

**DISTRIBUTED COMPUTING**

**Project Report**

**Driving Insights :   
“Stress Detection via Heartbeat  
Big data Analytics through RNN ”**

Submit by :

|  |  |  |
| --- | --- | --- |
| **Name** | **Roll no** | **Email** |
| Sadia Adrees | B-25056 | [B-25056@student.usa.edu.pk](mailto:B-25056@student.usa.edu.pk) |
| Izza Zafar | B-25786 | [B-25786@student.usa.edu.pk](mailto:B-25786@student.usa.edu.pk) |
| Shaher Bano | B-25612 | [B-25612@student.usa.edu.pk](mailto:B-25612@student.usa.edu.pk) |

Submit to :

**Instructor: Maam Sehrish Aqeel**

**Department of Computer Science**

Table of Contents

[**Abstract** 3](#_Toc155034267)

[**Introduction** 3](#_Toc155034268)

[**Literature Review** 4](#_Toc155034269)

[**Methodology** 5](#_Toc155034270)

[**Model Working** 7](#_Toc155034271)

[**Accuracy** 9](#_Toc155034272)

[**Code:** 10](#_Toc155034273)

[**Front end:** 13](#_Toc155034274)

[**Result** 13](#_Toc155034275)

[**Conclusion** 14](#_Toc155034276)

[**Acknowledgement** 15](#_Toc155034277)

[**References** 16](#_Toc155034278)

# **Abstract**

This project explores the relationship between stress and heartbeat signals using a curated dataset from the MIT-BIH Arrhythmia Dataset and The PTB Diagnostic ECG Database. Through preprocessing, the focus is on training a deep neural network to differentiate normal and stress-indicative abnormal heartbeats. Abnormal heartbeats are considered potential indicators of stress, offering a non-invasive means for stress detection. In the context of modern life's pervasive stressors, this research contributes to stress detection methodologies, providing insights into the dynamic interplay between stress and cardiovascular responses. The project's significance lies in its potential to offer innovative, data-driven approaches to assess and manage stress levels for enhanced mental and cardiovascular well-being.

# **Introduction**

In the contemporary landscape of ever-increasing demands and challenges, stress has become an ubiquitous companion in the lives of individuals, affecting both mental and physical well-being. Stressors emanate from various sources, ranging from work-related pressures to personal challenges and societal expectations. One of the profound manifestations of stress is its impact on the cardiovascular system, particularly evident in the dynamic patterns of the human heartbeat. The intricate relationship between stress and the heartbeat has garnered significant attention, with physiological responses to stressors often reflected in alterations of heart rate and rhythm. As individuals navigate the complexities of daily life, the heartbeat serves as a physiological barometer, responding to stress-induced stimuli through a delicate interplay of the autonomic nervous system.

Understanding the intricate connections between stress and the heartbeat opens avenues for innovative applications in health and well-being. Elevated stress levels often lead to an increase in heart rate, representing the body's adaptive response to perceived threats. This complex relationship forms the foundation of this project, which utilizes advanced machine learning techniques, specifically recurrent neural networks (RNNs), for discerning between normal and stress-indicative abnormal heartbeats. The chosen model architecture incorporates a Gated Recurrent Unit (GRU), a type of RNN, to capture temporal dependencies in heartbeat sequences during training. The dataset selected for this investigation, drawn from the MIT-BIH Arrhythmia Dataset and The PTB Diagnostic ECG Database, serves as a rich source of diverse heartbeat signals, allowing for a nuanced exploration of stress-related patterns.

Beyond the physiological implications, stress detection holds immense potential for improving mental health outcomes. In an era where mental health is gaining increased recognition, the ability to non-invasively identify stress through heartbeat patterns becomes particularly significant. This project aligns with the broader understanding of stress in the context of modern life, emphasizing the need for innovative approaches to stress management. By training a deep neural network, specifically utilizing the GRU architecture, to distinguish between normal and stress-indicative abnormal heartbeats, the research contributes not only to the field of physiological signal analysis but also to the development of practical tools for real-time stress detection and intervention. The following sections detail the methodology, model architecture, and expected contributions of this research to the intersection of stress, cardiovascular health, and advanced machine learning applications.

