YAŞAR UNIVERSITY

GRADUATE SCHOOL

AIE5509

BIOMEDICAL AND BIOINFORMATIC APPLICATIONS OF ARTIFICIAL INTELLIGENCE

FINAL PROJECT

PROJECT DESCRIPTION

EEG-BASED SCHIZOPRENIA DETECTION WITH ARTIFICIAL INTELLEGENCE

Prepared by: Fuat Deniz SERTKAYA

Instructor: Asst. Prof. Nalan ÖZKURT

Presentation Date: 26.01.2025

I. INTRODUCTION

The goal of this study is to create an artificial intelligence model that can detect schizophrenia from EEG signals. EEG signals are very comprehensive data. EEG signals are crucial as they provide valuable insights into brain activity, enabling behavioral analysis and playing a pivotal role in the detection of various neurological disorders such as epilepsy, schizophrenia, and Alzheimer's disease. Their ability to reflect underlying neural processes makes EEG an essential tool in both clinical diagnosis and research. In patients with schizophrenia, certain abnormalities can be observed in the electrical activity of the brain, in this context there may be irregularities, especially in alpha waves, theta waves and beta waves. In this context, today we will address the problem of detecting schizophrenia from those given EEG signals. This is my first time working with EEG signals. With what I learned from here, especially with what I learned within the scope of feature engineering, I can work with EEG signals more. I can use EEG data to detect different behaviors. I can examine other diseases that can be detected by other EEG signals and classify them, such as ADHD.

In this project report, I explained all the stages of my artificial intelligence model that detects schizophrenia with EEG signals that I have been working on. The dataset I use and the process of processing the signals in the dataset. Then, I trained the signals into artificial intelligence models. For this study, I trained my data with 2 different machine learning models. Then, we analyzed the performance of the models and discussed how to work with EEG signal data in the future, and this is how the report is completed.

II. DATASET

The Task I have determined is an artificial intelligence model that will classify schizophrenia (binary classification) through EEG signals. We will need recordings of EEG signals of both schizophrenic and non-schizophrenic people in the dataset. We can create this ourselves, with the necessary equipment and participants. In our first approach to this problem, we solved the dataset problem with an open-source dataset approach on the internet.

2.1. About

EEG in Schizoprenia belongs to IBIB PAN Department of Methods of Brain Imaging and Functional Research of Nervous System. The dataset comprised 14 patients with paranoid schizophrenia and 14 healthy controls. Data were acquired with the sampling frequency of 250 Hz using the standard 10-20 EEG montage with 19 EEG channels: Fp1, Fp2, F7, F3, Fz, F4, F8, T3, C3, Cz, C4, T4, T5, P3, Pz, P4, T6, O1, O2. The reference electrode was placed between electrodes Fz and Cz (IBIB PAN - Department of Methods of Brain Imaging and Functional Research of Nervous System, 2017).

There are 28 “.edf” extension files in the data set. The 14 files starting with the letter ‘h’, from h01 to h14, are the recorded EEG signals from the 14 healthy people. There are 14 files starting with the letter 's', from s01 to s14, and these are the recorded EEG signals from the people with schizophrenia. Files with the European Data Format (.edf) extension are a high-standard file format widely used for recording and storing biomedical signals (such as EEG, ECG, EMG).

2.2. Approaches for Signal Classification

1. **Signal Based Approach**

   Numerical features extracted in the time domain on the raw signal data are used as input data of the model and trained in this way (numeric based).

2. **Image Based Approach**

   After applying some feature extraction to the signals, we can visualize the signal using a visualization parameter such as a spectrogram and perform model training on this visual data. Olejarczyk and Jernajczyk observed that there are noticeable differences in the graphical representation of EEG signals between schizophrenic patients and healthy individuals, particularly in the alpha waves (Olejarczyk & Jernajczyk, 2017).

In this large-scale project, we will apply the signal based approach for this study report.

