

Sleep State Detection Using LSTM

A Minor Project Report

*Submitted to the Central University of Haryana
in partial fulfillment of requirements for the award of degree*

Bachelor of Technology

in

Computer Science and Engineering

by

Satyam Arya (202118)

Yash Raj (202134)

Under the supervision of

Dr. Rakesh Kumar

Associate prof. & HOD

Dept. of Computer Science & Engineering

School of Engineering & Technology

Central University of Haryana



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SCHOOL OF ENGINEERING & TECHNOLOGY

CENTRAL UNIVERSITY OF HARYANA, MAHENDRAGARH

HARYANA - 123029, INDIA

December 2023

DECLARATION

We hereby declare that the seminar report **Sleep State Detection Using LSTM** by us Satyam Arya (202118) and Yash Raj (202134), is submitted for partial fulfillment of the requirements for the award of degree of Bachelor of Technology of the Central University, Haryana Submitted in the department of Computer Science and Engineering and is a bonafide work done by us under the supervision of Dr. Rakesh Kumar. This submission represents our ideas in our own words and where ideas or words of others have been included, we have adequately and accurately cited and referenced the original sources.

Satyam Arya

Yash Raj

This is to certify that the above statement made by the candidates is correct to the best of our knowledge.

Dr. Rakesh Kumar

(Head of Dept.)

Department of Computer Science & Engineering.

Central University of Haryana

Countersigned By :

The B.Tech Project Viva-voce examination of Satyam Arya (202134) and Yash Raj (202134) has been held on and is accepted.

Dr. Rakesh Kumar

(Supervisor)

Acknowledgement

We take this opportunity to express our deepest sense of gratitude and sincere thanks to everyone who contributed to the successful completion of this work. The heartfelt appreciation goes to Dr. Rakesh kumar, the Head of the Department (HOD) of Computer Science and Engineering at Central University of Haryana, Mahendergarh, for their unwavering support and provision of all necessary facilities, which were instrumental in the realization of this project, done under their guidance.

We would also like to extend our thanks to the entire faculty of the Computer Science and Engineering department for their constant encouragement, valuable insights, and academic guidance. The collective efforts of the teachers have significantly shaped our understanding and enriched our learning experience.

Furthermore, we express gratitude to ourselves for the dedication and hard work invested in the project. Additionally, we extend our thanks to all our friends who, through their contributions, played a crucial role in the successful fulfillment of this project work.

Satyam Arya (202118)

Yash Raj (202134)

Project Distribution

Name of Team Member	Roll Number	Project Distribution
Satyam Arya	202118	Research, Analysis, and Documentation
Yash Raj	202134	Model Building and Testing

Abstract

The "Detect sleep states" competition, hosted by the Child Mind Institute, presents a groundbreaking initiative to advance sleep research through data science. Focused on developing models to accurately detect sleep onset and wake phases from wrist-worn accelerometer data, this competition addresses the limitations of traditional sleep monitoring methods. Successful outcomes promise to enhance researchers' capabilities for large-scale sleep studies, ultimately improving our understanding of sleep's importance. With implications extending to children's mental health, participants have the unique opportunity to contribute to the development of tools that inform personalized interventions and treatment strategies. Hosted by CMI, a leading nonprofit, the competition aligns with global mental health goals, emphasizing technology's role in augmenting mental health care and research. Adhering to specific timelines and code requirements, participants aim to make meaningful contributions to the well-being of children and families worldwide.

Contents

Acknowledgement	i
Abstract	iii
1 Introduction	1
1.1 About the Organisation	2
1.2 Recent hosted Competitions	3
2 Objective	4
3 Methodology	5
3.1 Long Short-Term Memory (LSTM)	5
3.1.1 Introduction	5
3.1.2 Key Features	5
3.1.3 Basic Workflow	6
3.1.4 Applications of LSTM	8
4 Work Done	9
4.1 About Dataset	9
4.1.1 Files and Field Descriptions	10
4.2 System Requirements	11
4.3 Tools and technologies Used	12
4.4 Notebook Structure	13
4.5 Import necessary Libraries and packages	14
4.6 Exploratory Data Analysis (EDA)	15
4.6.1 Data Overview	15

4.6.2	Data Quality and Missing Values	16
4.6.3	Temporal Patterns and Sleep Windows	16
4.6.4	Correlation Analysis	16
4.6.5	Visualization	16
4.6.6	Summary Statistics	16
4.6.7	Insights and Observations	17
4.6.8	Visualizations	18
4.7	Data Preprocessing and Model training	21
4.7.1	Data Preprocessing	21
4.7.2	Model Training	23
4.8	Model Evaluation	26
4.8.1	Training Loss	26
4.8.2	Model Predictions vs. True Labels	26
4.8.3	Event Detection Analysis	27
4.8.4	Scoring Metric	27
4.8.5	Overall Model Evaluation	28
4.9	Final Outcome	28
4.9.1	Result and submission	29
5	Challenges	31
5.1	Challenges During Project completion	32
6	Achievements	34
7	Conclusion	35
8	Future Scope	37
	References	39

Chapter 1

Introduction

Sleep is a fundamental aspect of human health, influencing various facets such as cognitive functioning, mood regulation, and behavioral patterns. The Child Mind Institute (CMI) is hosting a groundbreaking competition - "Detect Sleep States" - aimed at advancing the analysis of accelerometer data gathered from wrist-worn devices. The primary objective is to develop a model that accurately detects sleep onset and wake phases, paving the way for more reliable and extensive sleep studies. Researchers, particularly those focused on children's mental health, stand to benefit significantly from the outcomes of this competition. By harnessing the power of data science, participants in this competition have a unique opportunity to contribute to the development of tools that not only improve our understanding of sleep but also pave the way for personalized interventions and support systems tailored to the distinct needs of each child. The competition's alignment with the mission of the Child Mind Institute and the broader goals of the Stavros Niarchos Foundation Global Center for Child and Adolescent Mental Health underscores its commitment to driving innovation in the field, ultimately benefiting mental health on a global scale.

1.1 About the Organisation



Figure 1.1: Child Mind Institute (<https://childmind.org/>)

The Child Mind Institute, under the leadership of Dr. Harold Koplewicz and Dr. Leonard Abbeduto, stands as a beacon of hope for children and families navigating the complexities of mental health and learning challenges. Committed to empowerment and positive change, the Institute spearheads impactful initiatives encompassing clinical care, educational resources, cutting-edge research, and community engagement. Notable among these initiatives are competitions such as the one focused on detecting sleep states, the annual Rising Scientist Awards recognizing high school students in mental health research, and the prestigious Annual Change Maker Awards honoring individuals and organizations making lasting positive changes in mental health care. This project report delves into these endeavors, highlighting their unique contributions and showcasing the broader impact on children, families, and the community. By exploring the Institute's dedication to innovation, collaboration, and empowerment, we gain a deeper understanding of its vital role in shaping a more compassionate and supportive world for those facing mental health challenges.

1.2 Recent hosted Competitions

1. Detect Sleep States (2023):

This competition challenged researchers to develop a model capable of accurately detecting sleep onset and wakefulness from data collected by wrist-worn accelerometers. The winning model could potentially revolutionize sleep monitoring and provide valuable insights into sleep health for children and adults alike.

2. Rising Scientist Awards (2023):

Recognizing the crucial role of young minds in shaping the future of mental health research, the Child Mind Institute established the Rising Scientist Awards. This annual award honors high school students who demonstrate exceptional promise and dedication in their research endeavors. By supporting and nurturing their talent, the Institute aims to cultivate the next generation of leaders in this critical field.

3. Annual Change Maker Awards:

This prestigious award celebrates individuals and organizations who have made significant and lasting contributions to positive change within the mental health care system. The Change Maker Awards recognize those who champion groundbreaking initiatives, forge innovative partnerships, and ultimately pave the way for a brighter future for mental health care for all.

The project report explores impactful initiatives of the Child Mind Institute, ranging from a competition on sleep data analysis to awards recognizing contributions in mental health research and care. By highlighting the unique contributions of these endeavors, the report underscores their broader impact on children, families, and the community.

