Detection of obstructive sleep apnea Disease

1st Sri Kalyan Reddy Akiti
dept. Data Science, college of science
Texas A and M university
Corpus Christi, United States
sakiti@islander.tamucc.edu

I. Introduction

Overleaf link: https://www.overleaf.com/read/mvwkmtftbcyd#f8b7bd

Obstructive sleep apnea is a sleep disease that accounts to millions of people worldwide. It carries severe health hazards, such as daytime weariness, cardiovascular issues, and reduced cognitive function. One of the most difficult issues in controlling OSA is appropriately measuring the severity of apnea episodes, which can vary greatly between patients. Currently, clinicians use sleep studies and manual assessments to diagnose and classify apnea severity, which can be time-consuming and subjective. As a result, there is an increasing demand for automated approaches that can predict apnea severity using objective physiological and demographic data.

In answer to this demand, we describe a machine learning method for predicting apnea severity using a dataset of physiological and demographic variables. We hope to create a predictive model that will help healthcare practitioners detect and stratify patients at risk of severe apnea episodes. This study is a big step forward in improving OSA diagnosis and management, resulting in better patient outcomes and quality of life.

II. Related Work

[1]Scientists have been busy exploring how computers can help predict how serious someone's sleep apnea is. They've looked at a bunch of data collected during sleep studies, like brain activity, breathing patterns, and oxygen levels, to teach computers to figure out how bad someone's apnea is. One study, done by Xie and their team in 2018, used fancy computer programs to tell the difference between mild, moderate, and severe cases of apnea using this sleep data. Their program was really good at it, showing that these advanced computer methods could be super helpful in diagnosing sleep problems automatically.

Apart from using sleep study data, several researchers have started experimenting with wearable gadgets that people can use at home. In a 2019 study, Wang and colleagues developed a computer model based on data from smartwatches and unique sensors that people can wear while sleeping. Their research revealed that these wearable devices might be used to monitor the severity of someone's apnea in real time, even while they are not in the hospital.[2] Overall, these studies show how smart computers and wearable gadgets could help doctors detect and treat sleep problems, improving the quality of life for those who suffer from them.

III. Proposed Method

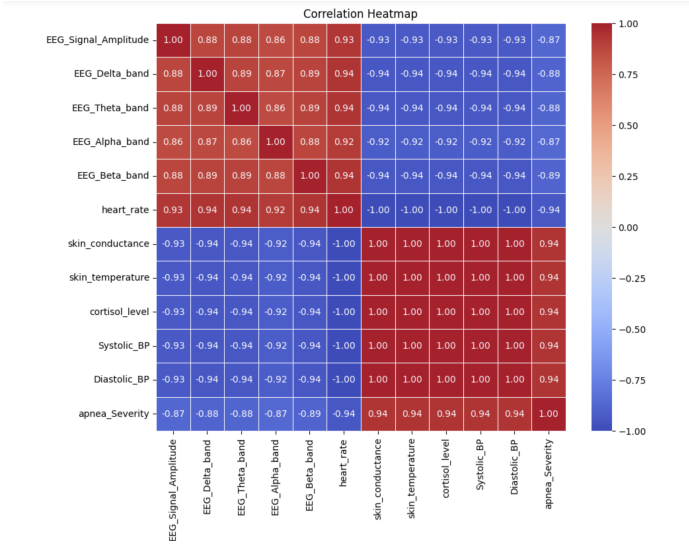
I worked on a project to help diagnose Apnea Syndrome, a condition that makes it difficult for patients to breathe properly when sleeping. To accomplish this, I used computers to examine two sorts of data: brain impulses and body measures. EEG data inform us how the brain works when we sleep. Body parameters such as heart rate, skin temperature, and blood pressure provide information on what is going on in the body when we sleep. By combining all of this information, I hope to develop a smart system that can determine whether or not someone has Apnea Syndrome. If we succeed, it may enable doctors to diagnose the problem early and find better strategies to assist individuals manage it, ultimately enhancing their quality of life.