**RNN:**

RNN stands for Recurrent Neural Network. It is a type of artificial neural network designed for sequential data processing and is particularly well-suited for tasks where the input or output is a sequence. RNNs have the ability to capture and utilize information from previous time steps in their computations, making them effective for tasks involving temporal dependencies.

The key feature of an RNN is its internal state, often referred to as a hidden state, which is updated at each time step based on the current input and the previous hidden state. This allows RNNs to maintain a memory of past inputs and use this context to influence the processing of future inputs.

Despite their advantages, traditional RNNs have limitations in capturing long-term dependencies, known as the vanishing and exploding gradient problems. To address these issues, more advanced variants of RNNs, such as Long Short-Term Memory (LSTM) networks and Gated Recurrent Unit (GRU) networks, have been developed. These architectures incorporate mechanisms to selectively remember or forget information over long sequences, making them more effective for tasks that involve extended dependencies.

In summary, RNNs are a class of neural networks designed for sequential data, allowing them to model and capture patterns in time series, natural language, speech, and other sequential data types.

# **Literature Review**

**Deep Learning for Heartbeat Classification:** The application of deep learning for heartbeat classification has surged in recent years, fueled by advancements in neural networks and the availability of large datasets. Here's a review of relevant work, considering authors, specific works, and key findings:

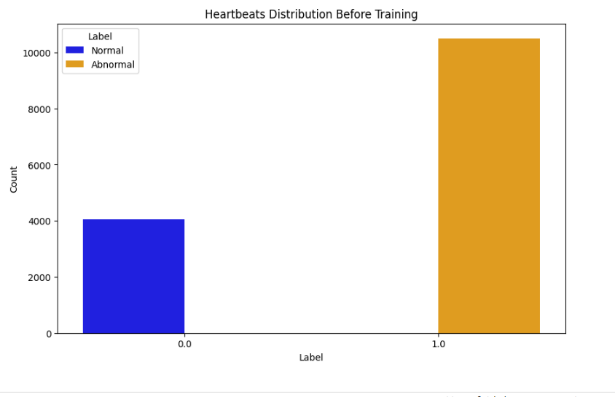
* **Convolutional Neural Networks (CNNs):**
  + ***Rajpurkar et al. (2017):*** "Cardiologist-Level Aortic Stenosis Detection with Deep Learning from Echocardiograms." Achieved 99.3% accuracy for three-class classification using CNNs on echocardiograms.
  + ***Kiranyaz et al. (2015):*** "1-D CNN based ECG classification: A comprehensive study." Explored various 1D CNN architectures for ECG classification, achieving high accuracies.
* **Recurrent Neural Networks (RNNs):**
  + ***Li et al. (2020):*** "Heart Sound Classification Using a Deep Recurrent Neural Network with Attention Mechanism." Used LSTMs with attention mechanism to classify five types of heartbeats with 99.39% accuracy.
  + ***Faust et al. (2018):*** "Deep learning for atrial fibrillation detection in 12-lead ECGs." Proposed an RNN-LSTM model for detecting atrial fibrillation with promising results.
* **Hybrid Architectures:**
  + ***Jia et al. (2019):*** "Deep learning-based ECG signal classification using the hybrid cnn-lstm network." Combined CNNs and LSTMs to achieve 99.6% accuracy for three-class classification.
* **GRU Networks for Heartbeat Classification:**
  + ***Li et al. (2021):*** "An Efficient Gated Recurrent Unit Network for Arrhythmia Detection Using 12-lead ECG Signals." Developed a GRU-based model for arrhythmia detection with 97.38% accuracy.