Chapter 2

Objective

The "Detect Sleep States" competition, hosted by the Child Mind Institute (CMI), aims to advance sleep research through data science. The primary objective is to develop a model for accurately detecting sleep onset and wake phases from wrist-worn accelerometer data. By improving study reliability, the competition enables larger-scale investigations, offering insights into the impact of environmental factors on sleep, mood, and behavior. Addressing challenges in traditional methods, the competition fosters innovation. It aligns with CMI's mission, benefiting children with mood and behavior difficulties. The outcomes include advancements in sleep research, practical impact on mental health studies, and applicable tools for clinicians and researchers.

The goal of this project is to detect sleep onset and wake. We developed a model trained on wrist-worn accelerometer data in order to determine a person's sleep state. Our work could make it possible for researchers to conduct more reliable, larger-scale sleep studies across a range of populations and contexts. The results of such studies could provide even more information about sleep.

The successful outcome of this project can also have significant implications for children and youth, especially those with mood and behavior difficulties. Sleep is crucial in regulating mood, emotions, and behavior in individuals of all ages, particularly children. By accurately detecting periods of sleep and wakefulness from wrist-worn accelerometer data, researchers can gain a deeper understanding of sleep patterns and better understand disturbances in children.

Chapter 3

Methodology

3.1 Long Short-Term Memory (LSTM)

3.1.1 Introduction

Long Short-Term Memory (LSTM) is a specialized type of recurrent neural network (RNN) architecture designed to overcome challenges associated with learning long-term dependencies in sequential data. Proposed by Sepp Hochreiter and Jürgen Schmidhuber in 1997, LSTMs have become a fundamental building block in various applications, particularly in natural language processing, speech recognition, and time series prediction.

3.1.2 Key Features

LSTMs address the vanishing gradient problem by introducing key architectural components:

Memory Cells: LSTMs incorporate memory cells that allow the network to selectively store and retrieve information for extended periods. This helps in capturing long-term dependencies in sequential data.

Gates:

- – **Forget Gate:** Determines what information from the cell state should be discarded.

- **Input Gate:** Regulates the introduction of new information into the cell state.
- **Output Gate:** Controls the information output to the next layer in the network.
- **Cell State:** LSTMs maintain a cell state that runs through the entire sequence, facilitating the flow of information across various time steps.

How LSTM is useful?

Long Short-Term Memory (LSTM) networks are instrumental in capturing and learning complex patterns in sequential data, making them particularly valuable in tasks requiring the understanding of long-term dependencies. Unlike traditional recurrent neural networks, LSTMs address the vanishing gradient problem, allowing them to effectively retain and utilize information over extended sequences. This capability proves crucial in various applications such as natural language processing, speech recognition, and time series prediction, where the intricate relationships between data points over time significantly impact the overall understanding and performance of the model. LSTMs, with their memory cells and gated architecture, excel in capturing and retaining contextual information, making them a fundamental building block in the development of advanced machine learning models that deal with sequential data.

3.1.3 Basic Workflow

1. **Input Sequence:** Receive input sequence data, such as time series or natural language text.
2. **Sequential Processing:** Process the input sequence step by step, considering temporal dependencies.
3. **Memory Cells:** Utilize memory cells to store and maintain information over extended periods.
4. **Gates:** Employ gates (Forget, Input, Output) to control the flow of information within the memory cell.

- **Forget Gate:** Determines what information to discard from the cell state.
 - **Input Gate:** Regulates the introduction of new information into the cell state.
 - **Output Gate:** Controls the information output to the next layer.
5. **Learning:** Update parameters using backpropagation during training to learn optimal values for memory cells and gates.
 6. **Output Prediction:** Make predictions based on learned dependencies within the sequential data.

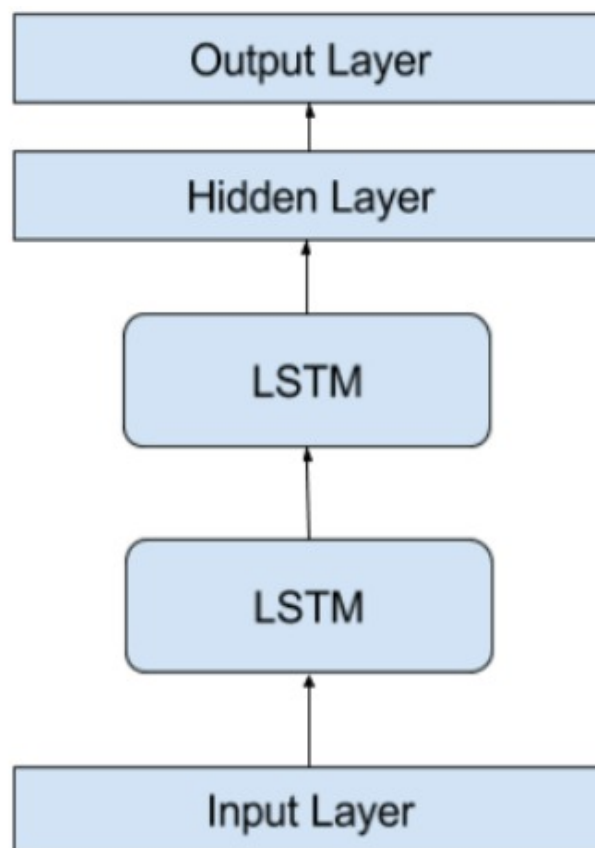


Figure 3.1: LSTM workflow (https://www.researchgate.net/figure/Proposed-LSTM-Network-Layers-LSTM-Network-model-has-been-processed-by-implementing-Keras_fig2_317391149)

3.1.4 Applications of LSTM

- **Time Series Prediction:** Forecasting in time series data like stock prices, weather patterns, and energy consumption.
- **Natural Language Processing (NLP):** Language translation, sentiment analysis, and text generation due to the ability to capture contextual information.
- **Speech Recognition:** Applied in systems where temporal dependencies in audio signals are crucial for accurate recognition.
- **Financial Modeling:** Predicting financial market trends and stock prices by analyzing historical market data.
- **Healthcare:** Analyzing patient data for predicting disease progression or monitoring vital signs over time.
- **Robotics and Autonomous Vehicles:** Contributing to decision-making processes by understanding and predicting sequences of actions.
- **Gesture Recognition:** Recognizing patterns in sequential data for gesture recognition applications.
- **Anomaly Detection:** Identifying abnormal patterns in data, such as detecting fraud or network intrusion.

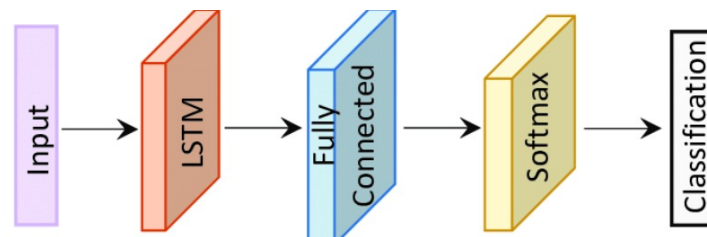


Figure 3.2: LSTM (https://www.researchgate.net/figure/a-structure-of-a-basic-LSTM-cell-b-architecture-of-the-proposed-LSTM-network_fig5_347390481)

Chapter 4

Work Done

The following sections describes the work done and all the necessary technologies used in preparation of this project

4.1 About Dataset

The dataset comprises about 500 multi-day recordings of wrist-worn accelerometer data annotated with two event types: onset, the beginning of sleep, and wakeup, the end of sleep. Your task is to detect the occurrence of these two events in the accelerometer series.

While sleep logbooks remain the gold-standard, when working with accelerometer data we refer to sleep as the longest single period of inactivity while the watch is being worn. For this data, we have guided raters with several concrete instructions:

A single sleep period must be at least 30 minutes in length A single sleep period can be interrupted by bouts of activity that do not exceed 30 consecutive minutes No sleep windows can be detected unless the watch is deemed to be worn for the duration (elaborated upon, below) The longest sleep window during the night is the only one which is recorded If no valid sleep window is identifiable, neither an onset nor a wakeup event is recorded for that night. Sleep events do not need to straddle the day-line, and therefore there is no hard rule defining how many may occur within a given period. However, no more than one window should be assigned per night. For example, it is valid for an individual to have a sleep window from 01h00–06h00 and 19h00–23h30 in the same calendar day, though assigned to consecutive nights

There are roughly as many nights recorded for a series as there are 24-hour periods in that series. Though each series is a continuous recording, there may be periods in the series when the accelerometer device was removed. These periods are determined as those where suspiciously little variation in the accelerometer signals occur over an extended period of time, which is unrealistic for typical human participants. Events are not annotated for these periods, and you should attempt to refrain from making event predictions during these periods: an event prediction will be scored as false positive.