2.3. Installation and Read the Datas on Python

My project development environment will be Google Colab and we will use the Python programming language. I followed the steps below to use EEG signals correctly.

1. **Activate Cloud:** I uploaded the currently recorded data set to the cloud (Google Drive). Then, I connected to the cloud via Google colab and pulled the .zip file containing the dental images from the cloud.

2. **Extract the Data Files:** EEG signals can be stored in edf format. In order to capture EEG signals, it is necessary to first decode the edf' format files. Thanks to the 'mne' library in Python, we can extract signals from files with edf extension. Magnetoencephalography (MNE) is specifically designed for storing and processing EEG, MEG and other neurophysiological data.

2.4. Data Preprocessing

The following data preprocessing applications were used to make the data suitable for training the model.

- **Preparing Signal Reference:** The EEG tuning process makes raw data suitable for modeling. Reference adjustment organizes signals according to a common reference, making the signal more meaningful. In this part of the project the reference value can be customized (it can only be set to 'Cz' or 'Fz'). However, we used automatic or default reference at this stage.

- **Band-Pass Filtering:** Filtering is done to purify the signal from noise at low and high frequencies. This process focuses on a specific frequency band of EEG data, using only the information or frequency waves that the model needs. The signal frequency range we use is between 0.5 Hz and 45 Hz. The signal frequency range we use is between 0.5 Hz and 45 Hz. The names of the waves we need here are as follows:

    1. **Delta:** (0.5-4 Hz)
    2. **Theta:** (4-8 Hz)
    3. **Alpha:** (8-13 Hz)
    4. **Beta:** (13-30 Hz)
    5. **Gamma:** (30> Hz).

- **Data Labeling:** In order to perform binary classification of the data set to be used in training the model, it is necessary to label the data as 0 and 1. The signal data of healthy people in the dataset is labeled as '0', and the signals of schizophrenia patients are labeled as '1'.

- **Splitting Signals:** Within the scope of time series, I divided the signals into 5-second epochs. Entered the following line of code using the MNE library function in Python:

```
epochs=mne.make_fixed_length_epochs(data, duration=5, overlap=1)
```

Re-sizing signals with the ‘make_fixed_length()’ function. The parameters are 'data', our current dataset, 'duration', how many seconds an epoch will be, and the 'overlap' parameter, which determines how much overlap there is between the two data.

- **Stacking:** Merging the data into a single large numpy array is done to make the data available for training and testing phases. It also matches tags and group information.

- **Statistical Feature Extraction:**

We enrich the representation of data by extracting features of EEG signals. In the feature engineering phase, a set of statistical features that carry important information in the time domain are calculated.

In this context, for each signal: mean, standard deviation (std), min (minim), max (maxim), variance (var), and peak (ptp), root mean square (rms), sum of absolute differences (abs_diff_signal), skewness, kurtosis, minimum (argminim) and maximum (argmaxim) values of the signal were calculated.

These features provide meaningful data that can be used to improve the accuracy of the classification model.

EEG signals are subjected to a lot of pre-processing. Implementing the data preprocessing phase correctly and setting the correct parameters is essential for model training.

III. METHODOLOGY

3.1. General Approach

The first step in machine learning based artificial intelligence projects is to identify the problem. Our problem is to create an artificial intelligence model that will detect schizophrenia. After determining the problem, it is necessary to create a dataset. The dataset we consider in our problem is EEG signals. After finding the Data Set, it is necessary to pre-process the EEG signals. The signal preprocessing steps in part 2 of the report are very effective practices. After completing the preprocessing phase of the signal data, we will move on to the model creation phase to train the dataset. We discussed the data preprocessing part in the second part of the report. In the rest of the report, we will develop 2 different machine learning algorithms to train the dataset and try to optimize their parameters. And finally, we will examine the binary classification ability of the model.