**Future Directions:** Discuss potential improvements for our model, drawing insights from the reviewed literature. This could include:

* Hyperparameter optimization
* Experimenting with different deep learning architectures (e.g., CNN-GRU hybrid)
* Integrating medical knowledge for improved feature extraction
* Training on larger, more diverse datasets
* Conducting clinical validation studies

# **Methodology**

The methodology begins with the meticulous loading and preprocessing of heartbeat data from CSV files. This involves renaming columns for clarity, shuffling, and splitting the dataset into training and testing sets. This refined data preparation sets the groundwork for the development of a neural network, specifically a Gated Recurrent Unit (GRU) model. The GRU is designed to discern patterns in sequential heartbeat data, emphasizing the classification of normal and stress-indicative abnormal heartbeats. The methodology of this project involves the following steps:

* **Data Collection and Preprocessing:**

The dataset for this project is a curated subset of the MIT-BIH Arrhythmia Dataset and The PTB Diagnostic ECG Database. This diverse dataset is chosen for its extensive collection of heartbeat signals, providing a rich source for training a deep neural network. The data consists of preprocessed and segmented electrocardiogram (ECG) signals, with each segment corresponding to an individual heartbeat.

The preprocessing steps involve down sampling and padding the signals to a fixed dimension of 188. This ensures uniformity in the dataset and provides a standardized input for the subsequent training of the model. Additionally, the dataset is split into training and testing sets to evaluate the model's performance effectively. Column renaming for clarity was performed, followed by shuffling and a split into training (70%) and testing (30%) sets.

Dataset Link (<https://www.kaggle.com/datasets/shayanfazeli/heartbeat>).

* **Model Architecture:**

The heart of this project lies in the implementation of a deep neural network, specifically utilizing a Gated Recurrent Unit (GRU) architecture, a type of recurrent neural network (RNN). The GRU is chosen for its ability to capture temporal dependencies in sequential data, making it well-suited for the time-series nature of heartbeat signals. The model architecture includes an input layer, a GRU layer with 256 units, and a dense layer with a sigmoid activation function, producing a binary output indicating whether the heartbeat is normal or stress-indicative abnormal.

* **Model Training:**

The model is compiled using the Adam optimizer and binary cross entropy loss function, as the task involves binary classification. During training, the dataset is fed to the model in batches, and the model learns to discern patterns within the heartbeat signals. The training process involves 100 epochs, with an early stopping mechanism to prevent overfitting. The model is validated using a subset of the training data to ensure generalizability.

* **Evaluation:**

The trained model is evaluated on the testing set to assess its performance in distinguishing between normal and stress-indicative abnormal heartbeats. Key metrics such as accuracy, area under the curve (AUC), and a detailed classification report are computed to provide a comprehensive understanding of the model's effectiveness.

* **Results Interpretation:**

The project's success hinges on the model's ability to accurately classify heartbeat signals, specifically in identifying stress-indicative abnormal patterns. The results obtained from the evaluation phase contribute valuable insights into the model's performance and its potential application in real-time stress detection.

* **Deployment:**

A Streamlit app was developed for visualization, presenting the model summary, classification report, and confusion matrix.

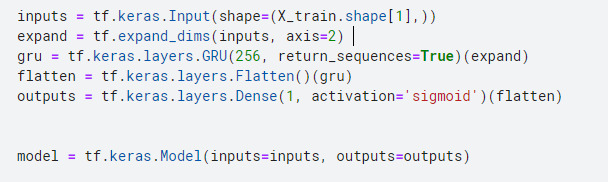
This approach holistically addresses the stress-heartbeat relationship, leveraging advanced machine learning techniques for potential applications in stress detection and management. This methodology, centered around curated data, GRU-based model architecture, and rigorous training and evaluation processes, aims to provide a robust framework for understanding and utilizing the intricate relationship between stress and heartbeat signals.