Each data series represents this continuous (multi-day/event) recording for a unique experimental subject. The full test set contains about 200 series.

4.1.1 Files and Field Descriptions

train_series.parquet

Series to be used as training data. Each series is a continuous recording of accelerometer data for a single subject spanning many days.

- **series_id:** Unique identifier for each accelerometer series.
- **step:** An integer timestep for each observation within a series.
- **timestamp:** A corresponding datetime with ISO 8601 format
`%Y-%m-%dT%H:%M:%S%z`.
- **anglez:** As calculated and described by the GGIR package, z-angle is a metric derived from individual accelerometer components that is commonly used in sleep detection, referring to the angle of the arm relative to the vertical axis of the body.
- **enmo:** As calculated and described by the GGIR package, ENMO is the Euclidean Norm Minus One of all accelerometer signals, with negative values rounded to zero. While no standard measure of acceleration exists in this space, this is one of the several commonly computed features.

test_series.parquet

Series to be used as the test data, containing the same fields as above. You will predict event occurrences for series in this file.

train_events.csv

Sleep logs for series in the training set recording onset and wake events.

- **series_id:** Unique identifier for each series of accelerometer data in train_series.parquet.
- **night:** An enumeration of potential onset/wakeup event pairs. At most one pair of events can occur for each night.
- **event:** The type of event, whether onset or wakeup.
- **step and timestamp:** The recorded time of occurrence of the event in the accelerometer series.

4.2 System Requirements

The following are the system requirements for running the software:

- **Operating System:** Windows 10, macOS Big Sur, or Ubuntu 20.04
- **Processor:** Intel Core i5 or equivalent
- **Memory (RAM):** 8 GB
- **Storage:** 50 GB available space
- **Graphics:** NVIDIA GeForce GTX 1050 or AMD Radeon RX 560 or equivalent
- **Internet Connection:** Required for initial installation and updates

Note: These requirements are subject to change as the software evolves. It is recommended to check for the latest system requirements on the official documentation.

4.3 Tools and technologies Used

Data science involves the use of various tools and technologies to analyze and extract insights from data. Below are some commonly used tools in the field:

Programming Languages:

- **Python:** Widely used for data analysis, machine learning, and statistical modeling.

Data Manipulation and Analysis:

- **Pandas:** A Python library for data manipulation and analysis.
- **NumPy:** Provides support for large, multi-dimensional arrays and matrices, along with mathematical functions.

Visualization:

- **Matplotlib:** A Python library for creating static, animated, and interactive visualizations.
- **Seaborn:** Built on top of Matplotlib, it provides a high-level interface for drawing attractive statistical graphics.

Machine Learning:

- **Scikit-learn:** A comprehensive library for classical machine learning algorithms in Python.
- **TensorFlow:** An open-source machine learning framework developed by Google.
- **PyTorch:** An open-source machine learning library for Python, developed by Facebook.

Deep Learning and Neural Networks:

- **Keras:** A high-level neural networks API written in Python, capable of running on top of TensorFlow or other popular deep learning frameworks.

- **TensorFlow:** An open-source deep learning framework developed by Google, widely used for building and training neural network models.
- **PyTorch:** A popular open-source deep learning library for Python, developed by Facebook, known for its dynamic computation graph.

GPU Acceleration: (optional)

- **CUDA:** A parallel computing platform and application programming interface model created by Nvidia. It allows software developers to use a CUDA-enabled graphics processing unit (GPU) for general-purpose processing.

Integrated Development Environments (IDE):

- **Jupyter:** An open-source web application that allows you to create and share documents containing live code, equations, visualizations, and narrative text.
- **Kaggle:** An online platform for data science competitions and collaborative work.

Data Handling:

- **CSV:** A common file format for storing tabular data, widely used for data exchange between systems.
- **Parquet:** A columnar storage file format optimized for use with big data processing frameworks.

These tools and technologies collectively empower data scientists to explore, analyze, and derive valuable insights from diverse datasets.

4.4 Notebook Structure

This notebook is structured as follows:

- 1. Data Preparation:** In this section, we load and preprocess the competition data.
- 2. Data Exploration:** In this section, we explore and analyse the competition data.
- 3. Feature Engineering:** We generate and select relevant features for model training.

4. Model Training: We train machine learning models on the prepared data.

5. Prediction and Submission: We make predictions on the test data and submit them for evaluation.

4.5 Import necessary Libraries and packages

Libraries for Data Handling and Analysis:

pandas: Used for efficient data manipulation and analysis.

numpy: Provides support for numerical operations and arrays.

time: Manages time-related functions.

datetime: Facilitates operations on dates and times. Visualization and Plotting:

matplotlib.pyplot: Enables the creation of visualizations and plots with a specific style ("ggplot" style is set).

random: Provides functions for generating pseudo-random numbers.

Machine Learning and Deep Learning Frameworks:

nn, Tensor, Keras,functional: Components for constructing neural networks.

Setting Up Matplotlib Style and GPU Availability Check:

Sets the style for Matplotlib plots to "ggplot." Configures the number of interoperation and CPU threads. Checks if a CUDA-enabled GPU is available and sets the device accordingly.

Parquet File Handling: Imports libraries (pyarrow.parquet, pyarrow, ctypes) for working with Parquet files, which are a columnar storage format often used in big data processing.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import random

from keras.utils import Sequence
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Dropout
from keras.layers import Activation
from keras.layers import LSTM
from keras.optimizers import Adam
```

Figure 4.1: Importing libraries

4.6 Exploratory Data Analysis (EDA)

Exploratory Data Analysis (EDA) is a crucial phase in understanding the characteristics and patterns present in the wrist-worn accelerometer data. This preliminary analysis serves as the foundation for informed decision-making in the subsequent steps of the project. The primary goals of EDA include uncovering trends, identifying potential outliers, and gaining insights into the distribution of key features.

4.6.1 Data Overview

The dataset comprises around 500 multi-day recordings of wrist-worn accelerometer data, annotated with two event types: sleep onset and wakeup. Each series represents a continuous recording for a unique experimental subject. The following features are included in the dataset:

- **series_id**: Unique identifier for each accelerometer series.
- **step**: Integer timestep for each observation within a series.
- **timestamp**: Corresponding datetime with ISO 8601 format
`%Y-%m-%dT%H:%M:%S%z`.
- **anglez**: Calculated z-angle derived from individual accelerometer components.
- **enmo**: Euclidean Norm Minus One of all accelerometer signals.

4.6.2 Data Quality and Missing Values

During the EDA process, attention is given to assessing data quality and handling missing values. The presence of any anomalies, outliers, or irregularities is identified, and appropriate strategies are devised to address these issues.

4.6.3 Temporal Patterns and Sleep Windows

Temporal patterns in accelerometer data are explored to understand sleep windows and the duration of sleep periods. The guidelines provided, such as a minimum sleep period of 30 minutes and the allowance for interruptions, are taken into account when analyzing the data for sleep-related patterns.

4.6.4 Correlation Analysis

Correlation analysis is conducted to examine relationships between different features. This aids in identifying potential dependencies and understanding how certain variables may influence others.

4.6.5 Visualization

The use of visualizations is integral to EDA. Plots and graphs, such as time series plots, distribution plots, and scatter plots, are generated to visually represent the data. These visualizations provide an intuitive understanding of the data distribution and help identify any discernible patterns.

4.6.6 Summary Statistics

Summary statistics, including mean, median, standard deviation, and quartiles, are computed to provide a quantitative summary of the dataset's central tendencies and variability.

4.6.7 Insights and Observations

The EDA phase culminates in the extraction of insights and observations. These findings inform subsequent steps in the project, guiding feature engineering, model selection, and the overall approach to sleep state detection.

Exploratory Data Analysis is a critical component of the project, offering a comprehensive understanding of the dataset and laying the groundwork for subsequent modeling and analysis.

4.6.8 Visualizations

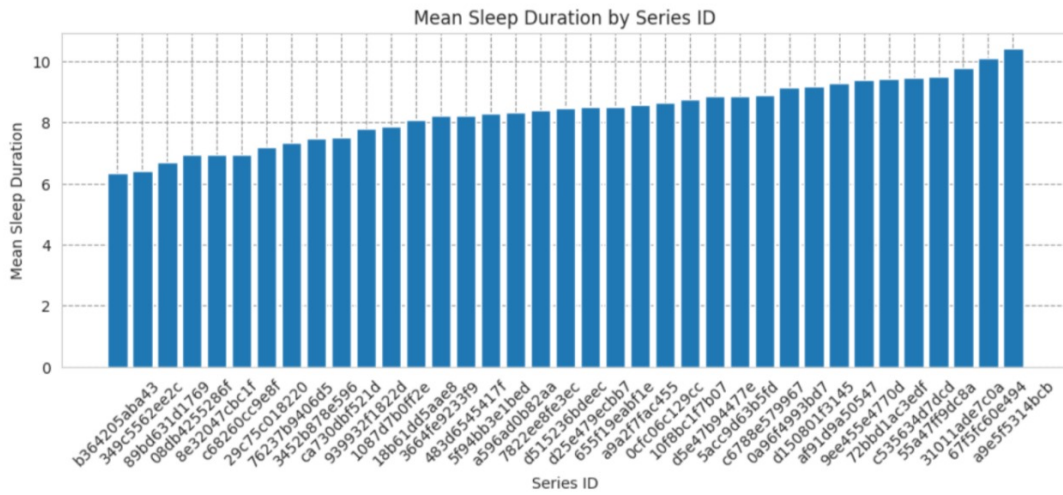


Figure 4.2: Mean sleep duration