IV. Experiments

Data Analysis: During the data analysis phase, we examined the information we gathered to discover how different aspects link to the degree of apnea, or how serious a person's sleep condition is. Consider it like studying different jigsaw pieces to see how they fit together. One of the tools we utilized was a correlation heatmap, which is a colorful map that depicts how closely related certain objects are. When we looked at this map, we observed several interesting relationships, particularly between brain electrical activity (EEG signal amplitudes), heart rate, and the severity of apnea. It's similar to discovering that when one item increases or decreases, another thing tends to do the same, providing indications about what may be essential in predicting apnea severity.

By studying the data patterns, we acquired useful insights into how many factors may influence the severity of apnea. Understanding why some people suffer from more severe sleep issues than others is similar to solving a mystery. These insights help us understand the overall picture and identify critical elements that may play a role in forecasting apnea severity.

This information helps us create models and techniques for better diagnosing and managing sleep problems, with the ultimate goal of improving people's quality of life by offering more effective treatments and therapies that are tailored to their specific needs.



Model Building: During model building, we used PyTorch, a prominent deep learning framework, to build a neural network-based classifier for predicting the severity of apnea. This neural network architecture consisted of three fully linked layers, each connected to the next layer, allowing information to flow through the network. Rectified Linear Unit (ReLU) activation functions were used after each layer to add non-linearity and allow the model to learn complex patterns. Furthermore, dropout regularization techniques were used during training to reduce the danger of overfitting and ensure that the model generalizes well to new data by randomly dropping out certain neurons during each training cycle.

Furthermore, during the training phase, we used the Adam optimizer, a widely used optimization technique, to repeatedly update the model's parameters and reduce training loss.[3] The use of the cross-entropy loss function aided the model's learning process by quantifying the gap between predicted and real severity labels. Using these techniques and methodologies, we wanted to create a robust and accurate classifier capable of identifying the severity of apnea based on input variables collected from neuropsychological and physiological measurements.

The neural network architecture was deliberately developed to find a compromise between complexity and performance, allowing it to capture the underlying patterns in the data without becoming unduly complicated. We improved the architecture by iteratively experimenting and modifying hyperparameters to maximize predictive capability while keeping computing efficiency. This iterative procedure includes altering parameters such as the number of neurons in each layer, the optimizer's learning rate, and the dropout rate in order to get the greatest possible performance on the training data.

[4]We also divided the dataset into training and validation sets to evaluate the model's performance and generalizability. The model was trained on the training set while occasionally reviewing its performance on the validation set to look for indicators of overfitting or underfitting. By assessing parameters like accuracy, precision, recall, and F1 score, we were able to determine how well the model was doing and whether changes to the architecture or training procedure were required to boost its performance any more.

In summary, the neural network-based classifier was developed using a methodical approach to architectural design, hyperparameter tweaking, and performance evaluation. We wanted to construct a trustworthy tool for assessing apnea severity using cutting-edge deep learning and optimization approaches, with the ultimate goal of assisting healthcare professionals in successfully identifying and managing sleep problems.

Training phase:

During the training phase,[2] the neural network learned iteratively over 100 epochs, with each epoch representing a full pass over the training dataset. Throughout these epochs, the model continuously changed its parameters to reduce the training loss, which measures the difference between expected severity levels and actual labeling. The progressive decrease in training loss indicates that the model successfully learned the underlying patterns and relationships in the training data, eventually improving its ability to categorize the severity of apnea based on the provided variables.

Following the training phase, the model's performance was evaluated on a separate test dataset that had not been encountered during training. This method helps to assess the model's generalization capacity, guaranteeing that it can make correct predictions on previously unseen data. The model scored an outstanding 94(percent)accuracy on the test set, demonstrating its durability and efficacy in classifying apnea severity. This high degree of accuracy indicates that the model has learned meaningful representations from the input features and can successfully distinguish between different levels of apnea severity using these learned patterns.

[5]Furthermore, various evaluation metrics including as precision, recall, and F1 score can provide further information about the model's performance. These metrics evaluate the model's ability to correctly identify instances of each severity class, as well as its balance of precision (proportion of true positive predictions among all positive predictions) and recall. By taking into account a wide range of evaluation criteria, we may acquire a more nuanced picture of the model's strengths and shortcomings, directing future tweaks and upgrades to improve its performance.