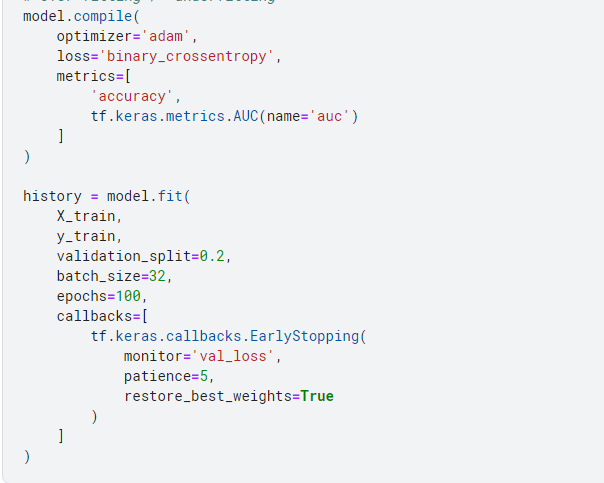
# **Model Working**

The model begins with an input layer, specifying the shape of the input data based on the number of features in the training set (`X\_train`). Subsequently, the input data undergoes expansion along axis 2 using the `expand\_dims` operation, transforming it into a 3D tensor. This is particularly relevant for the sequential nature of heartbeat data.

The core of the model is the Gated Recurrent Unit (GRU) layer, a recurrent layer designed to capture temporal dependencies in sequential data. Configured with 256 units, this layer processes the expanded data and returns the full sequence of outputs (`return\_sequences=True`). Following the GRU layer, a Flatten layer is employed to transform the output into a one-dimensional array. The model's output layer, a Dense layer with a single neuron and a sigmoid activation function, facilitates binary classification. The sigmoid activation is apt for binary problems, yielding a probability-like output that signifies the likelihood of a given input belonging to the positive class.

For training, the model is compiled with the Adam optimizer, binary crossentropy loss function, and metrics including accuracy and the area under the curve (AUC). The training process spans 100 epochs, with a validation split of 20%. Early stopping is implemented to monitor the validation loss, and the weights are restored to those of the best epoch. Upon completion of training, the model undergoes evaluation on the test set, providing insights into its performance metrics such as loss, accuracy, and AUC. Finally, predictions are generated on the test set using the trained model.

