Sleep Disorder Prediction

By
Sarkis Shil-Gevorkyan
and
German Wong

Introduction

Having enough sleep is very important for one's overall health and well-being. It is what allows your body to rest and recharge so that you are energized and ready to start the day. A person's overall health and lifestyle do in fact have a deep connection with one's quality of sleep, and some negative habits in one's life may even affect their sleep quality and potentially lead to sleeping disorders such as Insomnia and Sleep Apnea. Using the Sleep Health and Lifestyle dataset, obtained on Kaggle, we will **create a model that can predict whether or not a person has a possible sleeping disorder based on their recorded health and lifestyle data.** The dataset includes 12 features, which are a participant's: Gender, Age, Occupation, Hours of Sleep, Quality of Sleep (subjectively rated from 1 to 10), Physical Activity level (minutes/day), Stress Level(also rated from 1 to 10 by individual), BMI category (normal, overweight, obese), Blood Pressure (systolic/diastolic), Heart Rate (beats per minute), Number of Daily Steps, as well as the target, whether the participant has a sleeping disorder. (None, Insomnia, Sleep Apnea)

Data Preprocessing

Before implementing our machine models, we start by preparing the data. Originally, the dataset had 374 tuples, which is very little data to work with. To have more data, we implemented a method called bootstrapping, which is a statistical technique that repeatedly resamples a single dataset with replacement and creates many simulated samples. We decided to create 9 separate bootstrapped datasets (that number can vary depending on the user) and then concatenated them all, using the pandas function .concat(...), into a single dataset. After bootstrapping, our dataset resulted in having 3366 tuples. We then first checked if we had missing values, that are null or NA, which we have done so using the functions: **.isnull()** and **.isna()**. Fortunately, we did not have any missing values in our dataset. In the occupations column, we noticed some instances said "Sales Representative" and "Software Engineer," we then renamed those to "Salesperson" and "Engineer" respectively, to combine with the others. Similarly, in the"BMI Category" column, some instances said "Normal Weight", so we then renamed those to "Normal."

```
# Number of bootstrapped datasets to create
num_bootstraps = 9

# Initialize an empty list to store bootstrapped datasets
bootstrapped_datasets = []

# Randomly sample rows with replacement to create a bootstrapped dataset
for x in range(num_bootstraps):
    bootstrapped_sample = df.sample(n = len(df), replace=True)

    # Append the bootstrapped dataset to the list
    bootstrapped_datasets.append(bootstrapped_sample)

#Save the bootstrapped datasets to separate CSV files
for i, bootstrapped_data in enumerate(bootstrapped_datasets):
    bootstrapped_data.to_csv(f'bootstrapped_dataset_{i}.csv', index=False)
```

We then did Feature Engineering for our dataset, starting by replacing categorical variables with numerical data, which we've done so by using a label encoder, **LabelEncoder()**, through the Scikit Learn library, which transformed categorical variables into

```
from sklearn.preprocessing import LabelEncoder

#Using LabelEncoder to Convert Occupations, Gender and BMI to Numerical Values
LE = LabelEncoder()
df["Occupation"] = LE.fit_transform(df["Occupation"])
df["Gender"] = LE.fit_transform(df["Gender"])
df["BMI Category"] = LE.fit_transform(df["BMI Category"])

#Converting Sleep Disorder to Numerical Values
df = df.replace(['None'], 0)
df = df.replace(['Insomnia'], 1)
df = df.replace(['Sleep Apnea'], 2)
```

numerical values, in the “Occupations”, “BMI Category”, “Gender” and “Sleep Disorder” features. The “Blood Pressure” was initially a string type, in the format of a fraction,(systolic/diastolic) To convert them to integer values, we created a new dataframe splitting the blood pressure values into two separate columns, “Upper_Bloodpressure_Value” and “Lower_Bloodpressure_Value,” and changed the type of the values to floating-point values. In order to know what the right features are for our models, we used a Random Forest Classifier to describe which features would be relevant, using the built-in function

```
#Splitting Blood Pressure to Upper Value and Lower Value
df = pd.concat([df, df['Blood Pressure'].str.split('/', expand=True)], axis=1).drop('Blood Pressure', axis=1)
df = df.rename(columns={0: 'BloodPressure_Upper_Value', 1: 'BloodPressure_Lower_Value'})

#Changing Blood Pressure Upper and Lower Values to type float
df['BloodPressure_Upper_Value'] = df['BloodPressure_Upper_Value'].astype(float)
df['BloodPressure_Lower_Value'] = df['BloodPressure_Lower_Value'].astype(float)
```

