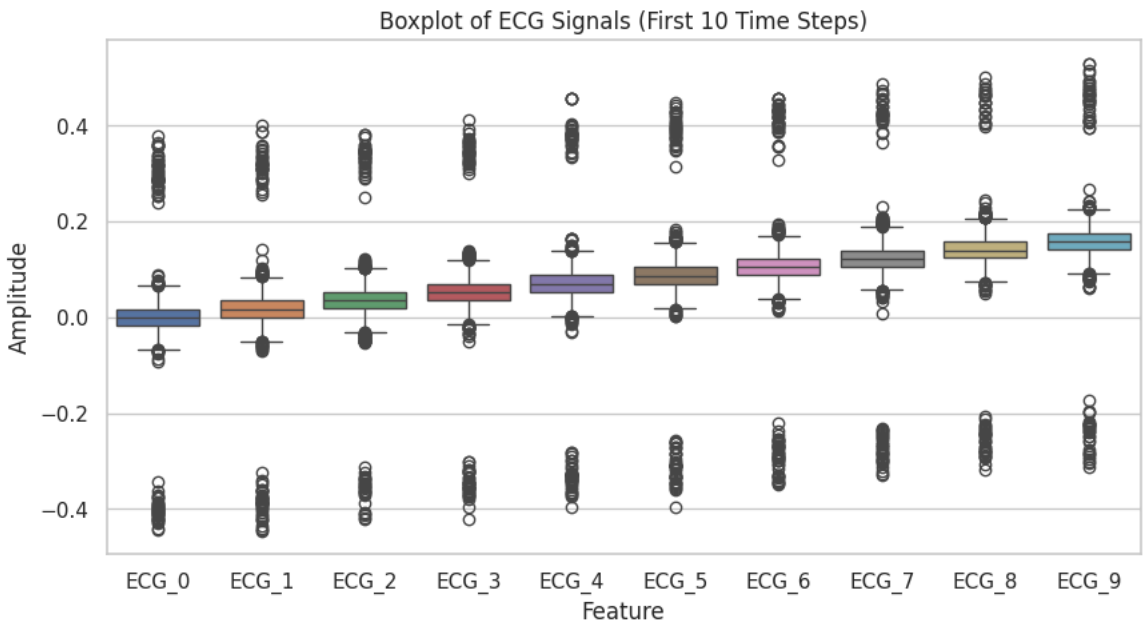



```
# Flatten the entire ECG matrix for distribution
ecg_values = df.drop('Target', axis=1).values.flatten().astype(float)

plt.hist(ecg_values, bins=100, color='skyblue', edgecolor='black')
plt.title("Distribution of ECG Signal Amplitudes")
plt.xlabel("Amplitude")
plt.ylabel("Frequency")
plt.grid(True)
plt.show()
```



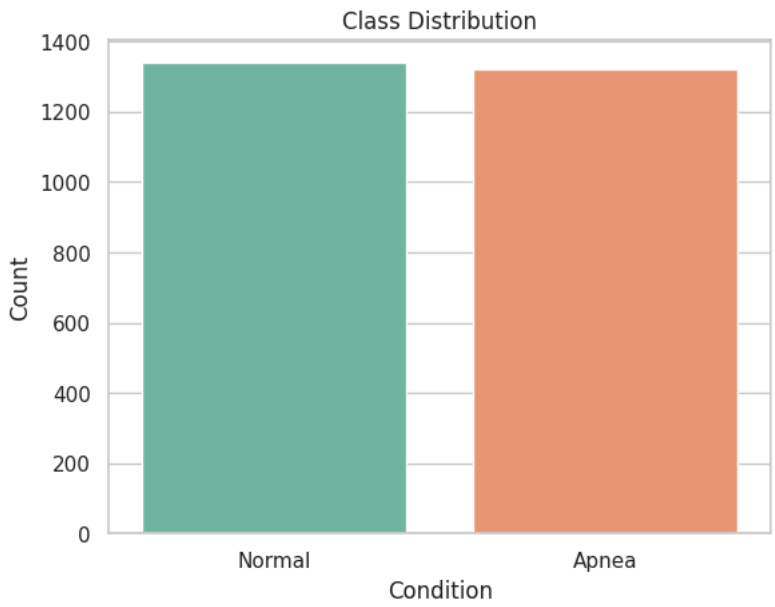

```
subset = df.iloc[:, :10].astype(float)  # first 10 features
subset.columns = [f"ECG_{i}" for i in range(10)]

plt.figure(figsize=(10, 5))
sns.boxplot(data=subset)
plt.title("Boxplot of ECG Signals (First 10 Time Steps)")
plt.xlabel("Feature")
```

```
plt.ylabel("Amplitude")
plt.show()
```

