Nurse Stress Prediction Wearable Sensors

Data Source: https://www.kaggle.com/datasets/priyankraval/nurse-stress-prediction-wearable-sensors/data

Author: Woraphon Pharapromrat

```
# Import library for analysis
import pandas as pd
import numpy as np

import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, accuracy_score
from sklearn.preprocessing import StandardScaler

from sklearn.metrics import confusion_matrix
```

```
# Load csv file
df = pd.read_csv('Nurse_Stress_Prediction_Sensors.csv')
```

```
<ipython-input-2-da7dffa619b1>:2: DtypeWarning: Columns (6) have mixed ty
  df = pd.read_csv('Nurse_Stress_Prediction_Sensors.csv')
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11509051 entries, 0 to 11509050
Data columns (total 9 columns):
 #   Column    Dtype
---  ------    -----
 0   X         float64
 1   Y         float64
 2   Z         float64
 3   EDA       float64
 4   HR        float64
 5   TEMP      float64
 6   id        object
 7   datetime  object
 8   label     float64
```

```
# Take a smaller 1% chunk of the data to make it easier for Jupyter Notebo
df = df.sample(frac=0.01, random_state=37)

# Save the sampled data to a new CSV file
df.to_csv('nurse_stress_factor_1percent.csv', index=False)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 115091 entries, 10472456 to 5987762
Data columns (total 9 columns):
 #   Column    Non-Null Count   Dtype  
---  ------    --------------   -----  
 0   X         115091 non-null  float64
 1   Y         115091 non-null  float64
 2   Z         115091 non-null  float64
 3   EDA       115091 non-null  float64
 4   HR        115091 non-null  float64
 5   TEMP      115091 non-null  float64
 6   id        115091 non-null  object 
 7   datetime  115091 non-null  object 
 8   label     115091 non-null  float64
dtypes: float64(7), object(2)
memory usage: 8.8+ MB
```

Dataset Column Descriptions

- `X`, `Y`, `Z`: These columns represent accelerometer data in three dimensions. They are numeric and measure movement or orientation.
- `EDA`: Stands for Electrodermal Activity. It's a physiological marker of emotional or physiological arousal.
- `HR`: Heart Rate, typically measured in beats per minute. It's a key indicator of physical exertion or stress.
- `TEMP`: Body Temperature in Celsius. Variances in body temperature can be indicative of health conditions.
- `id`: A unique identifier for each participant or recording session.
- `datetime`: The timestamp for each data recording, important for time-series analysis.
- `label`: This column indicates varying levels of stress, with 0 representing low stress, 1 representing moderate stress, and 2 indicating high stress.

```
df.head()
```