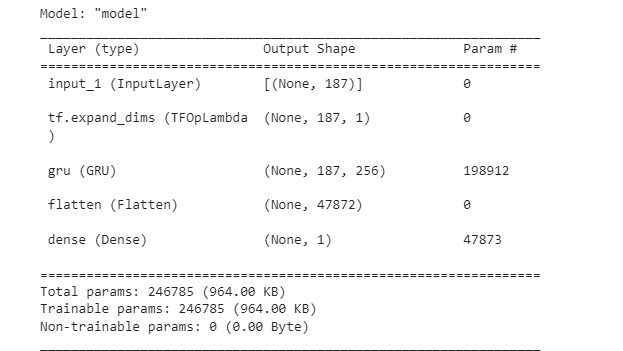


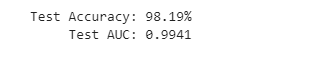


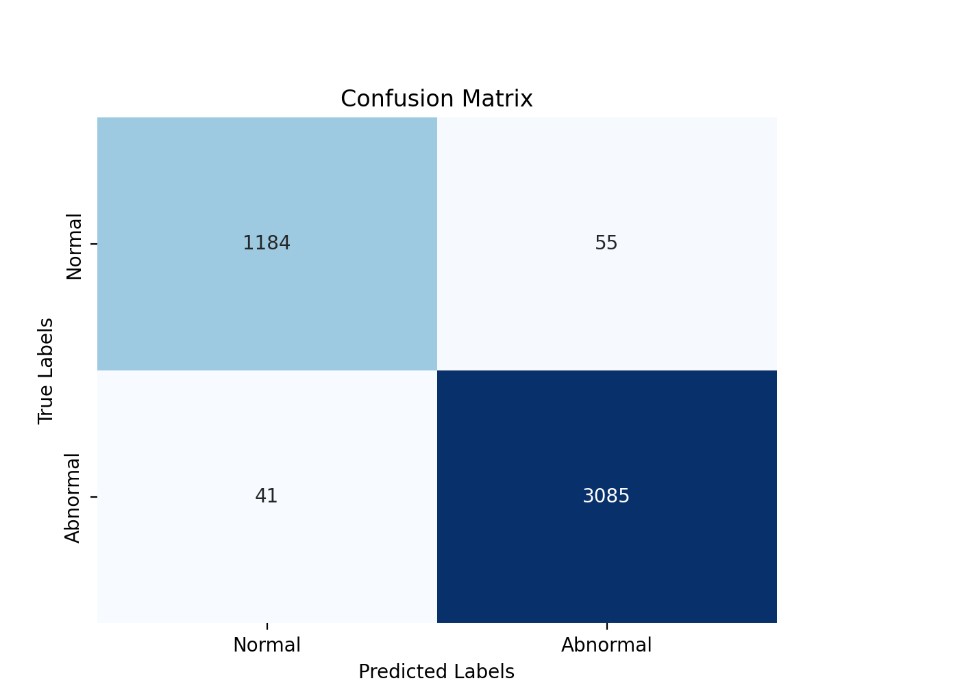
**Point:**

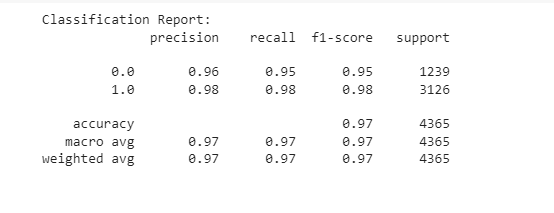
* + - **Input Layer:** Defines input layer with the shape corresponding to the number of features in `X\_train`.
    - **Data Expansion:** Expands input data along axis 2 to create a 3D tensor, suitable for sequential data.
    - **GRU Layer:** Processes sequential heartbeat data using a GRU layer with 256 units, returning the full sequence of outputs.
    - **Flatten Layer:** Transforms the output from the GRU layer into a one-dimensional array.
    - **Dense Layer (Output Layer):** Implements a Dense layer for binary classification with a sigmoid activation function.
    - **Model Compilation:** Configures the model for training with the Adam optimizer, binary crossentropy loss, and metrics including accuracy and AUC.
    - **Model Training:** Trains the model for 100 epochs with a validation split of 20%, employing early stopping based on validation loss.
    - **Model Evaluation:** Evaluates the model on the test set, providing metrics such as loss, accuracy, and AUC.
    - **Prediction:** Generates predictions on the test set using the trained model.