The flow chart below represents the summary of the solution to the problem of detecting schizophrenia from EEG signals:

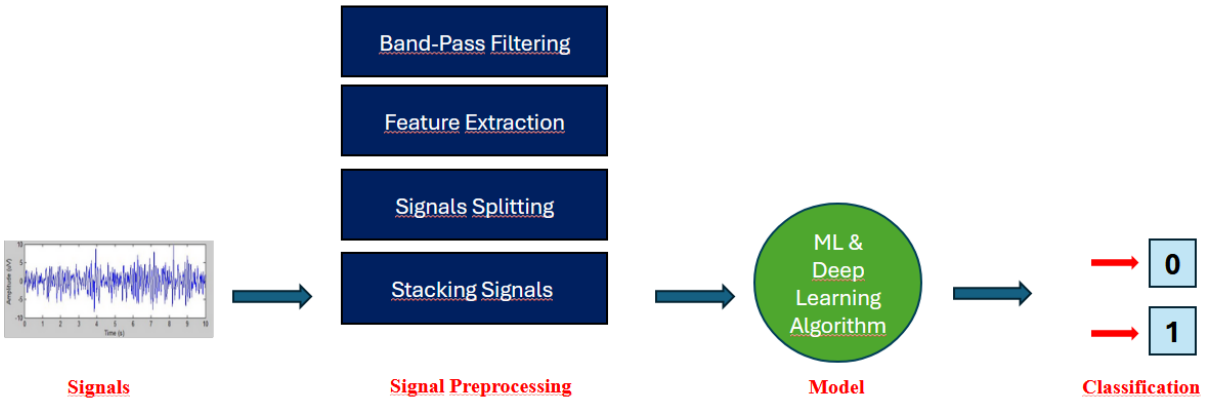


Fig.1. Flowchart that represents the methodology of EEG based schizoprenia detection.

3.2. Artificial Intelligence Model

3.2.1. Logistic Regression

One of the solution methods we use to address the problem of schizophrenia classification from EEG signals is logistic regression. Logistic regression is one of the classification algorithms of supervised learning. We used the **StandardScaler()** function to standardize the signal data for our logistic regression model. The standard scaling method, when used outside of deep learning, positively impacts the performance of machine learning models.

Cross validation is a statistical resampling technique used to improve the performance of the model. We apply **5-fold cross validation** for hyperparameter optimization when training the data with logistic regression. The 'C' hyperparameter in logistic regression controls the level of regularization of the model and strikes a balance between the complexity of the model and its generalization power.

The 5 different C hyperparameters we analyzed are: {0.001,0.01,0.1,1,10,100}. In our logistic regression EEG signal schizophrenia detection model, the C hyperparameter that showed the best performance out of 5 values was {C=100}. And the best score value is {Best_Score ≈ 67%}.

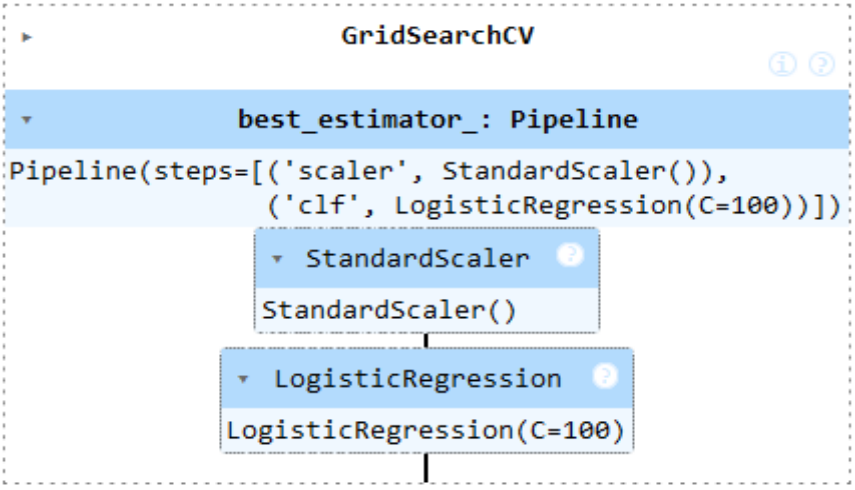


Fig.2. Architecture of the logistic regression model.