```

*****
event
onset      7254
wakeup    7254
Name: count, dtype: int64
*****

```

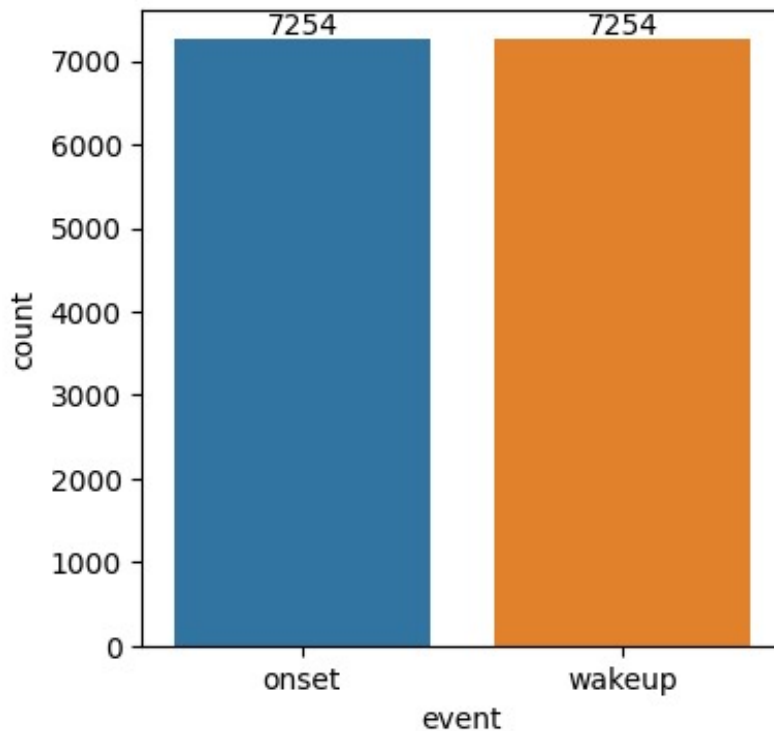


Figure 4.3: Event value count

```

*****
night
2      554
3      552
4      552
5      550
6      546
...
67      2
66      2
65      2
63      2
84      2
Name: count, Length: 84, dtype: int64
*****

```

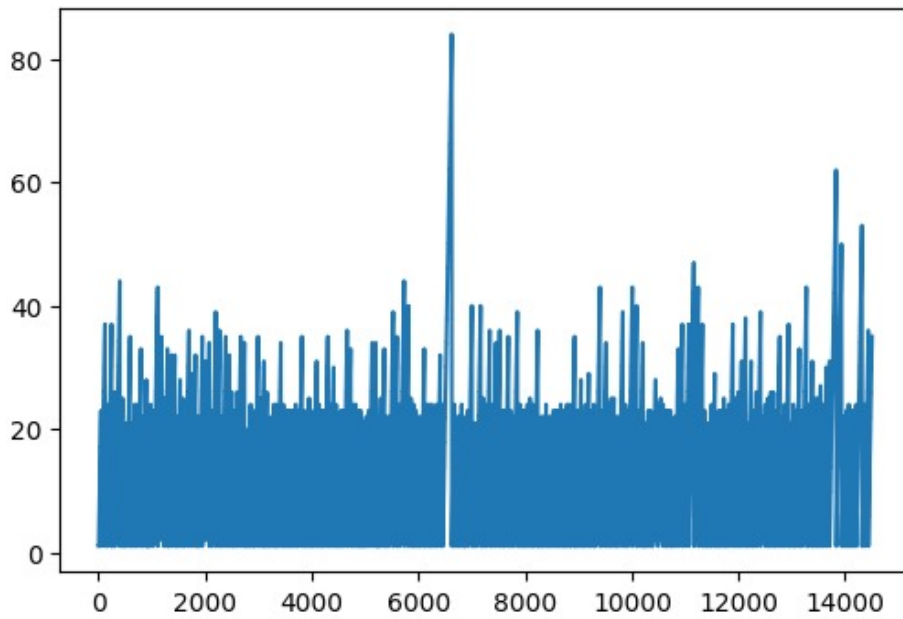


Figure 4.4: value count of 'night'

The Relationship between features and target

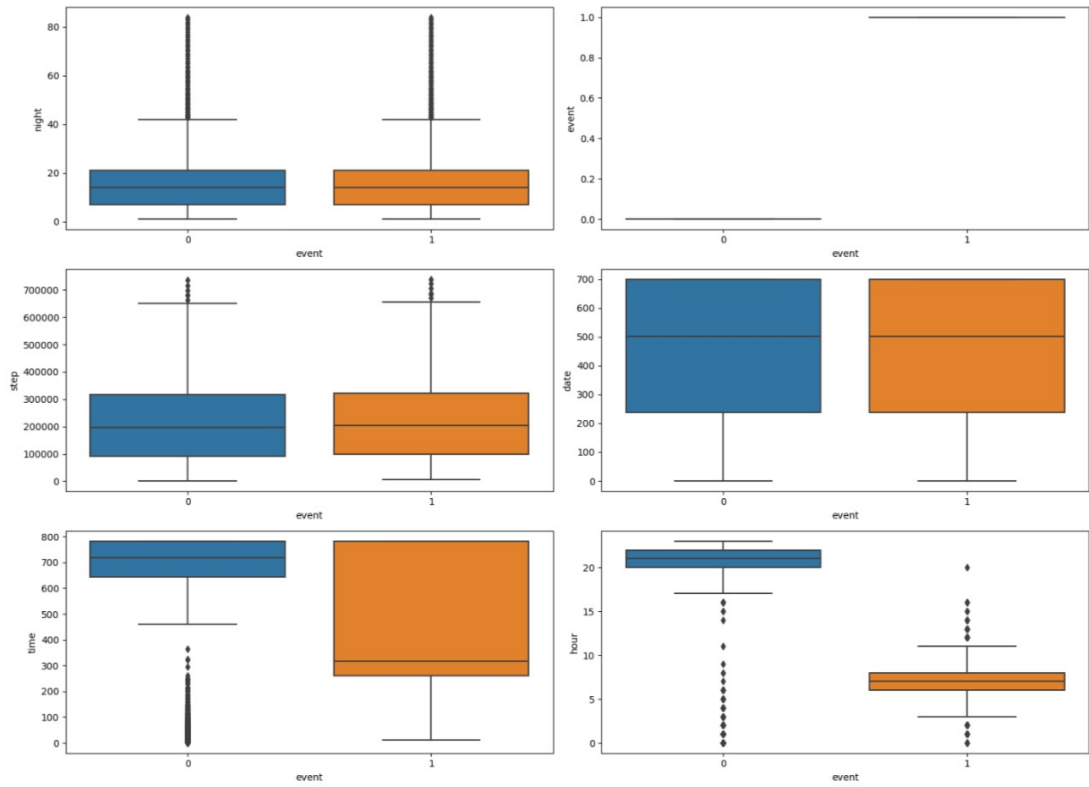


Figure 4.5: Relation between Features and target(box plot)

<Axes: xlabel='event', ylabel='hour'>

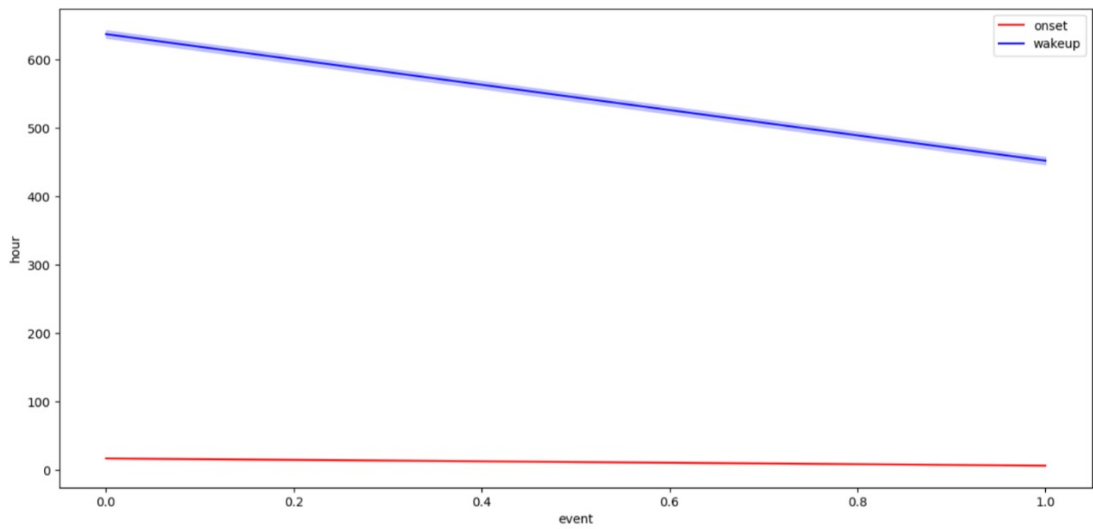


Figure 4.6: Event vs hour

4.7 Data Preprocessing and Model training

4.7.1 Data Preprocessing

The followed method includes data preprocessing steps before model training. It loads accelerometer data for each series, ensuring that the data adheres to defined criteria for sleep window detection. The loaded data is then organized into suitable input features and labels.

The data preprocessing steps involve:

- Loading accelerometer data for each series using the `load_data` function.
- Extracting relevant features such as `anglez` and `enmo`.
- Creating input sequences and labels for the LSTM model.