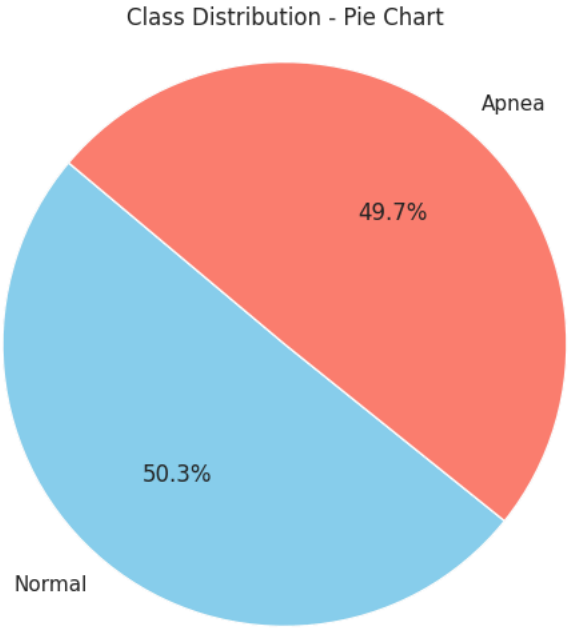



```
sns.countplot(x='Target', data=df, palette='Set2')
plt.title('Class Distribution')
plt.xticks([0, 1], ['Normal', 'Apnea'])
plt.ylabel('Count')
plt.xlabel('Condition')
plt.show()
```

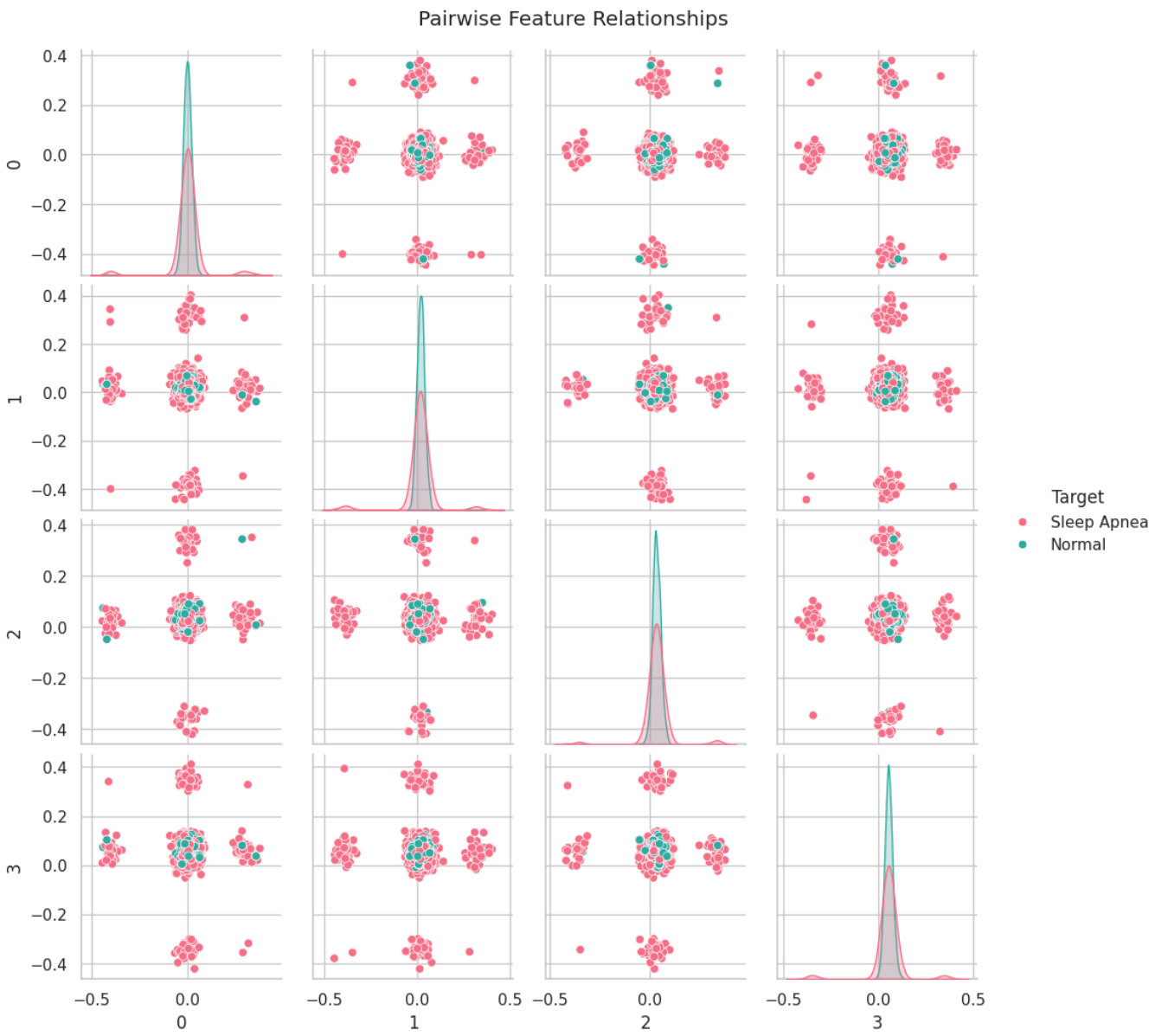
```
/tmp/ipython-input-1903269681.py:1: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `le

  sns.countplot(x='Target', data=df, palette='Set2')
```