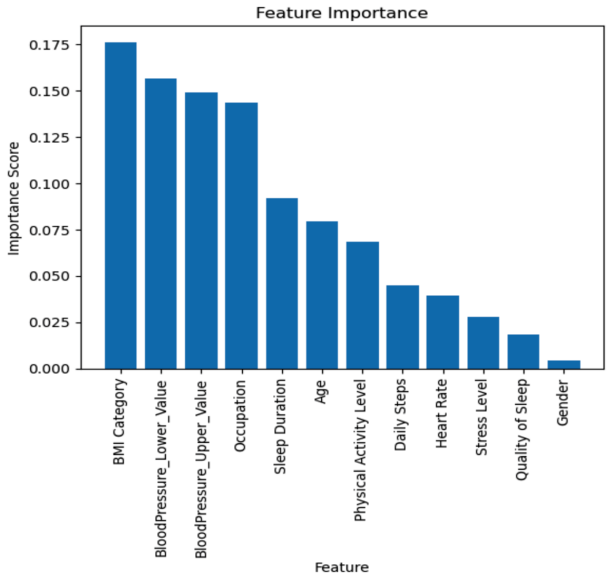
“feature_importance_”. The way its feature importance is calculated is through impurity importance, which is the sum of the impurity decrease of all trees when a variable is selected to split a node. Hence if the decrease is low, then the feature is not important. Once the list of importance scores is calculated, it is then sorted and then plotted onto a barchart. According to the Feature Importance Bar Chart, the features, “Gender,” “Quality of Sleep,” “Stress Level” and “Daily Steps” have low importance scores, therefore being the least important features to be used.

```
#Using random forest to find feature importance
from sklearn.ensemble import RandomForestClassifier

clf = RandomForestClassifier()

#Trains the model
clf.fit(X, Y)

#Calculates the feature importance
importances = clf.feature_importances_

#Prints the feature importance
print(importances)
names = X.keys()
```

```
[0.00440743 0.08943961 0.13156528 0.10179909 0.020447   0.05142642
 0.02323044 0.1789966  0.04925721 0.04971148 0.1338224  0.16589705]
```

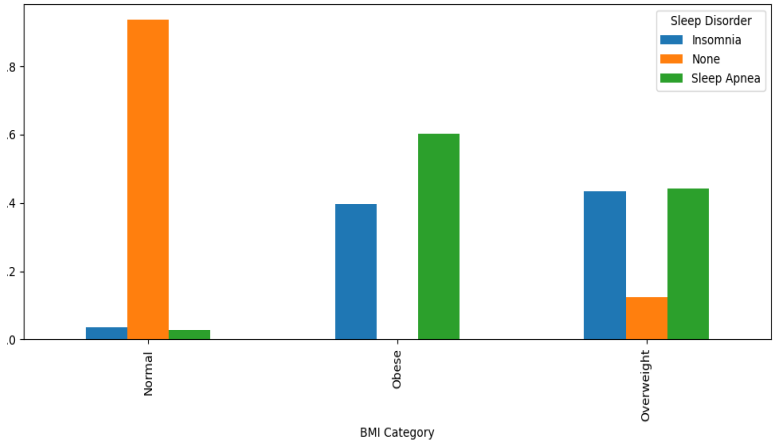
```
#Sorting Features and Feature names
indices = np.argsort(importances)[::-1]
sorted_feature_names = [names[i] for i in indices]
sported_importances = importances[indices]
```

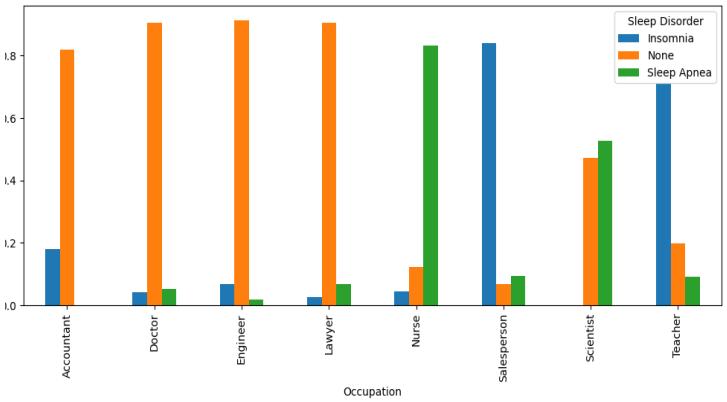
```
#Plotting Feature Importance in a barchart
plt.bar(range(len(sorted_importances)), sorted_importances)
plt.xlabel('Feature')
plt.ylabel('Importance Score')
plt.title('Feature Importance')
plt.xticks(range(len(sorted_importances)), sorted_feature_names, rotation='vertical')
plt.show()
```