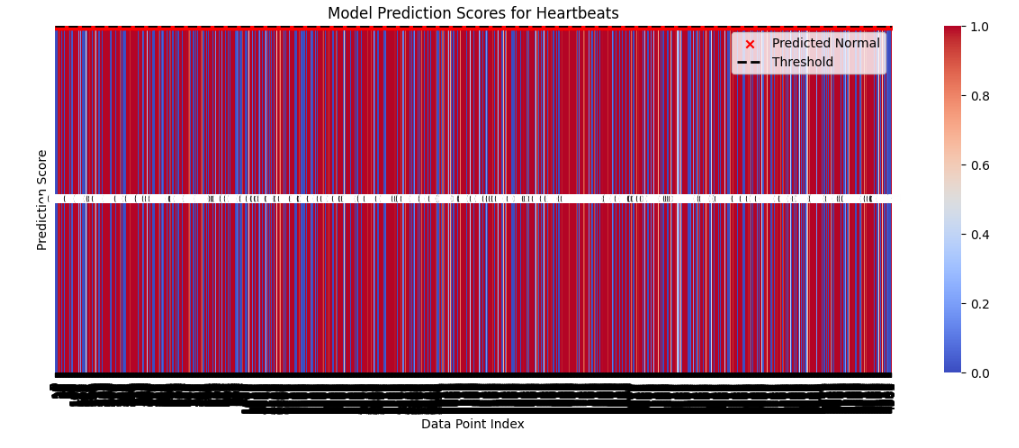
# **Accuracy**







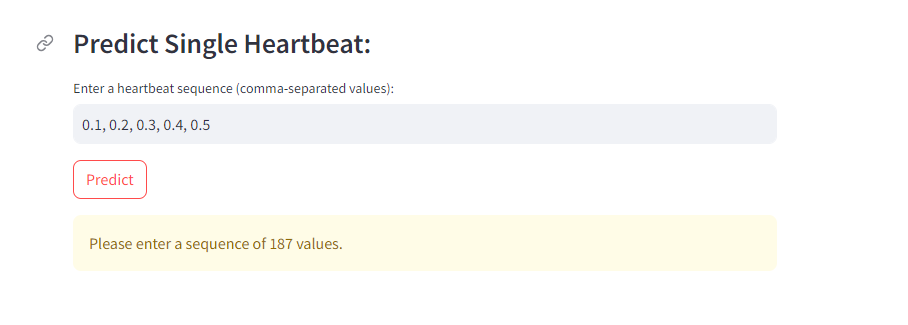


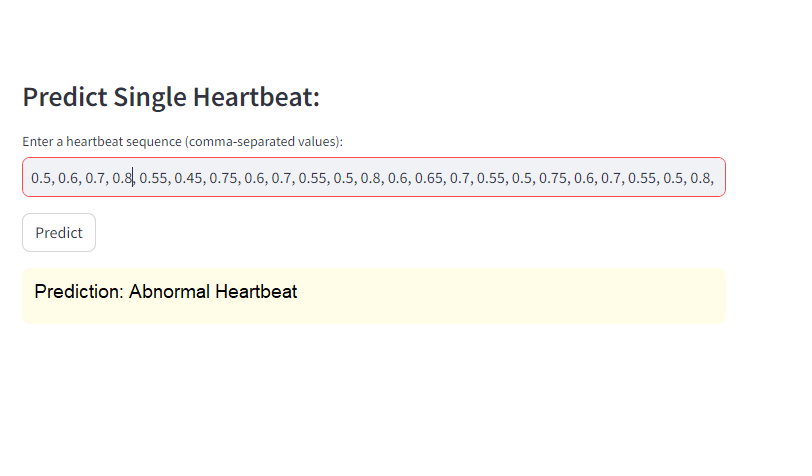


# **Code:**

import numpy as np  
import os  
import pandas as pd  
import tensorflow as tf  
from sklearn.model\_selection import train\_test\_split  
from sklearn.metrics import classification\_report, confusion\_matrix  
import streamlit as st  
import seaborn as sns  
import matplotlib.pyplot as plt  
  
# Disable the PyplotGlobalUseWarning  
st.set\_option('deprecation.showPyplotGlobalUse', False)  
  
# Load the data  
dfs = [pd.read\_csv('F:/7th Semester/Distributed Computing/project/ptbdb\_' + x + '.csv') for x in ['normal', 'abnormal']]  
for df in dfs:  
 df.columns = list(range(len(df.columns)))  
data = pd.concat(dfs, axis=0).sample(frac=1.0, random\_state=1).reset\_index(drop=True)  
data = data.rename({187: 'Label'}, axis=1)  
y = data['Label'].copy()  
X = data.drop('Label', axis=1).copy()  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, train\_size=0.7, random\_state=1)  
  
# Display bar chart of heartbeats distribution before training  
fig\_before\_training, ax\_before\_training = plt.subplots(figsize=(10, 6))  
sns.countplot(x='Label', data=data, hue='Label', palette={0: 'blue', 1: 'orange'})  
plt.title('Heartbeats Distribution Before Training')  
plt.xlabel('Label')  
plt.ylabel('Count')  
plt.legend(title='Label', labels=['Normal', 'Abnormal'])  
st.pyplot(fig\_before\_training)  
  
# Build and train the model  
inputs = tf.keras.Input(shape=(X\_train.shape[1],))  
expand = tf.expand\_dims(inputs, axis=2)  
gru = tf.keras.layers.GRU(256, return\_sequences=True)(expand)  
flatten = tf.keras.layers.Flatten()(gru)  
outputs = tf.keras.layers.Dense(1, activation='sigmoid')(flatten)  
model = tf.keras.Model(inputs=inputs, outputs=outputs)  
model.compile(  
 optimizer='adam',  
 loss='binary\_crossentropy',  
 metrics=[  
 'accuracy',  
 tf.keras.metrics.AUC(name='auc')  
 ]  
)  
  