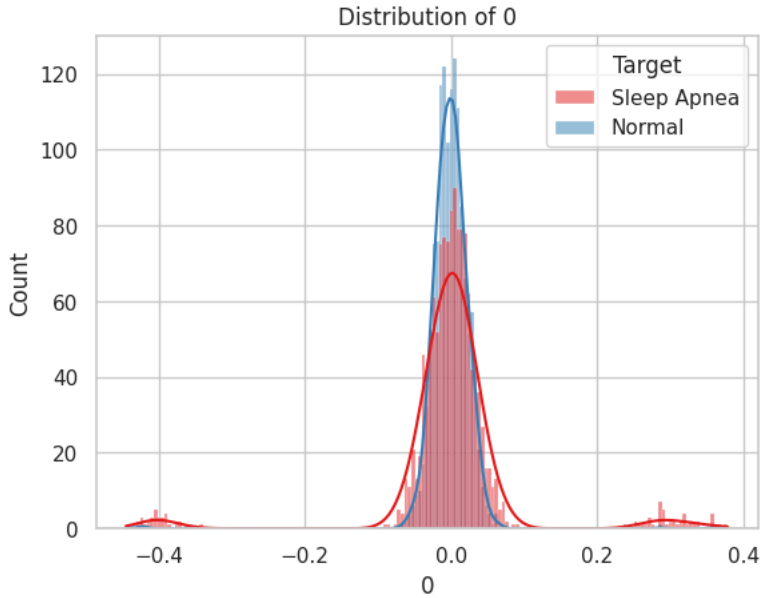

```
labels = ['Normal', 'Apnea']
sizes = df['Target'].value_counts()
plt.figure(figsize=(6,6))
plt.pie(sizes, labels=labels, autopct='%1.1f%%', startangle=140, colors=['skyblue', 'salmon'])
plt.title('Class Distribution - Pie Chart')
plt.axis('equal')
plt.show()
```