3.2.2. Deep Learning

Aims to improve accuracy performance, a deep learning model was developed for schizophrenia classification. The model is designed to learn the properties of time series signals, and in this context, we trained EEG signals containing time series with a recursive neural network (RNN) deep learning algorithm. While designing this architecture, we were inspired by the ChronoNet model, which showed very successful results in the EEG signal processing task. ChronoNet is an architecture that includes long term memory (LSTM) and convolutional neural networks (CNN). We have stated below the network architectures we use for the EEG signal. some adaptations have been made inspired by the ChronoNet architecture.

Deep Recurrent Neural Networks (DRNN), text, speech, time series, etc. It is a type of machine learning algorithm specifically designed to analyze and understand data sets such as While input and output are independent from each other in classical deep learning, on the contrary, since the RNN structure can remember past information, it gives more successful results than classical deep learning models in time series data (Akköse, 2020).

- ChronoNet

ChronoNet is a deep closed RNN architecture developed by combining densely connected recurrent layers with inception layers with exponentially varying kernel lengths in 1D convolution layers. This architecture can perform classification by working directly on the raw EEG time series signal without the need to extract manually engineered features. As a result, ChronoNet achieved 7.77% (Roy, et al., 2019) higher accuracy than traditional methods, creating a new benchmark in the EEG classification task. I used the deep learning model I built for the schizophrenia detection problem by examining the ChronoNet model, that is, a deep learning model inspired by ChronoNet.

- Long Short Term Memory (LSTM)

LSTM layer is used in ChronoNet architecture. LSTM is a member of the RNN family and is a very effective model, especially for learning long-term dependencies in time series data. They can be very useful in EEG signals because they are a model that is robust to the problem of vanishing gradients in long-term time series.

- Convolutional Neural Network (CNN)

CNN layer is used in ChronoNet architecture. The 1D convolutional and Max Pooling layers we use in our model architecture are part of convolutional neural networks. Convolutional layers are very powerful layers for feature extraction. They will also be effective during signal processing.

You can see the image of the Deep learning architecture in the figure below.

Layer (type)	Output Shape	Param #
conv1d_14 (Conv1D)	(None, 226, 64)	256
max_pooling1d_14 (MaxPooling1D)	(None, 113, 64)	0
dropout_35 (Dropout)	(None, 113, 64)	0
conv1d_15 (Conv1D)	(None, 111, 128)	24,704
max_pooling1d_15 (MaxPooling1D)	(None, 55, 128)	0
dropout_36 (Dropout)	(None, 55, 128)	0
lstm_14 (LSTM)	(None, 55, 100)	91,600
dropout_37 (Dropout)	(None, 55, 100)	0
lstm_15 (LSTM)	(None, 100)	80,400
dropout_38 (Dropout)	(None, 100)	0
dense_14 (Dense)	(None, 64)	6,464
dropout_39 (Dropout)	(None, 64)	0
dense_15 (Dense)	(None, 1)	65

Fig.3. The deep neural network architecture model for EEG Based schizoprenia detectio

Now let's analyze layers, parameters and hyperparameters:

Layers with Parameters & Hyperparameters:

- **Conv1D:** In the first Conv1D layer, a layer with 64 filters and 3 kernel size (64.3) was entered, and in the second Conv1D layer (128,3).
- **MaxPooling1D:** It comes after Conv1 layers. Used twice in architecture. 2x2 pooling was applied
- **Dropout:** It was used 5 times and placed at the end of each feature learning layer. (0.2 for Conv1D layers, 0.3 for LSTM layers and 0.4 for the final Dense layer)
- **LSTM:** The main learning layer after the Conv1D layer 2 times. 100 neurons were choosed in both LSTM layers.
- **Dense:** Dense layer with 64 neurons and last dense layer is output layer