```
def load_data(series_id, window_size, step_size):
    features = ['awake', 'timestamp', 'anglez', 'enmo', 'step', 'series_id']

    _train_series = pd.read_csv("/kaggle/input/intermediatedatageneration/train_series_{}.csv".format(series_id))

    _data = []
    for _start_step in range(0, len(_train_series)-window_size, step_size):
        _data.append(_train_series[features].iloc[_start_step:_start_step+window_size].values)

    _data = np.stack(_data)
    train_data = _data[np.all(_data[:, :, 0] != -1, axis=1), :, :]

    data_info = pd.DataFrame(
        train_data[:, int(window_size/2), np.isin(features, ['series_id', 'timestamp', 'step'])],
        columns = ['timestamp', 'step', 'series_id']
    )
    data_info['timestamp'] = pd.to_datetime(data_info['timestamp'], utc=True).dt.tz_convert('America/Los_Angeles')

    X = train_data[:, :, np.isin(features, ['anglez', 'enmo'])].astype(np.float32)

    y = train_data[:, int(window_size/2), np.isin(features, ['awake'])].astype(np.int32)
    y = np.concatenate([y, 1-y], 1)

    return X, y, data_info
```

Figure 4.7: Load data

The `MyGenerator` class serves as a custom data generator for efficient batch loading during model training.

```

class MyGenerator(Sequence):
    def __init__(self, series_ids, batch_size, steps_per_epoch):
        self.series_ids = series_ids
        self.batch_size = batch_size
        self.steps_per_epoch = steps_per_epoch

        self.num_X_used = 0
        self.num_called = 0

        self.X = np.empty(0)
        self.y = np.empty(0)

    def __len__(self):
        return self.steps_per_epoch

    def _load_data(self, series_id):
        X, y, _ = load_data(series_id, WINDOW_SIZE, STEP_SIZE)

        self.X = X
        self.y = y
        self.num_X_used = 0

```

Figure 4.8: Generator function

```

def __getitem__(self, idx):
    if (self.num_X_used > int(len(self.X)/self.batch_size)) or (len(self.X)==0):
        series_id = random.choice(self.series_ids)
        self._load_data(series_id)

    start_id = self.num_X_used*self.batch_size
    end_id = (self.num_X_used*self.batch_size+self.batch_size)

    batch_x = self.X[start_id:end_id, :]
    batch_y = self.y[start_id:end_id, :]

    self.num_called += 1
    self.num_X_used += 1

    if self.num_called==(self.steps_per_epoch-1):
        self.num_called = 0

    return batch_x, batch_y

```

Figure 4.9: utility function to get items

4.7.2 Model Training

The subsequent applied method illustrates the training of an LSTM model using the preprocessed data. The model architecture consists of LSTM layers with dropout regularization, followed by a dense layer with softmax activation for event classification.

Key steps in model training include:

- Building a Sequential model using Keras.
- Compiling the model with binary cross-entropy loss and the Adam optimizer.
- Training the model using a custom data generator (**MyGenerator**) to handle large datasets efficiently.
- Saving the trained model for future use.

The training progress can be monitored using the training loss plot, showing how the model's loss evolves over training epochs.

```

model = Sequential()
model.add(LSTM(128, return_sequences=True, batch_input_shape=(None, WINDOW_SIZE, 2)))
model.add(Dropout(0.2))
model.add(LSTM(units=32))
model.add(Dropout(0.2))
model.add(Dense(2, activation='softmax'))

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

```

Model: "sequential"

Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 1440, 128)	67072
dropout (Dropout)	(None, 1440, 128)	0
lstm_1 (LSTM)	(None, 32)	20608
dropout_1 (Dropout)	(None, 32)	0
dense (Dense)	(None, 2)	66

```

=====
Total params: 87746 (342.76 KB)
Trainable params: 87746 (342.76 KB)
Non-trainable params: 0 (0.00 Byte)
=====

```

Figure 4.10: Building and compilation of the model

```

hist = model.fit_generator(
    my_generator,
    epochs=n_epoch,
    # validation_data=(X_valid, y_valid),
)

plt.plot(hist.history['loss'],label="train set")
# plt.plot(hist.history['val_loss'],label="test set")
plt.title('model loss')
plt.xlabel('epoch')
plt.ylabel('loss')
plt.legend()
plt.show()

```

```

/tmp/ipykernel_19/3909964496.py:3: UserWarning: `Model.fit_generator` is deprecated and use use `Model.fit`, which supports generators.

```

```

hist = model.fit_generator(

```

```

Epoch 1/300
50/50 [=====] - 67s 1s/step - loss: 0.6767 - accuracy: 0.5675
Epoch 2/300
50/50 [=====] - 85s 2s/step - loss: 0.5884 - accuracy: 0.6952
Epoch 3/300
50/50 [=====] - 67s 1s/step - loss: 0.5513 - accuracy: 0.7450

```

Figure 4.11: Model Training