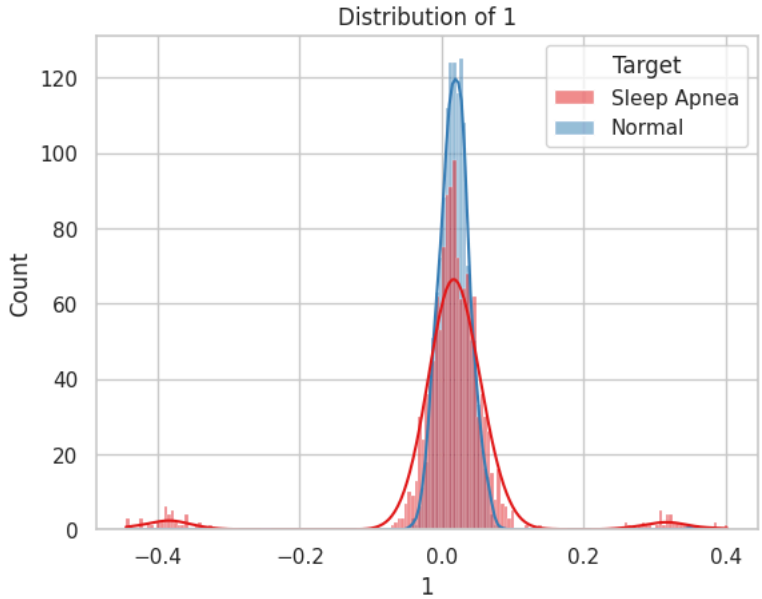


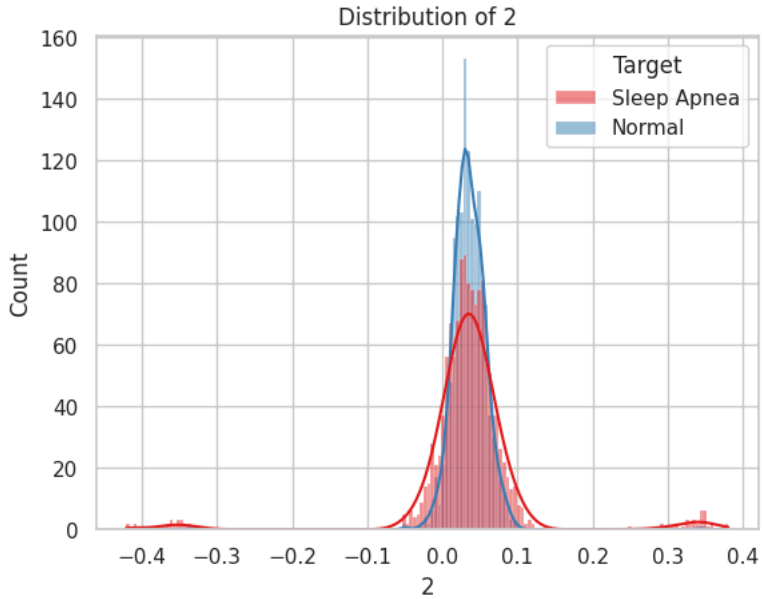

```
# You can change to top 4 numeric columns
selected_features = df.select_dtypes(include=[np.number]).columns[:4].tolist()
sns.pairplot(df[selected_features + ['Target']], hue='Target', palette='husl')
plt.suptitle('Pairwise Feature Relationships', y=1.02)
plt.show()
```

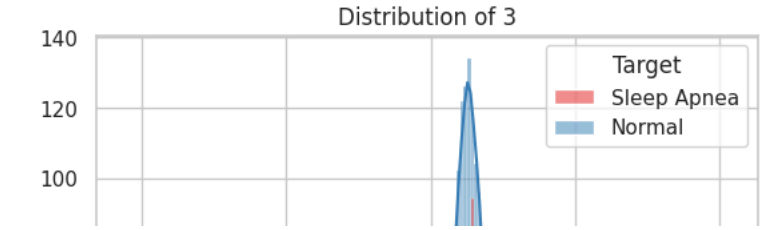


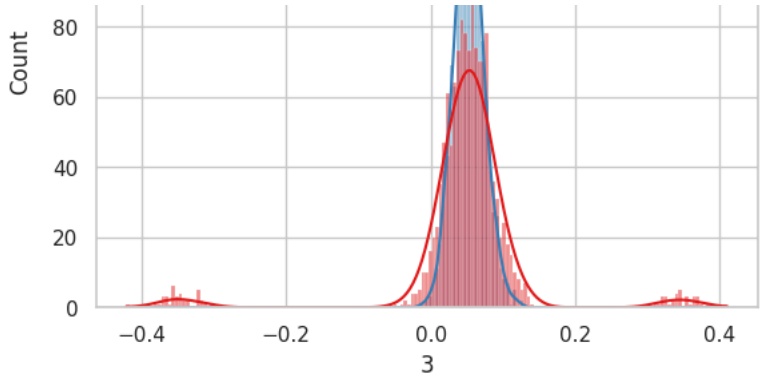

```
numerical_cols = df.select_dtypes(include=np.number).columns[:4]
for col in numerical_cols:
    plt.figure()
    sns.histplot(data=df, x=col, hue='Target', kde=True, palette='Set1')
    plt.title(f'Distribution of {col}')
    plt.show()
```