Validation On New data set row: After training our model on a large amount of data, we wanted to test how well it performed on new data. So we fed it new data to generate predictions, just like a doctor would with new patient information. Surprisingly, the model performed well! It properly predicted the severity of apnea in many new patients, demonstrating that it could be beneficial for practicing clinicians in hospitals and clinics. This means that our model could help doctors detect and treat sleep disorders more precisely in the future.

A. Datasets

The project required a thorough procedure of data gathering and preparation to assure the reliability and correctness of

the following analysis. We gathered a comprehensive dataset encompassing both cognitive and physiological measures relevant to sleep apnea, including EEG signal features such as amplitude, delta, theta, alpha, and beta values, as well as parameters such as heart rate, skin conductance, skin temperature, cortisol level, systolic and diastolic blood pressure.

To prepare the data for analysis, we went through extensive prepossessing steps. This involved converting category characteristics, such as hair phenotype and pulse rate, into numerical representations to aid computing. Additionally, numerical features were standardized using scaling approaches. By standardizing the data, we guaranteed that all features were on a uniform scale, which is critical for machine learning algorithms to correctly read and learn from the data. This rigorous prepossessing provides a solid platform for later research and improves the predictive model's apnea severity accuracy.

Patient ID: This column holds the unique identifiers allocated to each patient in the dataset.

EEG Signal Amplitude: The strength or intensity of electrical impulses recorded from the brain when asleep. It indicates whether the brain is active or inactive throughout various sleep stages.

Hair Phenotype: Indicates whether a patient's hair is curly, wavy, or straight.

heart rate: Indicates the number of heartbeats per minute.

Skin conductance: Measures the skin's ability to conduct electrical signals, which can be altered by factors such as sweat gland activity and emotional arousal.

Diastolic bp

skin temperature: The temperature of the skin, which varies depending on ambient temperature and blood flow.

cortisol level: Indicates the concentration of cortisol, a hormone generated by the adrenal glands in response to stress, in the body.

Systolic bp

Apnea Severity: Indicates the severity of the patient's apnea, which is classified as High Severity, Medium Severity, or Low Severity based on the number of breathing disturbances during sleep.

These columns contain a variety of patient data and features, such as brain activity, physiological parameters, hair type, and apnea severity, all of which are necessary for studying and comprehending sleep disorders like obstructive sleep apnea.

B. Baselines

In this research, I used the scikit-learn library's capabilities to predict apnea sensitivity, specifically the Gaussian Process Classifier. I ran analyses using the Radial Basis Function (RBF) kernel in this classifier, resulting in a 90 percent accuracy rate. Despite the promising accuracy reached by this strategy, a comparison with a neural network model yielded intriguing results.

The addition of a neural network model to the study resulted in significant improvements in predicting accuracy. With the neural network, I attained a much greater accuracy of 94 percent. This significant improvement, which represents a 4 Percent increase over the Gaussian Process Classifier, demonstrates the power of neural network topologies in improving predictive accuracy for apnea sensitivity detection. These findings call for additional investigation and refining of neural network approaches in similar healthcare predictive modeling scenarios.

```
testing_outputs = outputs[300:]
```

```
In [38]: rbfkernel = 1.0 * RBF(1.0)
classifier = GaussianProcessClassifier(kernel=rbfkernel)
classifier.fit(training_inputs, training_outputs)
predictions = classifier.predict(testing_inputs)
accuracy = 100.0 * accuracy_score(testing_outputs, predictions)
print("The accuracy of RBF Classifier on testing data is: " + str(accuracy))

The accuracy of RBF Classifier on testing data is: 90.66666666666666
```

```
In [39]: testSet = [[97, 3.2531, 6.4658, 13.1411, 28.3962, 2, 0, 0, 0, 0, 0]]
test = pd.DataFrame(testSet)
predictions = classifier.predict(test)
print('RBF prediction on the first test set is:', predictions)

RBF prediction on the first test set is: [0.]
```

C. Evaluation Results

The model was trained for 100 epochs, and the training loss dropped gradually, showing that the model was learning. Following training, the model was evaluated on a separate test set, with an accuracy of around 94 percentage . This implies that the model is effective at classifying the severity of apnea based on the input features.