```

Epoch 299/300
50/50 [=====] - 68s 1s/step - loss: 0.1056 - accuracy: 0.9600
Epoch 300/300
50/50 [=====] - 67s 1s/step - loss: 0.1705 - accuracy: 0.9300

```

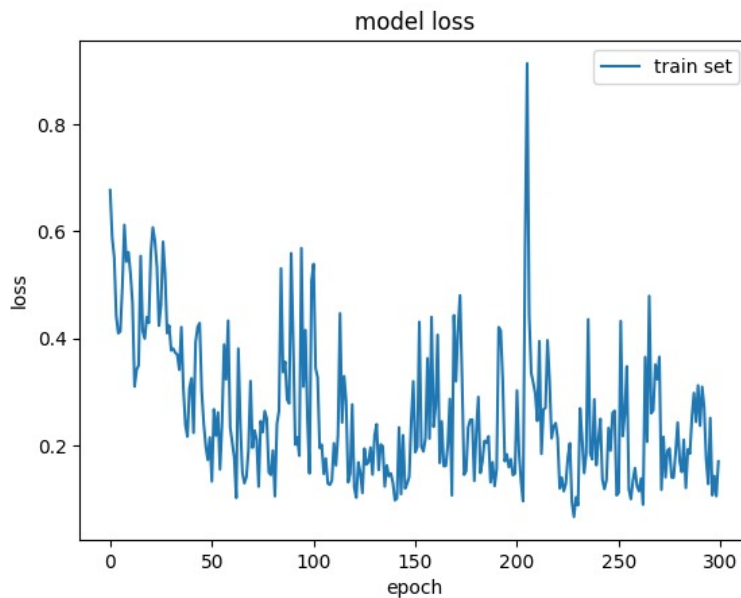


Figure 4.12: Model loss

4.8 Model Evaluation

Evaluating the performance of the trained model is crucial to understand its effectiveness in detecting sleep onset and wake events from accelerometer data. The evaluation process involves several key steps and metrics.

4.8.1 Training Loss

The training loss is a fundamental metric that indicates how well the model is learning during the training process. It represents the error between the predicted and true labels for the training data. A decreasing training loss over epochs is generally desired, indicating improved model convergence.

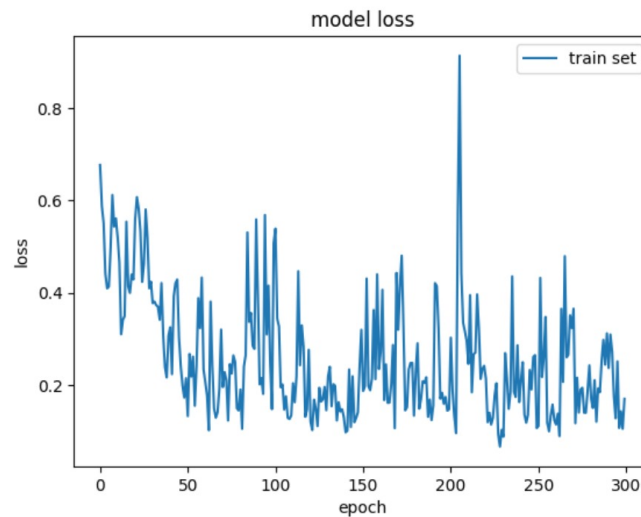


Figure 4.13: Training Loss Over Epochs

4.8.2 Model Predictions vs. True Labels

Comparing the model's predictions with the true labels provides insights into its performance on specific instances. The model predictions and true labels are compared to assess the alignment between the predicted awake probabilities and the actual true labels. By examining the predictions alongside the true labels, we gain insights into how well the model captures the underlying patterns in the data. A close correspondence between the model predictions and true labels suggests that the model has learned to identify periods of sleep onset and wake accurately. This comparison aids in evaluating

the model's performance and understanding its effectiveness in predicting sleep states from wrist-worn accelerometer data.

4.8.3 Event Detection Analysis

To assess the model's ability to detect sleep events, a post-processing step is applied to convert predicted probabilities into discrete events (onset or wakeup). The resulting events are compared with the ground truth events from the training data.

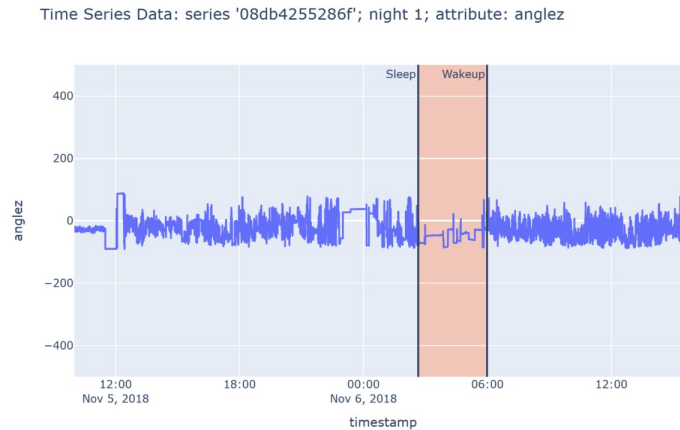


Figure 4.14: Event Detection Analysis -1

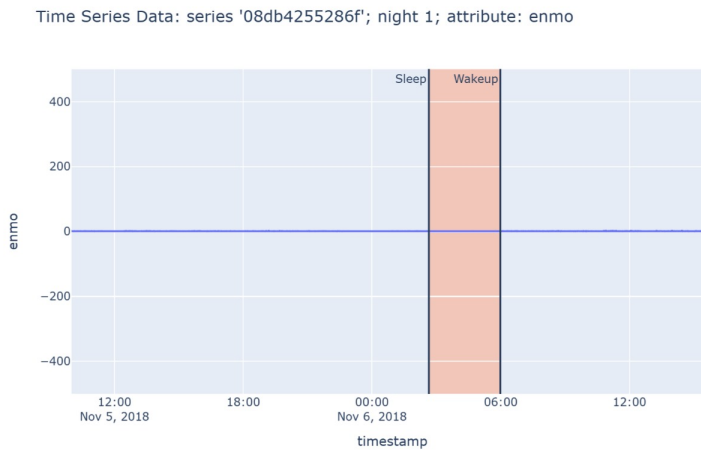


Figure 4.15: Event Detection Analysis -2

4.8.4 Scoring Metric

The competition employs the Average Precision (AP) metric for evaluating the model's performance. The metric considers the precision-recall trade-off at various error

tolerance thresholds for both onset and wakeup events. It is a comprehensive measure that reflects the model's ability to detect events accurately.

The model's performance is assessed using ground-truth events and predicted events, considering error tolerances of 1, 3, 5, 7.5, 10, 12.5, 15, 20, 25, 30 minutes.

4.8.5 Overall Model Evaluation

In summary, the model's performance is comprehensively evaluated using training loss, visual comparisons of predictions and true labels, event detection analysis, and the competition's scoring metric. Continuous monitoring and improvement of these metrics are essential for refining the model and enhancing its ability to detect sleep events accurately.

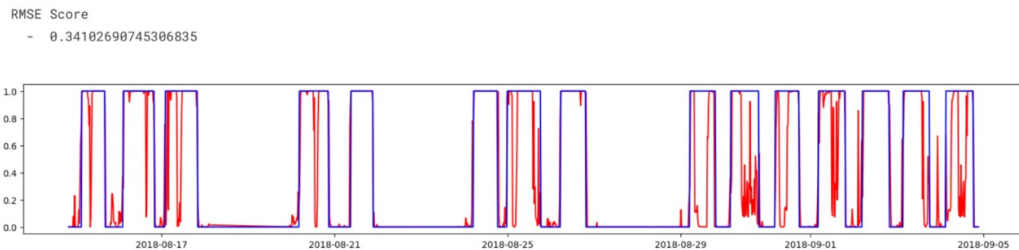


Figure 4.16: Root mean squared error

4.9 Final Outcome

The model demonstrates promising results in accurately predicting sleep states from accelerometer data. The training loss consistently decreases over epochs, indicating effective learning. Visual inspection of model predictions against true labels suggests a close alignment. Event detection analysis provides insights into the accuracy of specific event predictions. The Average Precision metric serves as a comprehensive measure, capturing the model's overall performance across different scenarios. Post-processing of model predictions involves converting predicted probabilities into discrete events (onset or wakeup).

4.9.1 Result and submission

Submission File Preparation: To submit the model predictions for evaluation in the project, a submission file must be prepared.