Distribution of 0
Target
Sleep Apnea
Normal
Count
0
120
100
80
60
40
20
0
−0.4
−0.2
0.0
0.2
0.4


Distribution of 1
Target
Sleep Apnea
Normal
Count
1
120
100
80
60
40
20
0
−0.4
−0.2
0.0
0.2
0.4

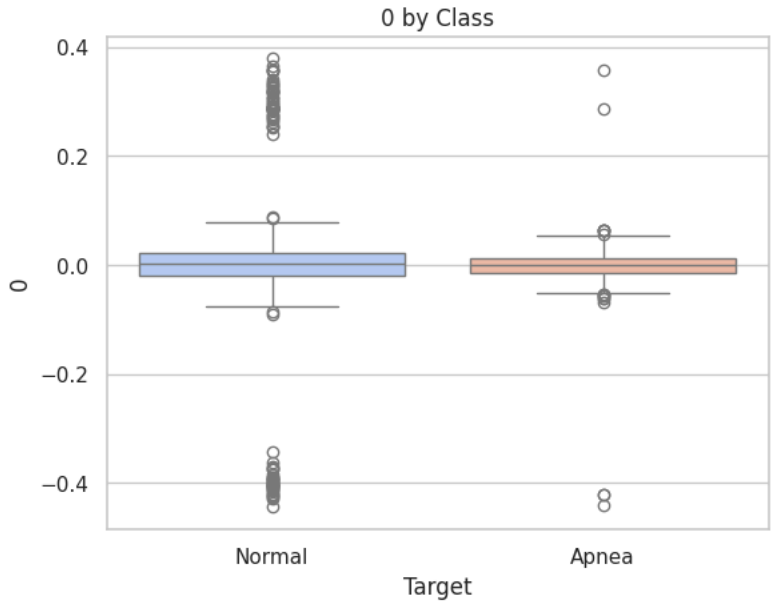

Distribution of 2
Target
Sleep Apnea
Normal
Count
2
160
140
120
100
80
60
40
20
0
−0.4
−0.3
−0.2
−0.1
0.0
0.1
0.2
0.3
0.4


Distribution of 3
Target
Sleep Apnea
Normal
140
120
100

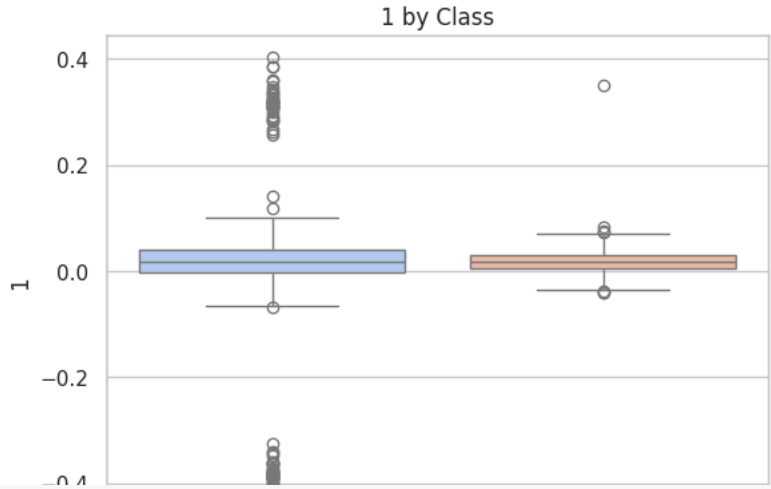



```
for col in numerical_cols:
    plt.figure()
    sns.boxplot(x='Target', y=col, data=df, palette='coolwarm')
    plt.title(f'{col} by Class')
    plt.xticks([0, 1], ['Normal', 'Apnea'])
    plt.show()
```

```
/tmp/ipython-input-1331057851.py:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `

  sns.boxplot(x='Target', y=col, data=df, palette='coolwarm')
```