Other Parameters & Hyperparameters:

- **Batch Size:** 64
- **Epoch:** 30
- **Activation Function:** ReLu
- **Output Activation Function:** Sigmoid
- **Loss & Optimizer:** Binary crossentropy, Adam

IV. RESULTS

The performance comparison of the two models Logistic Regression vs Deep Learning is given below:

Model / Perf.Metrics	Logistic Regression Model	Deep Learning Model
Train Accuracy	0.9104	0.9446
Test Accuracy	0.8910	0.9404
Memory Usage	11.61 MB	4633.07 MB
Total Parameters	-	610469

Table 1. The logistic regression and the deep learning models comparison according to performance metrics

I made changes to see if I could increase the accuracy percentage of the deep learning model. Changes I made from the first deep learning model: batch_size=16, I increased the two dropout in Conv1D layers to 0.25.

Model / Perf.Metrics	Deep Learnin Model (Updated Paremeters)	Deep Learning Model
Train Accuracy	0.9572	0.9446
Test Accuracy	0.9500	0.9404
Memory Usage	4575.78 MB	4633.07 MB
Total Parameters	610469	610469

Table 2. Two deep learning models comparison according to performance metrics

V. DISCUSSION

5.1 Models

I) Logistic Regression

Train accuracy ≈ 91 and test accuracy ≈ 89. test accuracy is actually in a pretty good position. We can say that logistic regression shows a good performance value for the binary classification task of schizophrenia detection. Train accuracy is a little more than 91, there is some overfitting here. One of the reasons affecting this is, of course, the C hyperparameter value. The C hyperparameter value that gave the best results in our model was the highest value of 100. As the C parameter increases, the learning capacity increases, but the regularization ability begins to decrease, which causes overfitting.

II) Deep Learning Models

I used a Chrononet-based architecture as a deep learning model. This method was chosen due to its success in EEG signals. With a few parameters and hyperparameter settings, training accuracy ≈ 94, test accuracy ≈ 94. I can say that the test score is quite good, and overfitting problems were not seen much. Epoch=30 is the most optimized. When I reduce it to 20, the training ends before I can reach my best performance. When it starts to go above 30, it becomes overfitting. By playing with batch size, the model got the value of test accuracy ≈ 95 and test accuracy ≈ 95. 95 was the highest value I got, but there was also a small overfitting in the model.

5.2 Logistic Regression vs Deep Learning

As expected, we achieved a better accuracy score with deep learning models. However, deep learning models consume around 4600 MB, while the logistic regression algorithm only uses 11.6 MB. There is a significant difference in memory usage here. Logistic regression is a simple and cost-effective algorithm. Deep learning, by its nature, is much more complex and costly to implement. If you have no issues with memory and are focused on improving performance, a deep learning model can be chosen. However, for a model that is both good in performance and optimized in terms of cost, logistic regression is a suitable choice.

VI. CONCLUSION

We In this study, the problem of an artificial intelligence model that can detect schizophrenia from EEG signals was discussed. Our dataset consisted of EEG signals from 14 healthy and 14 schizophrenia patients. We performed some operations on these signals, and these signal processing steps were one of the factors that greatly increased the performance of our model. We considered 2 different approaches to train the model. One of the traditional machine learning methods is logistic regression. The other is a deep learning model based on ChronoNet. Although the deep learning model gave good results as expected, the accuracy rates of both models were quite good. This was achieved thanks to finding a quality dataset and applying correct feature engineering to the EEG data.

A self-criticism would be that this study does not include a visual analysis on EEG signals. In my future work, we can consider the same dataset and analyze the graphic images of the signals this time. Maybe we can then create a new dataset from the signal images and discuss the image classification model.