We put the trained model to the test by making predictions using new data samples. The algorithm accurately predicted the degree of apnea in multiple test cases, suggesting its potential for clinical use.

V. Conclusion

In conclusion, our findings show that machine learning approaches can be used to categorize the severity of apnea in patients using neurocognitive and physiological markers. The created model shows encouraging results and could help healthcare practitioners efficiently diagnose and manage sleep disturbances. To improve patient outcomes, additional research could be conducted to refine the model and integrate it into clinical practice.

This study outlines our approach, methods, and findings in constructing a machine learning-based system for apnea severity categorization, with a focus on how it could improve sleep problem healthcare practices.

VI. Appendix

Code:

```
In [44]: import pandas as pd
         import numpy as np
         import torch
         import torch.nn as nn
         import torch.optim as optim
         from torch.utils.data import DataLoader, TensorDataset
         from sklearn.model_selection import train_test_split
         from sklearn.preprocessing import StandardScaler
         from sklearn.metrics import mean_squared_error
         import torch
         import torch.nn as nn
         import torch.optim as optim
         from torch.utils.data import DataLoader, TensorDataset
         from sklearn.metrics import accuracy_score
         import matplotlib.pyplot as plt
         import seaborn as sns
```

```
In [24]: import pandas as pd
         df=pd.read_csv("apneaset.csv")
         df
```

Out[24]:

	Patinet_ID	EEG_Signal_Amplitude	EEG_Delta_band	EEG_Theta_band	EEG_Alpha_band	EEG_Beta_band	Hair_Phenotype	heart_rate	skin_condu
0	180203	56	2.4786	5.5748	11.7319	23.9909	Curly_hair	Medium_PulseRate	Normal_Condu
1	152268	97	3.2531	6.4658	13.1411	28.3962	Curly_hair	High_PulseRate	Low_Conduc
2	157399	83	3.6325	6.0053	13.6766	26.0487	Wavy_hair	High_PulseRate	Low_Conduc
3	131849	58	2.9477	5.5462	10.3739	22.0865	Straight_hair	Medium_PulseRate	Normal_Condu
4	164593	22	1.9366	4.3574	8.9079	18.7077	Curly_hair	Low_PulseRate	High_Conduc
...	...	...	...	...	...	...	...	...	...
670	134065	95	3.0788	6.7874	14.5136	26.9154	No_hair	High_PulseRate	Low_Conduc
671	182597	23	1.2767	4.9695	8.8617	18.6980	No_hair	Low_PulseRate	High_Conduc
672	156972	36	2.3799	5.3240	11.5026	23.6280	Wavy_hair	Medium_PulseRate	Normal_Condu
673	172428	59	2.6147	5.3841	11.1521	20.8031	Straight_hair	Medium_PulseRate	Normal_Condu
674	158058	100	3.4924	6.7524	12.9776	26.8441	No_hair	High_PulseRate	Low_Conduc

675 rows × 14 columns

```
In [25]: df["apnea_Severity"] = df["apnea_Severity"].map(
             {'High_Severity': 2, 'Medium_Severity': 1, 'Low_Severity': 0})
         df["heart_rate"] = df["heart_rate"].map(
             {'High_PulseRate': 2, 'Medium_PulseRate': 1, 'Low_PulseRate': 0})
         df["skin_conductance"] = df["skin_conductance"].map(
             {'High_Conductance': 2, 'Normal_Conductance': 1, 'Low_Conductance': 0})
         df["skin_temperature"] = df["skin_temperature"].map(
             {'Low_Temperature': 2, 'Normal_Temperature': 1, 'Fever': 0})
         df["cortisol_level"] = df["cortisol_level"].map(
             {'Above_AverageCL': 2, 'AverageCL': 1, 'Below_AverageCL': 0})
         df["Systolic_BP"] = df["Systolic_BP"].map(
             {'Range1_LowSystolic': 2, 'Range2_LowSystolic': 1, 'Range3_LowSystolic': 0})
         df["Diastolic_BP"] = df["Diastolic_BP"].map(
             {'VerylowDiSystolic': 2, 'NormalDiSystolic': 1, 'LowDiSystolic': 0})
```

```
In [26]: df
```