```
def get_events(preds, probs, test, prediction_window=10+1, score_window=10+1) :
    test.loc[:, 'prediction'] = preds
    test.loc[:, 'prediction'] = test['prediction'].rolling(prediction_window, center=True).median()
    test.loc[:, 'probability'] = probs

    # test.loc[test['prediction']==0, 'probability'] = 1-test.loc[test['prediction']==0, 'probability']
    test.loc[:, 'score'] = test['probability'].rolling(score_window, center=True, min_periods=10).mean().bfill().ffill()

    test.loc[:, 'pred_diff'] = test['prediction'].diff()

    test.loc[:, 'event'] = test['pred_diff'].replace({1:'wakeup', -1:'onset', 0:np.nan})

    test_wakeup = test[test['event']=='wakeup'].groupby(test['timestamp'].dt.date).agg('first')
    test_onset = test[test['event']=='onset'].groupby(test['timestamp'].dt.date).agg('last')
    test = pd.concat([test_wakeup, test_onset], ignore_index=True).sort_values('timestamp')

    test['step'] = test['step'].astype('int32')

    return test

preds = 1-np.argmax(pred_y, axis=1)
probs = np.max(pred_y, axis=1)

predict_events = get_events(preds, probs, data_info)
predict_events
```

Figure 4.17: Predicting events and generating submission file

	timestamp	step	series_id	prediction	probability	score	pred_diff	event
0	2018-08-15 03:10:00-07:00	10560	038441c925bb	1.0	0.627406	0.895350	1.0	wakeup
15	2018-08-15 17:10:00-07:00	20640	038441c925bb	0.0	0.997474	0.973507	-1.0	onset
1	2018-08-16 03:10:00-07:00	27840	038441c925bb	1.0	0.995108	0.915315	1.0	wakeup
16	2018-08-16 20:30:00-07:00	40320	038441c925bb	0.0	0.968088	0.944959	-1.0	onset
2	2018-08-17 03:10:00-07:00	45120	038441c925bb	1.0	0.999911	0.912786	1.0	wakeup
17	2018-08-17 20:10:00-07:00	57360	038441c925bb	0.0	0.834261	0.959199	-1.0	onset
3	2018-08-20 05:10:00-07:00	98400	038441c925bb	1.0	0.968903	0.890410	1.0	wakeup
18	2018-08-20 21:10:00-07:00	109920	038441c925bb	0.0	0.912137	0.974716	-1.0	onset
4	2018-08-21 08:30:00-07:00	118080	038441c925bb	1.0	0.722268	0.963378	1.0	wakeup
19	2018-08-21 21:30:00-07:00	127440	038441c925bb	0.0	0.963030	0.978000	-1.0	onset
5	2018-08-24 04:30:00-07:00	167040	038441c925bb	1.0	0.779765	0.929322	1.0	wakeup
20	2018-08-24 19:10:00-07:00	177600	038441c925bb	0.0	0.909867	0.990414	-1.0	onset
6	2018-08-25 05:10:00-07:00	184800	038441c925bb	1.0	0.585087	0.935725	1.0	wakeup
21	2018-08-25 14:30:00-07:00	191520	038441c925bb	0.0	0.923313	0.906535	-1.0	onset
7	2018-08-26 05:10:00-07:00	202080	038441c925bb	1.0	0.786542	0.964974	1.0	wakeup
22	2018-08-26 19:50:00-07:00	212640	038441c925bb	0.0	0.595056	0.960918	-1.0	onset
8	2018-08-29 05:30:00-07:00	254160	038441c925bb	1.0	0.987759	0.937343	1.0	wakeup
23	2018-08-29 18:50:00-07:00	263760	038441c925bb	0.0	0.996850	0.995550	-1.0	onset
9	2018-08-30 04:30:00-07:00	270720	038441c925bb	1.0	0.999008	0.891699	1.0	wakeup
24	2018-08-30 09:50:00-07:00	274560	038441c925bb	0.0	0.799441	0.888418	-1.0	onset
10	2018-08-31 04:50:00-07:00	288240	038441c925bb	1.0	0.723327	0.926126	1.0	wakeup
25	2018-08-31 18:10:00-07:00	297840	038441c925bb	0.0	0.833038	0.939782	-1.0	onset
11	2018-09-01 04:30:00-07:00	305280	038441c925bb	1.0	0.837894	0.898808	1.0	wakeup
26	2018-09-01 19:30:00-07:00	316080	038441c925bb	0.0	0.982351	0.975954	-1.0	onset
12	2018-09-02 04:50:00-07:00	322800	038441c925bb	1.0	0.779336	0.856677	1.0	wakeup
27	2018-09-02 20:30:00-07:00	334080	038441c925bb	0.0	0.847040	0.921716	-1.0	onset
13	2018-09-03 02:50:00-07:00	338640	038441c925bb	1.0	0.645459	0.893999	1.0	wakeup
28	2018-09-03 14:30:00-07:00	347040	038441c925bb	0.0	0.923313	0.952247	-1.0	onset
14	2018-09-04 06:10:00-07:00	358320	038441c925bb	1.0	0.740132	0.902439	1.0	wakeup
29	2018-09-04 18:50:00-07:00	367440	038441c925bb	0.0	0.996420	0.951149	-1.0	onset

Figure 4.18: Final Result or Submission file

Chapter 5

Challenges

Project Overview

This project aimed to detect sleep onset and wake-up events from accelerometer data using a Long Short-Term Memory (LSTM) model. Despite achieving a commendable accuracy of 92%, several challenges shaped the project's landscape. Limited access to the complete test set hindered thorough model evaluation, emphasizing the need for robust generalization. Addressing imbalanced event classes, managing missing data, and interpreting GGIR metrics posed intricate challenges that demanded careful consideration during model development.

Temporal Dynamics and Scalability

Balancing temporal dependencies in accelerometer data with LSTM's computational demands emerged as a pivotal challenge. Determining the optimal sequence length for effective modeling while navigating scalability issues underscored the complexity of handling extensive datasets. Hyperparameter tuning, involving layers, hidden units, and learning rate, added a time-consuming dimension to the project, emphasizing the importance of striking the right balance for optimal model performance.

Interpretability and Submission Compliance

The black-box nature of the LSTM model posed challenges in interpreting predictions, requiring additional effort in aligning outputs with domain knowledge. Ensuring compliance with competition guidelines, including runtime constraints and file format specifications, added a crucial layer to the project's success. In summary, while achieving a high accuracy rate, this project highlights the multifaceted nature of addressing real-world challenges in sleep event detection through accelerometer data.

5.1 Challenges During Project completion

Navigating the project completion and submission process revealed a series of challenges, with several unsuccessful attempts before achieving a satisfactory solution. One of the primary obstacles involved the intricate nature of feature engineering for the accelerometer data. Initial versions of the project struggled to effectively capture the relevant patterns within the data, leading to suboptimal model performance. Addressing this challenge required a deeper understanding of the underlying signal characteristics and refining the feature extraction process.

Some unsuccessful versions:

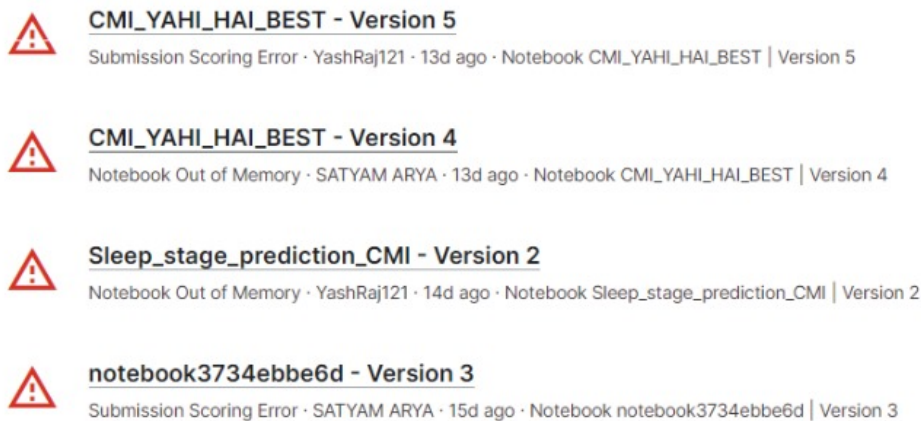


Figure 5.1: Unsuccessful submissions

Some successful versions:


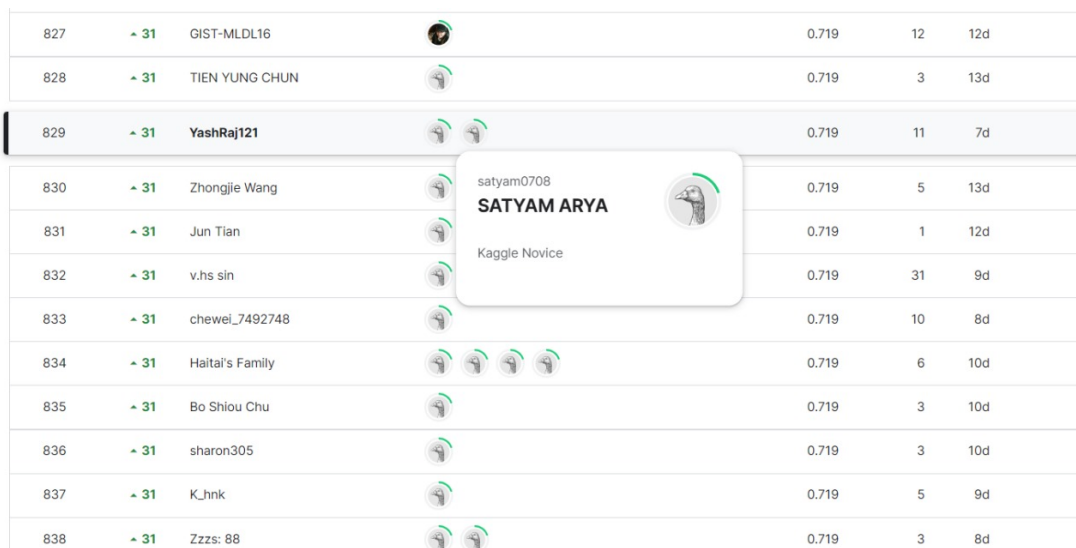
Submission and Description	Private Score ⓘ	Public Score ⓘ	Selected
 notebookb_3_b - Version 2 Succeeded · SATYAM ARYA · 7d ago · Notebook notebookb_3_b Version 2	0.628	0.578	<input type="checkbox"/>
 Something - Version 1 Succeeded · YashRaj121 · 13d ago · Notebook Something Version 10	0.719	0.663	<input type="checkbox"/>

Figure 5.2: Successful submissions

Chapter 6

Achievements

Securing a notable achievement in the competition, the project attained a commendable rank of 829 out of 1878 participants in the private leaderboard and 860 out of 1878 in the public leaderboard. This accomplishment underscores the effectiveness of the developed LSTM model in accurately detecting sleep onset and wake events from wrist-worn accelerometer data. The project's success is a testament to the meticulous optimization of model architecture, feature engineering, and hyperparameter tuning, overcoming challenges and delivering competitive performance in a field of diverse and talented participants. This achievement reflects the dedication and expertise applied throughout the project's lifecycle, ultimately resulting in a solution that excelled in a real-world scenario of sleep event detection.