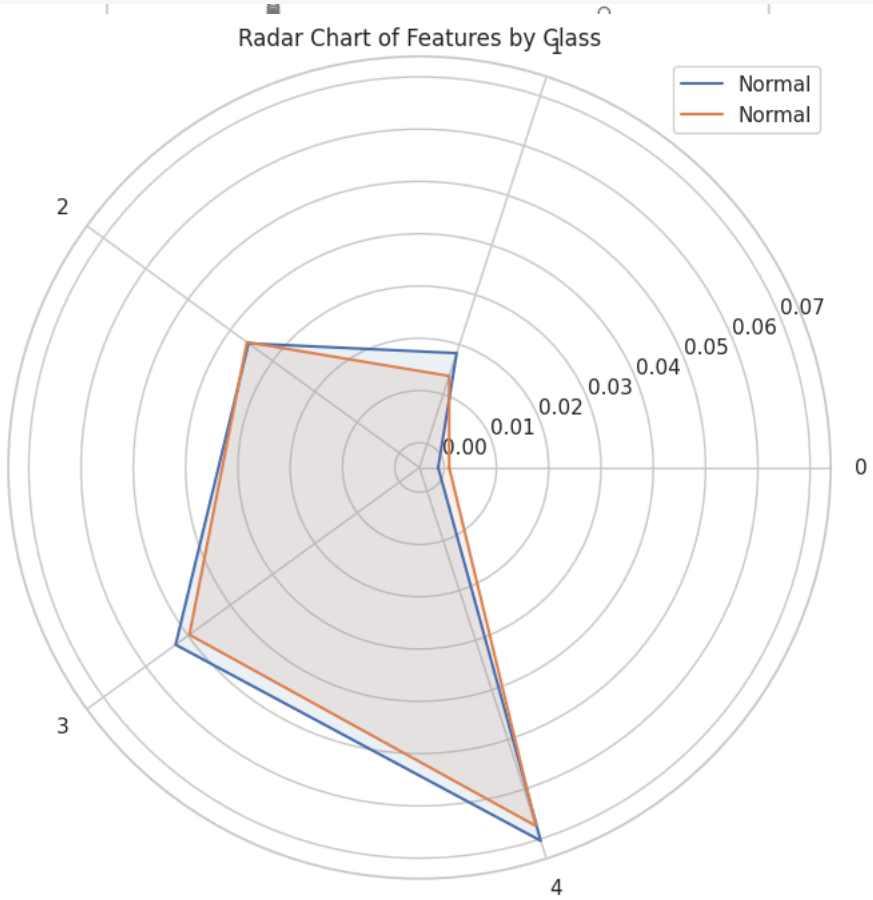



```
/tmp/ipython-input-1331057851.py:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `

  sns.boxplot(x='Target', y=col, data=df, palette='coolwarm')
```




```
from math import pi

# Pick a few features for radar
features = df.select_dtypes(include=np.number).columns[:5]
```

```
grouped = df.groupby('Target')[features].mean()

# Setup for Radar
labels = features
angles = np.linspace(0, 2 * np.pi, len(labels), endpoint=False).tolist()
angles += angles[:1]  # close the loop

fig = plt.figure(figsize=(8,8))
ax = plt.subplot(111, polar=True)

for i, row in grouped.iterrows():
    values = row.tolist()
    values += values[:1]
    label = 'Apnea' if i == 1 else 'Normal'
    ax.plot(angles, values, label=label)
    ax.fill(angles, values, alpha=0.1)

ax.set_thetagrids(np.degrees(angles[:-1]), labels)
plt.title('Radar Chart of Features by Class')
plt.legend(loc='upper right')
plt.show()
```




```
. Assign the `x` variable to `hue` and set `
```

```
for col in numerical_cols:
    plt.figure()
    sns.violinplot(x='Target', y=col, data=df, palette='Set3', inner='quartile')
    plt.title(f'Violin Plot of {col}')
    plt.xticks([0, 1], ['Normal', 'Apnea'])
    plt.show()
```