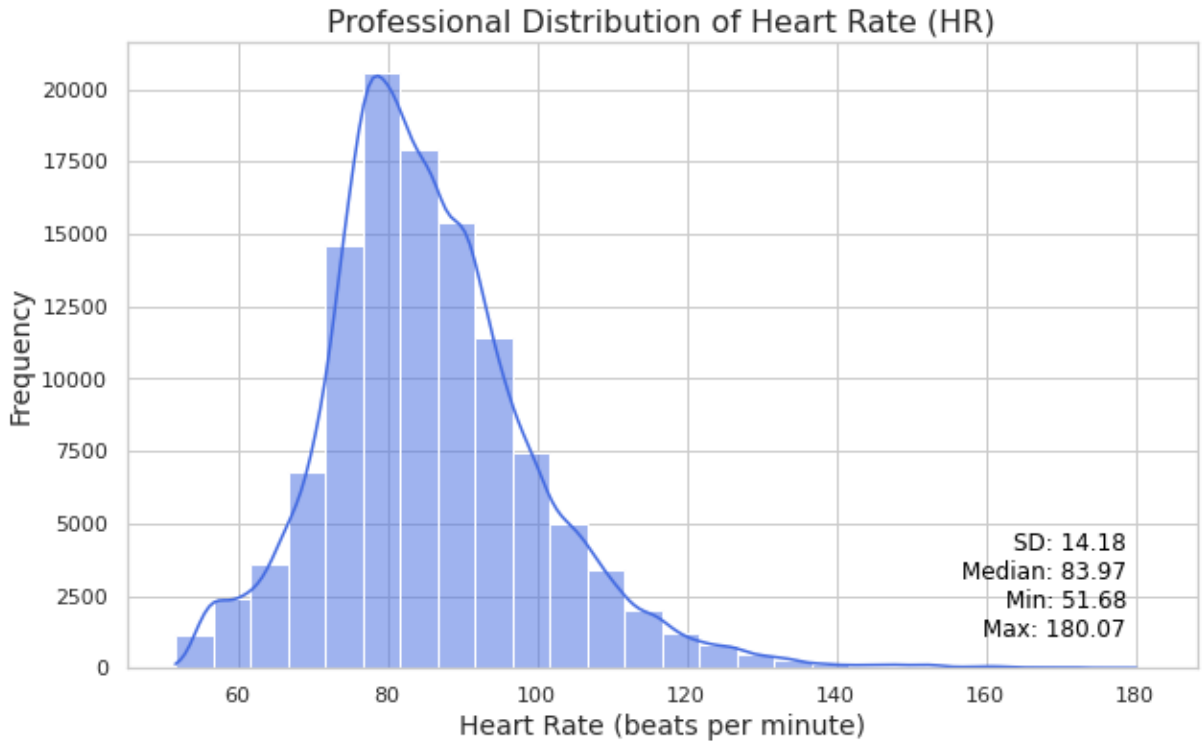
	X	Y	Z	EDA	HR	TEMP	id	datetime	label
10472456	44.0	-5.0	44.0	0.473021	88.62	32.63	EG	2020-11-08 16:15:07.343749888	2.0
9158867	-22.0	-1.0	57.0	3.310991	112.82	35.11	E4	2020-04-20 15:15:25.375000064	2.0
10223432	18.0	-10.0	60.0	19.800661	67.45	35.05	E4	2020-07-03 17:10:25.531249920	0.0
8221077	-23.0	-1.0	58.0	0.084558	79.53	30.29	DF	2020-07-22 16:28:27.906249984	2.0
7728560	-59.0	6.0	26.0	0.075585	69.22	29.27	CE	2020-07-02 16:58:56.875000064	2.0

```
# Check Missing values
df.isnull().sum()
# there are no missing values.
```

```
X           0
Y           0
Z           0
EDA         0
HR          0
TEMP        0
id          0
datetime    0
label       0
dtype: int64
```

```
# Setting a style theme
sns.set_theme(style="whitegrid")

# Distribution of Heart Rate
plt.figure(figsize=(10, 6))
histplot = sns.histplot(df['HR'], kde=True, color='royalblue', binwidth=5)
plt.title('Professional Distribution of Heart Rate (HR)', fontsize=16)
plt.xlabel('Heart Rate (beats per minute)', fontsize=14)
plt.ylabel('Frequency', fontsize=14)

# Calculating descriptive statistics
sd_hr = np.std(df['HR'])
median_hr = df['HR'].median()
min_hr = df['HR'].min()
max_hr = df['HR'].max()

# Adding text for the statistics
stats_text = f"SD: {sd_hr:.2f}\nMedian: {median_hr:.2f}\nMin: {min_hr:.2f}
plt.text(x=0.95*plt.xlim()[1], y=0.05*plt.ylim()[1],
         s=stats_text, horizontalalignment='right', fontsize=12, color='bl

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

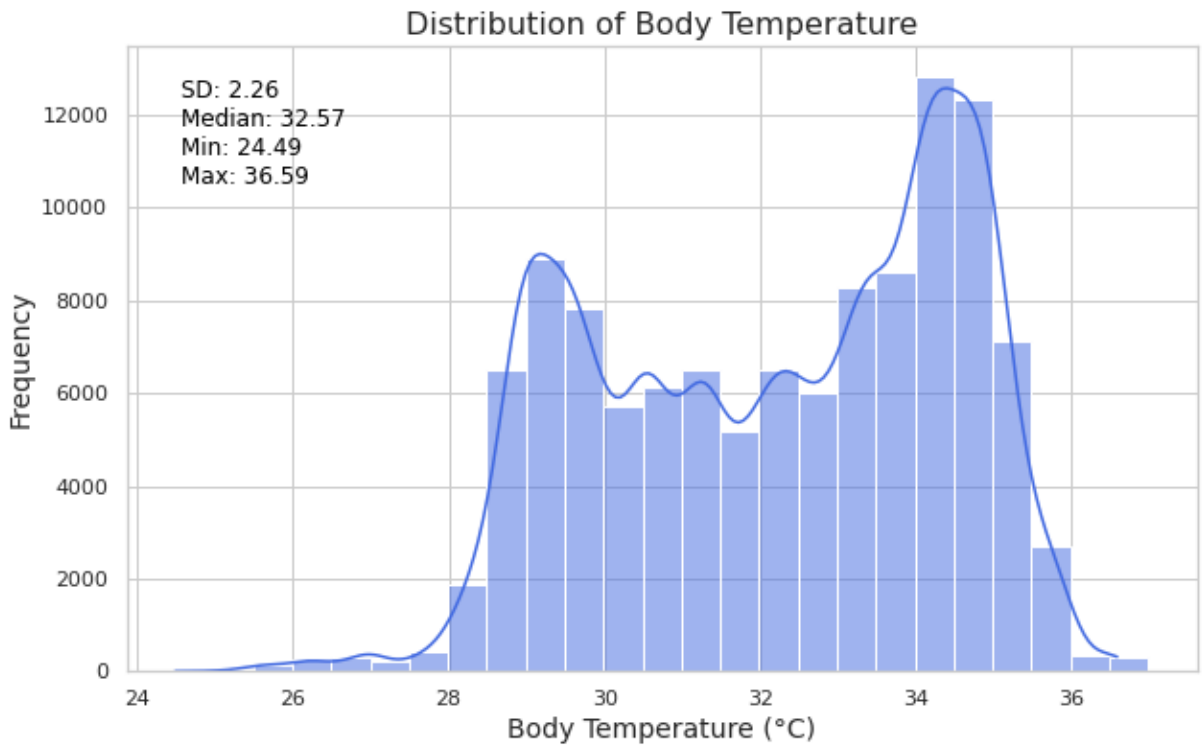
Download



- **Shape**: It normally distributed with a slight right skew.
- **Median HR**: 83.97 beats per minute, which is typical resting heart rate ranges for adults.
- **Variability**: Standard deviation is 14.18.
- **Range**: The HR values range from 51.68 bpm (minimum) to 180.07 bpm (maximum),a wide range of heart rates could be due to various factors like age, stress, activity levels,