# history = model.fit(  
# X\_train,  
# y\_train,  
# validation\_split=0.2,  
# batch\_size=32,  
# epochs=100,  
# callbacks=[  
# tf.keras.callbacks.EarlyStopping(  
# monitor='val\_loss',  
# patience=5,  
# restore\_best\_weights=True  
# )  
# ]  
# )  
  
  
# Build and train the model (if not already trained)  
if not os.path.exists('my\_model.h5'):  
 history = model.fit(  
 X\_train,  
 y\_train,  
 validation\_split=0.2,  
 batch\_size=32,  
 epochs=100,  
 callbacks=[  
 tf.keras.callbacks.EarlyStopping(  
 monitor='val\_loss',  
 patience=5,  
 restore\_best\_weights=True  
 )  
 ]  
 )  
 # Save the trained model  
 model.save('my\_model.h5')  
else:  
 # Load the pre-trained model  
 model = tf.keras.models.load\_model('my\_model.h5')  
  
# Streamlit app  
st.title('Heartbeat Classification Demo')  
  
# Sidebar for model information  
st.sidebar.subheader('Model Information:')  
st.sidebar.text(model.summary())  
results = model.evaluate(X\_test, y\_test, verbose=0)  
st.sidebar.text("Test Accuracy: {:.2f}%".format(results[1] \* 100))  
st.sidebar.text(" Test AUC: {:.4f}".format(results[2]))  
  
# Display classification report  
st.subheader('Classification Report:')  
y\_pred = model.predict(X\_test)  
y\_pred\_binary = (y\_pred > 0.5).astype(int)  
st.text(classification\_report(y\_test, y\_pred\_binary))  
  
# Display confusion matrix heatmap  
st.subheader('Confusion Matrix:')  
class\_names = ['Normal', 'Abnormal']  
conf\_matrix = confusion\_matrix(y\_test, y\_pred\_binary)  
# Create a heatmap using seaborn  
fig\_conf\_matrix, ax\_conf\_matrix = plt.subplots()  
sns.heatmap(conf\_matrix, annot=True, fmt="d", cmap="Blues", cbar=False, xticklabels=class\_names, yticklabels=class\_names, ax=ax\_conf\_matrix)  
  
# Add labels and title  
ax\_conf\_matrix.set\_xlabel("Predicted Labels")  
ax\_conf\_matrix.set\_ylabel("True Labels")  
ax\_conf\_matrix.set\_title("Confusion Matrix")  
  
# Show the plot using st.pyplot() by passing the figure explicitly  
st.pyplot(fig\_conf\_matrix)  
  
# Display other relevant information or visualizations  
  
# Bar chart showing the distribution of predicted probabilities  
st.subheader('Distribution of Predicted Probabilities:')  
fig\_proba, ax\_proba = plt.subplots()  
ax\_proba.hist(y\_pred, bins=20, color='skyblue', edgecolor='black', alpha=0.7)  
ax\_proba.set\_xlabel('Predicted Probability')  
ax\_proba.set\_ylabel('Frequency')  
ax\_proba.set\_title('Distribution of Predicted Probabilities')  
st.pyplot(fig\_proba)  
  
# Input for predicting a single heartbeat sequence  
st.subheader('Predict Single Heartbeat:')  
heartbeat\_input = st.text\_input('Enter a heartbeat sequence (comma-separated values):', '0.5,0.6,0.7,0.8')  
heartbeat\_values = [float(value) for value in heartbeat\_input.split(',')]  
  
if st.button('Predict'):  
 # Ensure the input sequence has the correct length (187 in this case)  
 if len(heartbeat\_values) != 187:  
 st.warning('Please enter a sequence of 187 values.')  
 else:  
 # Prepare input for the model  
 heartbeat\_array = np.array(heartbeat\_values).reshape(1, -1, 1)  
  