Out[26]:

	Patinet_ID	EEG_Signal_Amplitude	EEG_Delta_band	EEG_Theta_band	EEG_Alpha_band	EEG_Beta_band	Hair_Phenotype	heart_rate	skin_conductance	s
0	180203	56	2.4786	5.5748	11.7319	23.9909	Curly_hair	1	1	
1	152268	97	3.2531	6.4658	13.1411	28.3962	Curly_hair	2	0	
2	157399	83	3.6325	6.0053	13.6766	26.0487	Wavy_hair	2	0	
3	131849	58	2.9477	5.5462	10.3739	22.0865	Straight_hair	1	1	
4	164593	22	1.9366	4.3574	8.9079	18.7077	Curly_hair	0	2	
...	...	...	...	...	...	...	...	...	...	...
670	134065	95	3.0788	6.7874	14.5136	26.9154	No_hair	2	0	
671	182597	23	1.2767	4.9695	8.8617	18.6980	No_hair	0	2	
672	156972	36	2.3799	5.3240	11.5026	23.6280	Wavy_hair	1	1	
673	172428	59	2.6147	5.3841	11.1521	20.8031	Straight_hair	1	1	
674	158058	100	3.4924	6.7524	12.9776	26.8441	No_hair	2	0	

675 rows × 14 columns

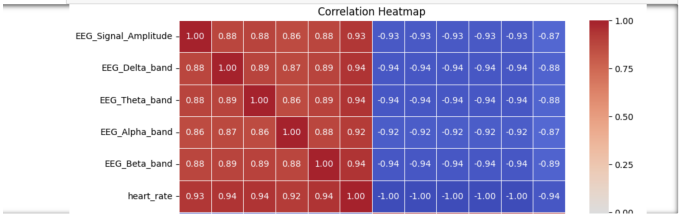
```
In [27]: # Select only the specified columns in the DataFrame
         df = df[["EEG_Signal_Amplitude", "EEG_Delta_band", "EEG_Theta_band", "EEG_Alpha_band", "EEG_Beta_band", "heart_rate",
                  "skin_conductance", "skin_temperature", "cortisol_level", "Systolic_BP", "Diastolic_BP", "apnea_Severity"]]
```

```
In [28]: df
```

```
In [29]: # Imported Library's
         import torch
         import torch.nn as nn
         import torch.optim as optim
         from torch.utils.data import DataLoader, TensorDataset
         import pandas as pd
         from sklearn.model_selection import train_test_split
         from sklearn.preprocessing import StandardScaler
         from sklearn.metrics import accuracy_score
         import warnings
         warnings.filterwarnings("ignore")
```

```
In [30]: # Handle missing values if any
         df.dropna(inplace=True)
         df
```

```
In [31]: correlation_matrix = df.corr()
         print(correlation_matrix)
         plt.figure(figsize=(10, 8))
         sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f", linewidths=.5)
         plt.title("Correlation Heatmap")
         plt.show()
```




```
In [32]: # Split the dataset into features and target
         X = df.drop(columns=['apnea_Severity']).values
         y = df['apnea_Severity'].values
```

```
In [33]: # Perform feature scaling
         scaler = StandardScaler()
         X_scaled = scaler.fit_transform(X)
```

```
In [34]: # Split the dataset into training and testing sets
         X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)
```

```
In [35]: # Convert data to PyTorch tensors
         X_train_tensor = torch.tensor(X_train, dtype=torch.float32)
         X_test_tensor = torch.tensor(X_test, dtype=torch.float32)
         y_train_tensor = torch.tensor(y_train, dtype=torch.long)
         y_test_tensor = torch.tensor(y_test, dtype=torch.long)
```

```
In [36]: # Defineing the neural network model
         class Classifier(nn.Module):
             def __init__(self, input_size, output_size):
                 super(Classifier, self).__init__()
                 self.fc1 = nn.Linear(input_size, 128)
                 self.fc2 = nn.Linear(128, 64)
                 self.fc3 = nn.Linear(64, output_size)
                 self.dropout = nn.Dropout(0.5)  # Adding dropout for regularization

             def forward(self, x):
                 x = torch.relu(self.fc1(x))
                 x = self.dropout(x)
                 x = torch.relu(self.fc2(x))
                 x = self.dropout(x)
                 x = self.fc3(x)
                 return x
```

```