This was my first time working with EEG signals and I will be able to work more due to the scope of EEG signals. Because EEG signals are very important data in detecting brain activities and diseases. As a result, today I detected schizophrenia through EEG signals, but what can be done with EEG signals is much greater. Learning the preprocessing steps of EEG signals here gave me experience and thanks to this, I will be able to be more efficient in future EEG signals studies.

VII. REFERENCES

[1] IBIB PAN - Department of Methods of Brain Imaging and Functional Research of Nervous System. (2017). *EEG in schizophrenia* [Dataset]. RE-POD Repository. https://doi.org/10.18150/repod.0107441.

[2] Olejarczyk, E., & Jernajczyk, W. (2017). Graph-based analysis of brain connectivity in schizophrenia. *PloS one*, *12*(11), e0188629.

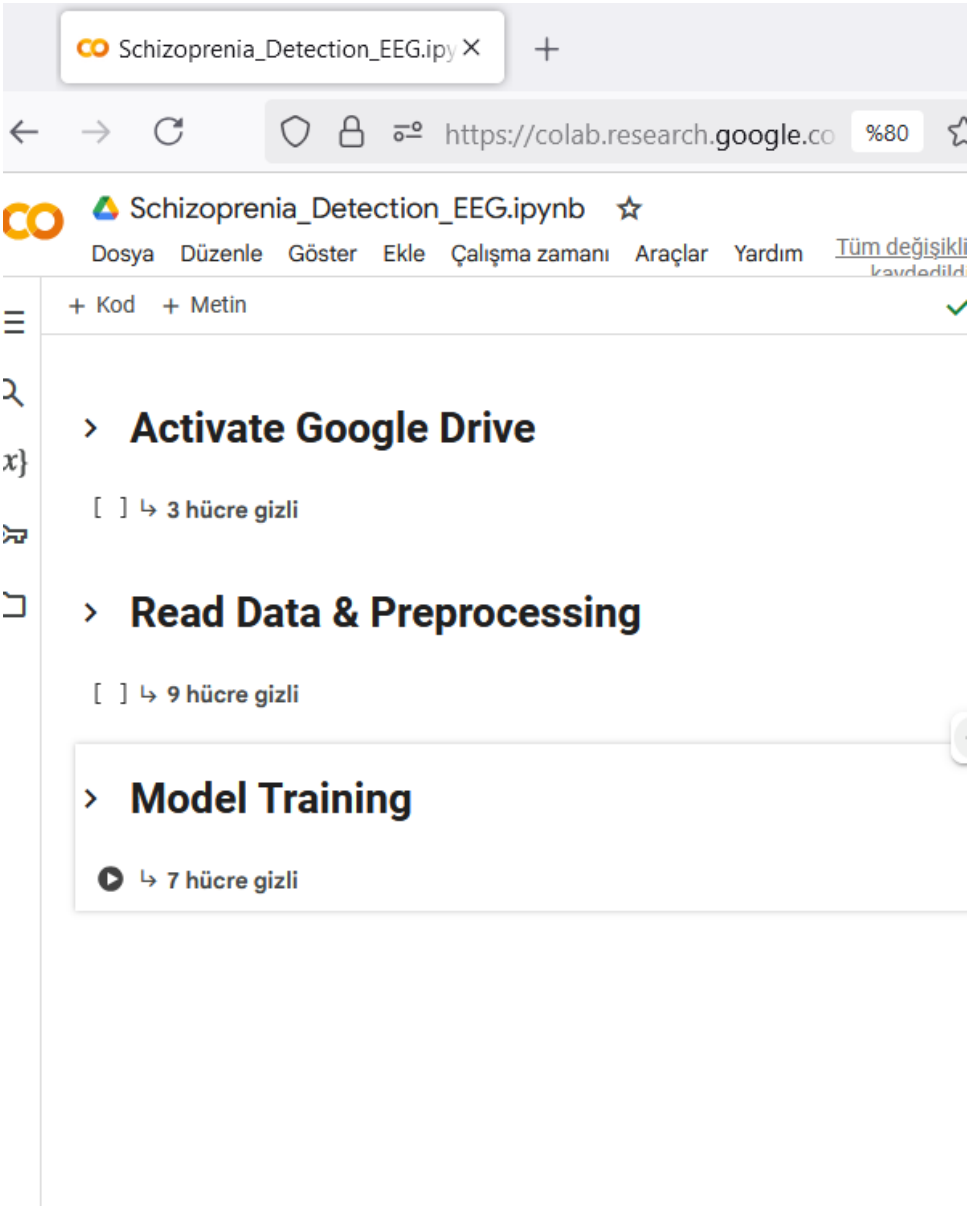
[3] Akköse, O. (2020, 22 Aralık). *Uzun-Kısa Vadeli Bellek (LSTM)*. Medium. https://medium.com/deep-learning-turkiye/uzun-k%C4%B1sa-vadeli-bellek-lstm-b018c07174a3.