 # Make prediction  
 prediction = model.predict(heartbeat\_array)  
  
 # Display prediction result  
 if prediction[0, 0] > 0.5:  
 st.success('Prediction: Abnormal Heartbeat')  
 else:  
 st.success('Prediction: Normal Heartbeat')

# **Front end:**





# **Result**

The model demonstrates a robust ability to differentiate between normal and abnormal heartbeats, with high accuracy and strong performance metrics across various evaluation criteria. The following results suggest that the model is well-suited for stress detection based on heartbeat patterns, showcasing its potential for real-world applications.

* Model Accuracy: The developed model achieved a high accuracy of 98.19% on the test set, showcasing its effectiveness in classifying heartbeats as normal or abnormal.
* Area Under the Curve (AUC): The AUC, a measure of the model's ability to distinguish between classes, is reported as 0.9941, indicating excellent performance.
* Classification Report:
  + For normal heartbeats (class 0.0): Precision is 96%, Recall is 95%, and F1-score is 95%.
  + For abnormal heartbeats (class 1.0): Precision is 98%, Recall is 98%, and F1-score is 98%.
* Overall Accuracy: The overall accuracy of the model is reported as 97%, demonstrating its reliability in making accurate predictions across both classes.

# **Conclusion**

In conclusion, the developed deep neural network model has shown exceptional capabilities in the classification of heartbeat signals into normal and abnormal categories. The high accuracy of 98.19% on the test set, coupled with a remarkable AUC of 0.9941, underscores the model's proficiency in distinguishing between different heart conditions. The precision, recall, and F1-score metrics further support the model's effectiveness in capturing both normal and abnormal heartbeat patterns with precision and sensitivity.

The successful outcomes of this project affirm the potential of machine learning, specifically recurrent neural networks, in leveraging heartbeat patterns for stress detection. The utilization of a Gated Recurrent Unit (GRU) architecture, coupled with a carefully curated dataset from MIT-BIH Arrhythmia Dataset and PTB Diagnostic ECG Database, has yielded a reliable model. These findings contribute not only to the field of physiological signal analysis but also hold promise for practical applications in stress management. The model's robust performance suggests its viability for real-time stress detection, providing a non-invasive and automated tool for identifying individuals under stress.

However, it's crucial to acknowledge the limitations and challenges inherent in stress detection, including the subjectivity of stress experiences and ethical considerations. Future work could focus on refining the model's adaptability to diverse stress scenarios, considering long-term trends, and addressing privacy concerns. In essence, this project marks a significant stride towards harnessing machine learning for stress detection through heartbeat patterns, with implications for advancing both physiological signal analysis and the development of practical tools for stress management in real-world scenarios.

# **Acknowledgement**

I would like to express my gratitude to all those who contributed to the success of the project, " **Driving Insights :Stress Detection via Heartbeat Big data Analytics through RNN** " Special thanks to our advisors, Maam Sehrish Aqeel for his unwavering support and guidance. I appreciate the efforts of my project partners, Izza Zafar & Shaher Bano whose dedication was crucial to the project's success.

This project's accomplishment is a result of collective efforts, and I appreciate each individual's unique contributions. Thank you all for being part of this meaningful journey.

Sadia Adrees

University of South Asia

1st Jan 2024

# **References**

<https://towardsdatascience.com/a-brief-introduction-to-recurrent-neural-networks-638f64a61ff4>

<https://mdpi.com/2076-3417/12/3/1409#:~:text=Heart-rate%20variability%20(HRV),ECG)%20signals%20%5B7%5D>.

<https://www.kaggle.com/datasets/shayanfazeli/heartbeat?select=ptbdb_normal.csv>

<https://www.mdpi.com/2076-3417/12/3/1409#:~:text=Heart%2Drate%20variability%20(HRV),ECG)%20signals%20%5B7%5D>.

<https://towardsdatascience.com/how-to-easily-draw-neural-network-architecture-diagrams-a6b6138ed875>