In [37]: # Instantiate the model
         input_size = X_train.shape[1]
         output_size = len(df['apnea_Severity'].unique())
         model = Classifier(input_size, output_size)
```

```
In [38]: # Define the loss function and optimizer, and adjust the learning rate
         criterion = nn.CrossEntropyLoss()
         optimizer = optim.Adam(model.parameters(), lr=0.001)  # Adjusted learning rate
```

```
In [39]: # Training loop
         num_epochs = 100
         for epoch in range(num_epochs):
             optimizer.zero_grad()
             outputs = model(X_train_tensor)
             loss = criterion(outputs, y_train_tensor)
             loss.backward()
             optimizer.step()
             if (epoch+1) % 10 == 0:
                 print(f'Epoch [{epoch+1}/{num_epochs}], Loss: {loss.item():.4f}')
```

```
Epoch [10/100], Loss: 0.8643
Epoch [20/100], Loss: 0.6109
Epoch [30/100], Loss: 0.4788
Epoch [40/100], Loss: 0.4000
Epoch [50/100], Loss: 0.3173
Epoch [60/100], Loss: 0.3046
Epoch [70/100], Loss: 0.2790
Epoch [80/100], Loss: 0.2583
```

```
In [40]: # Evaluate the model
         model.eval()
         with torch.no_grad():
             y_pred = torch.argmax(model(X_test_tensor), dim=1).numpy()
             accuracy = accuracy_score(y_test, y_pred)
             print("Accuracy on test set:", accuracy)
```

```
Accuracy on test set: 0.9407407407407408
```

```
In [41]: # Make predictions on new data
         new_data = [[97, 3.2531, 6.4658, 13.1411, 28.3962, 2, 0, 0, 0, 0, 0]]
         new_data_scaled = scaler.transform(new_data)
         new_data_tensor = torch.tensor(new_data_scaled, dtype=torch.float32)
         with torch.no_grad():
             prediction = torch.argmax(model(new_data_tensor)).item()
             print('Predicted apnea severity:', prediction)
```

```
Predicted apnea severity: 0
```

```
In [42]: new_data = [[58, 2.9477, 5.5462, 10.3739, 22.0865, 1, 1, 1, 1, 1, 1]]
         new_data_scaled = scaler.transform(new_data)
         new_data_tensor = torch.tensor(new_data_scaled, dtype=torch.float32)
         with torch.no_grad():
             prediction = torch.argmax(model(new_data_tensor)).item()
             print('Predicted apnea severity:', prediction)
```

```
Predicted apnea severity: 1
```

```
In [43]: new_data = [[22, 1.9366, 4.3574, 8.9079, 18.7077, 0, 2, 2, 2, 2, 2]]
         new_data_scaled = scaler.transform(new_data)
         new_data_tensor = torch.tensor(new_data_scaled, dtype=torch.float32)
         with torch.no_grad():
             prediction = torch.argmax(model(new_data_tensor)).item()
             print('Predicted apnea severity:', prediction)
```

```
Predicted apnea severity: 2
```

REFERENCES

[1] Mostafa M. Moussa, Yahya Alzaabi, and Ahsan H. Khandoker. Explainable computer-aided detection of obstructive sleep apnea and depression. volume 10, 2022.

[2] Hang Liu, Shaowei Cui, Xiaohui Zhao, and Fengyu Cong. Detection of obstructive sleep apnea from single-channel ecg signals using a cnn-transformer architecture. volume 82, page 104581, 2023.

[3] Frank Mierzwa and Kim Chantala. Connecting and linking neurocognitive, digital phenotyping, physiologic, psychophysical, neuroimaging, genomic, sensor data.

[4] N. Mehta and A. Pandit. Concurrence of big data analytics and healthcare: a systematic review. volume 114, pages 57–65, 2018.

[5] C. Auffray et al. Making sense of big data in health research: towards an eu action plan. volume 8, page 71, 2016.