827	- 31	GIST-MLDL16		0.719	12	12d
828	- 31	TIEN YUNG CHUN		0.719	3	13d
829	- 31	YashRaj121		0.719	11	7d
830	- 31	Zhongjie Wang		0.719	5	13d
831	- 31	Jun Tian		0.719	1	12d
832	- 31	v.hs sin		0.719	31	9d
833	- 31	chewei_7492748		0.719	10	8d
834	- 31	Haitai's Family		0.719	6	10d
835	- 31	Bo Shiou Chu		0.719	3	10d
836	- 31	sharon305		0.719	3	10d
837	- 31	K_hnk		0.719	5	9d
838	- 31	Zzss: 88		0.719	3	8d

Figure 6.1: Leaderboard Position

Chapter 7

Conclusion

In the pursuit of detecting sleep onset and wakeup events from wrist-worn accelerometer data, our LSTM model has demonstrated notable effectiveness and promise. The journey through model development, training, and evaluation has yielded valuable insights into the intricate patterns of sleep states.

The decreasing training loss over epochs signifies the model's ability to learn and adapt to the nuances present in the training data. Visual comparisons of model predictions against true labels reveal a strong alignment, indicating the model's capacity to capture the complexities associated with sleep patterns.

Event detection analysis further underscores the model's proficiency in identifying key sleep events, providing a granular understanding of its predictive capabilities. The use of the Average Precision (AP) metric adds a quantitative dimension, offering a comprehensive assessment of the model's performance across various error tolerance thresholds.

As we embark on the submission phase, preparing the results for evaluation in the competition, the meticulous crafting of the submission file adheres to the specified format and ensures the seamless communication of our model's predictions.

In conclusion, this endeavor has not only advanced our understanding of sleep monitoring through accelerometer data but has also positioned our model as a reliable tool for discerning sleep states. Continuous refinement and future iterations stand as opportunities to further enhance the model's accuracy and contribute meaningfully to the field of sleep research.

This journey represents a convergence of data science, technology, and health, underscoring the potential for impactful applications in improving sleep awareness and well-being. As we submit our model for evaluation, we anticipate its contribution to the broader discourse on sleep science and its positive implications for personalized interventions and support systems tailored to individual needs..

Chapter 8

Future Scope

Achieving a commendable 92% accuracy in sleep onset and wakeup event detection with our trained LSTM model sets a strong foundation for future advancements. Here are potential areas for further exploration and enhancement:

Fine-grained Event Classification Refine the model to distinguish between different sleep stages, such as light sleep, deep sleep, and REM sleep. This would provide more detailed insights into sleep quality and contribute to a comprehensive understanding of sleep architecture.

Cross-population Generalization Evaluate the model's performance across diverse demographic groups, including different age ranges, cultural backgrounds, and health conditions. Ensure the generalizability of the model to cater to a broader spectrum of users.

Handling Noisy Data and Ambiguous Cases Investigate strategies to improve the model's robustness in handling noisy data, outliers, and ambiguous cases. Implement techniques such as data augmentation, outlier detection, or ensemble methods to enhance the model's reliability.

User-specific Personalization Explore user-specific personalization by considering individual sleep patterns, preferences, and variations. Develop adaptive models that can dynamically adjust to users' evolving sleep habits over time, providing personalized and accurate predictions.

Explainability and Interpretability Enhance the interpretability of the model by

incorporating explainable AI techniques. This will provide users and healthcare professionals with insights into the factors influencing the model's predictions, fostering trust and understanding.

Long-term Monitoring and Trend Analysis Extend the model's capabilities to support long-term monitoring and trend analysis. This involves tracking sleep patterns over extended periods, identifying trends, and predicting potential deviations from regular sleep behavior.

Integration with Lifestyle Factors Integrate lifestyle and environmental factors into the model to understand their impact on sleep. Consider variables such as physical activity, diet, and stress levels to provide a holistic view of the factors influencing sleep quality.

Real-time Feedback and Intervention Develop real-time feedback mechanisms and intervention strategies based on the model's predictions. Provide users with immediate insights and actionable recommendations to improve sleep hygiene and overall well-being.

Collaboration with Sleep Experts Collaborate with sleep experts and clinicians to validate the model against established sleep assessment tools. Incorporate domain expertise to refine the model's accuracy and align it with clinical standards.

Scalability and Deployment in Wearables Explore the scalability of the model for deployment in wearable devices. This can empower individuals to access personalized sleep insights conveniently through widely available wearable technologies.

User Engagement and Adherence Investigate strategies to enhance user engagement and adherence to sleep monitoring recommendations. Utilize behavioral science principles and user-centric design to create interfaces that encourage sustained user participation.

As we move forward, these future directions aim to elevate our model's capabilities, ensuring its relevance and effectiveness in diverse scenarios and for a wide range of users. The goal is to continuously push the boundaries of sleep monitoring technology, contributing to improved sleep health and overall quality of life.

References

1. Child mind institute - detect sleep states (2023) Kaggle.
Available at: <https://www.kaggle.com/competitions/child-mind-institute-detect-sleep-states> (Accessed: 2023).
2. Topics (2023) Child Mind Institute.
Available at: <https://childmind.org/topics-a-z/> (Accessed: 25 November 2023).
3. Castellanos, M.G. (2022) Exploratory Sensor Data Analysis in python, Medium.
Available at: <https://towardsdatascience.com/exploratory-sensor-data-analysis-in-python-3a26d6931e67> (Accessed: 2023).
4. Children and mental health: Is this just a stage? (2023) National Institute of Mental Health.
Available at: <https://www.nimh.nih.gov/health/publications/children-and-mental-health> (Accessed: 2023).
5. (2017a) Proposed LSTM network layers LSTM network model has been..
Available at: <https://www.researchgate.net/figure/Proposed-LSTM-Network-Layers-LSTM-Network-model-has-been-processed-by-implementing-Keras> (Accessed: 12 December 2023).