[4] Roy, S., Kiral-Kornek, I., & Harrer, S. (2019). ChronoNet: A deep recurrent neural network for abnormal EEG identification. In *Artificial Intelligence in Medicine: 17th Conference on Artificial Intelligence in Medicine, AIME 2019, Poznan, Poland, June 26–29, 2019, Proceedings 17* (pp. 47-56). Springer International Publishing.

APPENDIX

Appendix A

A-1: Main Parts of Code



Appendix B

B-1: Data Reading & Preprocessing

```
[13] from glob import glob
     import os
     import mne
     import numpy as np
     import pandas
     import matplotlib.pyplot as plt
```

```
[14] dataset=glob('schizo_dataset/*.edf')
     print(len(dataset))
```

```
28
```

```
[15] healty=[i for i in dataset if 'h' in i.split('/')[1]]
     sick=[i for i in dataset if 's' in i.split('/')[1]]
```

```
[16] def read_data(file_path):
       data=mne.io.read_raw_edf(file_path,preload=True)
       data.set_eeg_reference()# reference to
       data.filter(l_freq=0.5,h_freq=45)#bandpass filter
       epochs=mne.make_fixed_length_epochs(data,duration=5,overlap=1)#split signals 5 second epochs
       array=epochs.get_data()
       return array
```

```
%%capture
healty_data=[read_data(i) for i in healty]
sick_data=[read_data(i) for i in sick]
```

```
[18] #Create Label
     hlabel=[len(i)*[0] for i in healty_data]
     slabel=[len(i)*[1] for i in sick_data]

     data_list=healty_data+sick_data
     label_list=hlabel+slabel

     grouplist=[[i]*len(j) for i,j in enumerate(data_list)]

     data_array=np.vstack(data_list)
     label_array=np.hstack(label_list)
     group_array=np.hstack(grouplist)
```

B-2: Data Reading & Preprocessing Cont.

```
[20] from scipy import stats
     def mean(x):
       return np.mean(x, axis=-1)
     def std(x):
       return np.std(x, axis=-1)
     def ptp(x):
       return np.ptp(x, axis=-1)
     def var(x):
       return np.var(x,axis=-1)
     def minim(x):
       return np.min(x,axis=-1)
     def maxim(x):
       return np.max(x,axis=-1)
     def argminim(x):
       return np.argmin(x,axis=-1)
     def argmaxim(x):
       return np.argmax(x,axis=-1)
     def rms(x):
       return np.sqrt(np.mean(x**2,axis=-1))
     def abs_diff_signal(x):
       return np.sum(np.abs(np.diff(x,axis=-1)),axis=-1)
     def skewness(x):
       return stats.skew(x,axis=-1)
     def kurtosis(x):
       return stats.kurtosis(x, axis=-1)
     def concatenate_features(x):
       return np.concatenate((mean(x), std(x), ptp(x), var(x), minim(x), maxim(x), argminim(x),
                              argmaxim(x), rms(x), abs_diff_signal(x),skewness(x),kurtosis(x)), axis=-1 )
```

```
     features=[]
     for d in data_array:
       features.append(concatenate_features(d))

     features=np.array(features)
```