```
# Setting a style theme
sns.set_theme(style="whitegrid")

# Distribution of Body Temperature
plt.figure(figsize=(10, 6))
histplot_temp = sns.histplot(df['TEMP'], kde=True, color='royalblue', binw
plt.title('Distribution of Body Temperature', fontsize=16)
plt.xlabel('Body Temperature (°C)', fontsize=14)
plt.ylabel('Frequency', fontsize=14)

# Calculating descriptive statistics
sd_temp = np.std(df['TEMP'])
median_temp = df['TEMP'].median()
min_temp = df['TEMP'].min()
max_temp = df['TEMP'].max()

# Adding text for the statistics - inside the chart
stats_text_temp = f"SD: {sd_temp:.2f}\nMedian: {median_temp:.2f}\nMin: {mi
plt.annotate(stats_text_temp, xy=(0.05, 0.95), xycoords='axes fraction',
             fontsize=12, color='black',
             horizontalalignment='left', verticalalignment='top')

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

Download



- **Shape**: Bimodal distributions
- **Median Temperature**: 32.57°C, Typical human body temperature averages around 37°C, the lower heart rate could be due to Environment temperature in the hospital, physical activity or

health status.
- **Variability**: 2.26
- **Range**: 24.49 to 36.59 which is broad range, some values appear unusually low for normal body temperature.