B-3: Build the Model – Logistic Regression

Logistic Regression

```
[ ] from sklearn.model_selection import train_test_split
    from sklearn.pipeline import Pipeline
    from sklearn.preprocessing import StandardScaler
    from sklearn.linear_model import LogisticRegression
    from sklearn.model_selection import GroupKFold, GridSearchCV
    from sklearn.metrics import accuracy_score
    import tracemalloc
    import psutil
```

```
X_train, X_test, y_train, y_test, groups_train, groups_test = train_test_split(
    features, label_array, group_array, test_size=0.2, random_state=42, stratify=label_array
)


clf = LogisticRegression()
pipe = Pipeline([('scaler', StandardScaler()), ('clf', clf)])
param_grid = {'clf__C': [0.001, 0.01, 0.1, 1, 10, 100]}
gkf = GroupKFold(5)
gscv = GridSearchCV(pipe, param_grid, cv=gkf, scoring='accuracy', n_jobs=-1)
tracemalloc.start()
gscv.fit(X_train, y_train, groups=groups_train)




current, peak = tracemalloc.get_traced_memory()
print(f"Current memory usage: {current / 1024**2:.2f} MB")
print(f"Peak memory usage: {peak / 1024**2:.2f} MB")
memory_usage = psutil.Process().memory_info().rss / 1024**2
print(f"Memory Usage (psutil): {memory_usage:.2f} MB")

# Eğitim ve test setlerinde doğruluk hesaplama
y_train_pred = gscv.predict(X_train)
y_test_pred = gscv.predict(X_test)

train_accuracy = accuracy_score(y_train, y_train_pred)
test_accuracy = accuracy_score(y_test, y_test_pred)

print(f"Train Accuracy: {train_accuracy:.4f}")
print(f"Test Accuracy: {test_accuracy:.4f}")
```

B-4: Build the Model – ChroneNet Based Deep Learning Model

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv1D, MaxPooling1D, LSTM, Dense, Flatten, Dropout
from tensorflow.keras.optimizers import Adam
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
import numpy as np
import time

def chrono_net(input_shape):
    model = Sequential()

    model.add(Conv1D(64, 3, activation='relu', input_shape=input_shape))
    model.add(MaxPooling1D(pool_size=2))
    model.add(Dropout(0.25))

    model.add(Conv1D(128, 3, activation='relu'))
    model.add(MaxPooling1D(pool_size=2))
    model.add(Dropout(0.25))

    model.add(LSTM(100, return_sequences=True))
    model.add(Dropout(0.3))

    model.add(LSTM(100, return_sequences=False))
    model.add(Dropout(0.3))


    model.add(Dense(64, activation='relu'))
    model.add(Dropout(0.4))
    model.add(Dense(1, activation='sigmoid'))

    model.compile(optimizer=Adam(), loss='binary_crossentropy', metrics=['accuracy'])

    return model


X = features
y = label_array

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
X_train = X_train.reshape(X_train.shape[0], X_train.shape[1], 1)
X_test = X_test.reshape(X_test.shape[0], X_test.shape[1], 1)

model = chrono_net(X_train.shape[1:])

history = model.fit(X_train, y_train, epochs=30, batch_size=16, validation_split=0.2, verbose=1)


train_accuracy = history.history['accuracy'][-1]
val_accuracy = history.history['val_accuracy'][-1]

print(f'Train Accuracy: {train_accuracy:.4f}')
test_loss, test_acc = model.evaluate(X_test, y_test, verbose=1)
print(f'Test Accuracy: {test_acc:.4f}')
```