Modeling

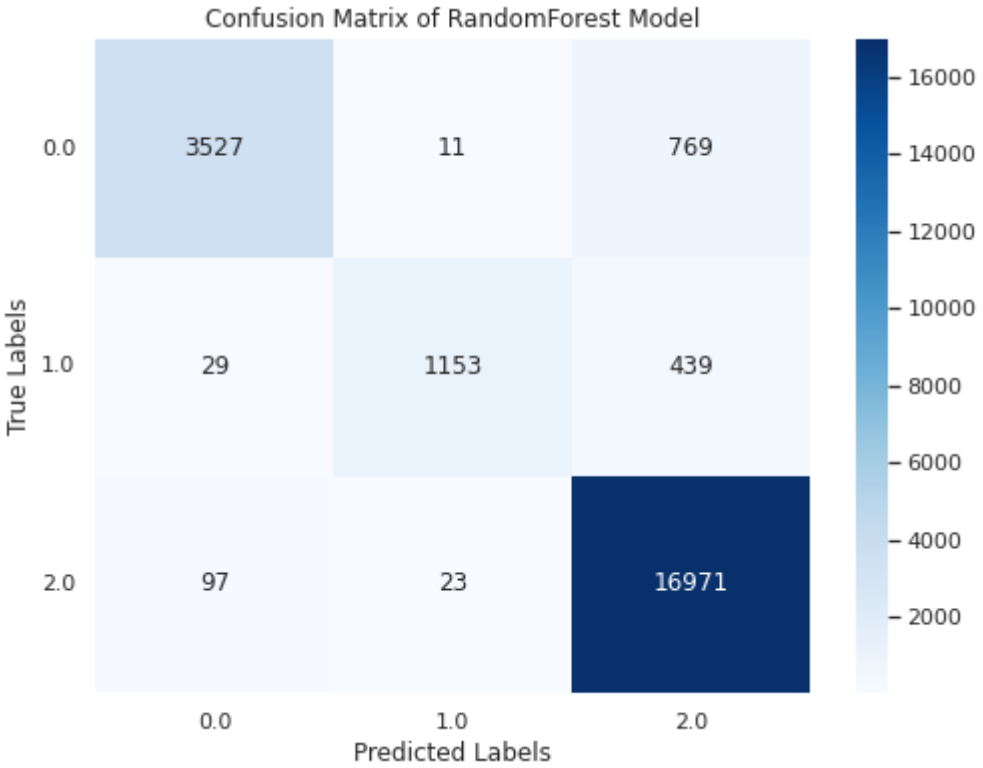
```
# Data Preprocessing
ml_data = df.drop(['id', 'datetime'], axis=1)  # Dropping non-relevant col

# Handling missing values after dropping non-relevant columns
ml_data = ml_data.dropna()

X = ml_data.drop('label', axis=1)  # Features
y = ml_data['label']  # Target variable

scaler = StandardScaler()  # Normalizing the data
X_scaled = scaler.fit_transform(X)

X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size

# Model Training
model = RandomForestClassifier(random_state=42)
model.fit(X_train, y_train)

# Model Evaluation
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred)
```

```
print(y_pred)
print(accuracy)
print(report)
```

```
[2. 1. 2. ... 0. 0. 0.]
0.9405708327903036
              precision    recall  f1-score   support

         0.0       0.97      0.82      0.89      4307
         1.0       0.97      0.71      0.82      1621
         2.0       0.93      0.99      0.96     17091

    accuracy                           0.94     23019
   macro avg       0.96      0.84      0.89     23019
weighted avg       0.94      0.94      0.94     23019
```

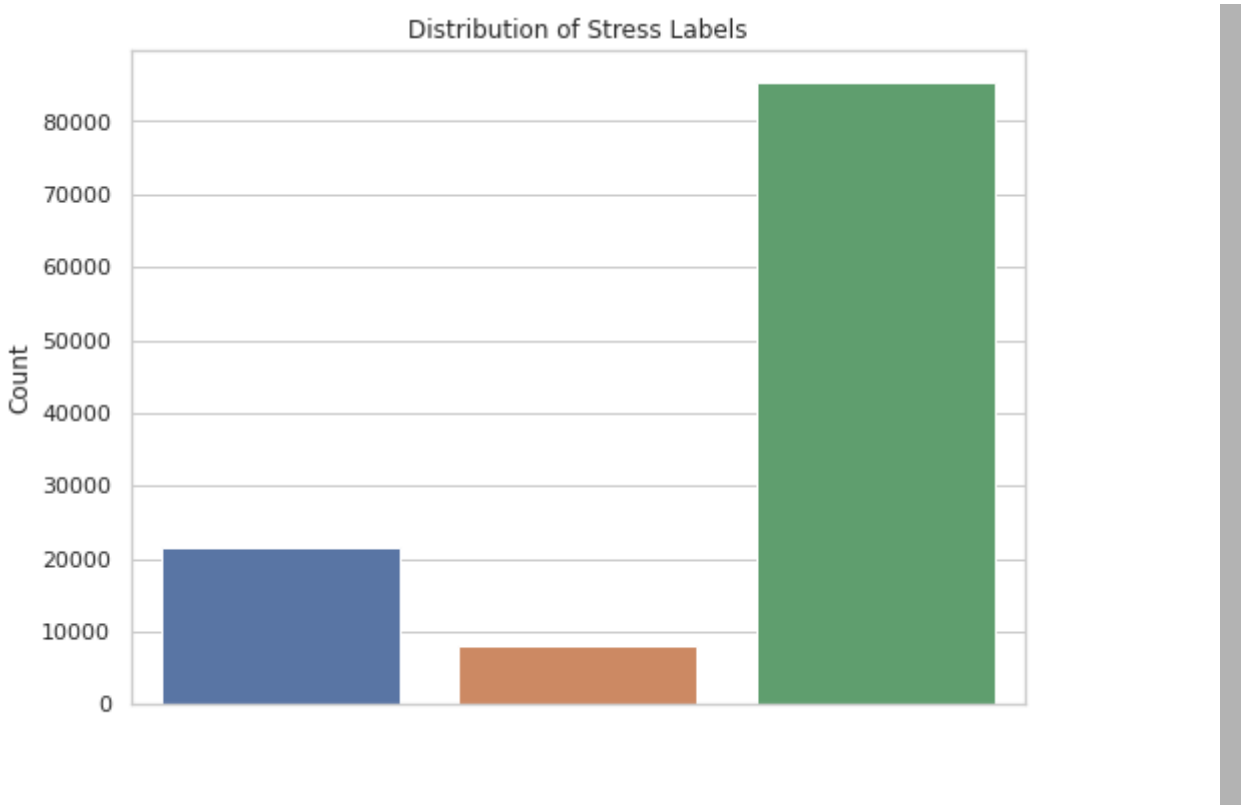
```
# Generating confusion matrix
conf_matrix = confusion_matrix(y_test, y_pred)

# Creating a heatmap for the confusion matrix
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="Blues")
plt.title('Confusion Matrix of RandomForest Model')
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.xticks(np.arange(len(np.unique(y))) + 0.5, np.unique(y))
plt.yticks(np.arange(len(np.unique(y))) + 0.5, np.unique(y), rotation=0)
plt.show()
```

Download




```
# Check the imbalance in the dataset by creating a bar chart to visualize

sns.set_theme(style="whitegrid")
# Plotting the distribution of the target variable 'label'
label_counts = df['label'].value_counts()

plt.figure(figsize=(8, 6))
sns.barplot(x=label_counts.index, y=label_counts.values)
plt.title('Distribution of Stress Labels')
plt.xlabel('Stress Label')
plt.ylabel('Count')
plt.show()
```

Download


Distribution of Stress Labels
Count
80000
70000
60000
50000
40000
30000
20000
10000
0