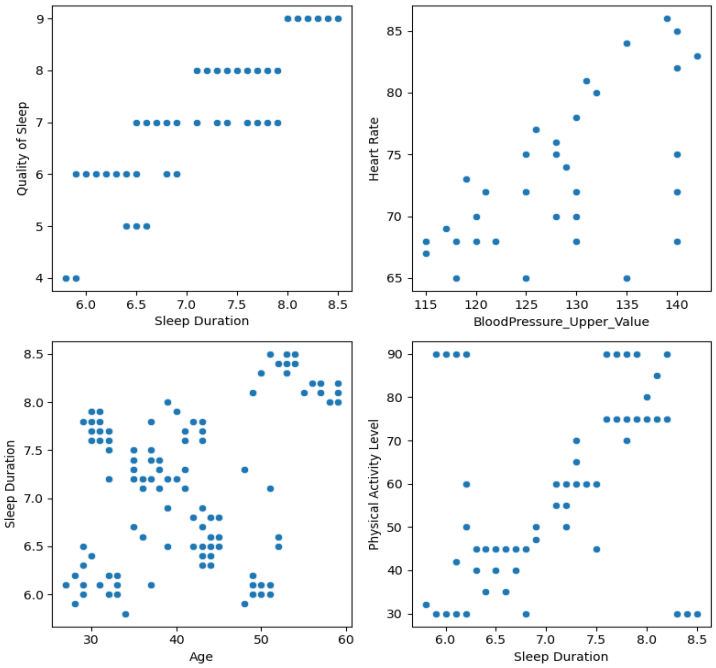
Data Visualization

After cleaning up the data, we then performed visualization of the data in order to see if there is correlation between the features and with the target. The following bar chart is to visualize the connection between the target, “Sleep Disorder,” with “BMI Category” and “Occupations.” Looking at the BMI barchart, we can see that those considered “Normal” are less likely to have a sleeping disorder, whereas those that are all in the category of overweight and obese have a much higher chance of having one. Now looking at these occupations barchart, we can see that a lot of “Teachers” and “Salespersons” happen to have Insomnia, whereas a very large number of “Nurses” have Sleep Apnea.



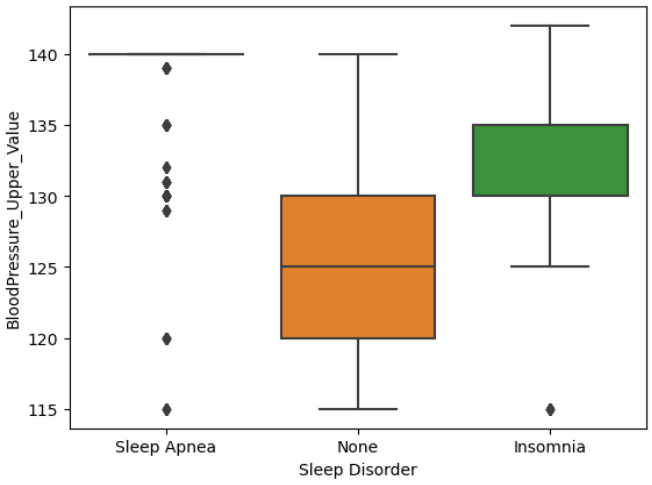


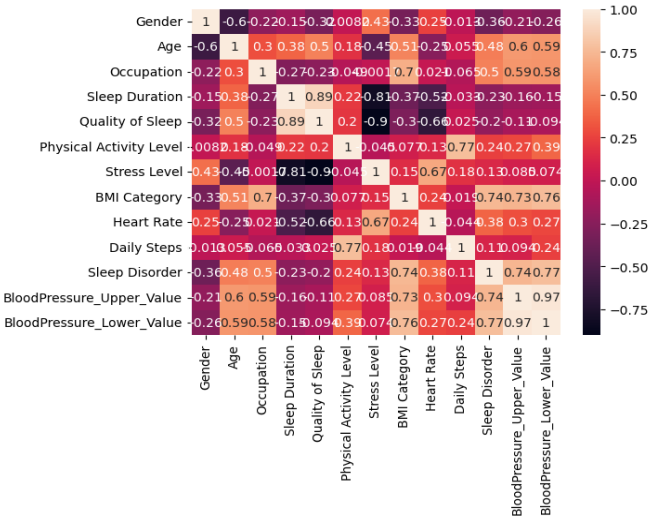
Another form of visualization, shown are scatterplots, which is where data points are plotted to show a correlation between two variables. Four scatter plots were created to show a connection, or correlation between two features. One of the scatterplot was plotted to show a connection between Sleep Quality and Sleep Duration, which showed a positive correlation suggesting that more sleep leads to a greater quality of sleep. Another scatter plot showed a relationship between Heart Rate and Blood Pressure, (BloodPressure_Upper_Value) which also showed a positive correlation, suggesting that a higher heart rate can lead to an increase in blood pressure. The third scatterplot shows a positive correlation between age and sleep duration, showing that older people, ages 50 and more, tend to sleep longer hours, whereas younger people get less than 8 hours of sleep. And The last scatterplot shows a relationship between Sleep Duration and Physical Activity Level, which shows a positive correlation, suggesting that individuals that exercise regularly, tend to also sleep more.



Another visualization, shown is the boxplot, where it shows the distribution of the data, also including any outliers, as well as the mean, median, and interquartile range. The following boxplot shows the relationship between Blood Pressure and Sleeping Disorders. First Looking at the None class, we can see that most of the individuals have their blood pressure range around the middle part. Looking at the Insomnia class, we can see that the majority have a higher blood pressure than those that don't have any sleeping disorders. And lastly, looking at Sleep Apnea, almost every individual has a very high blood pressure, and those that don't have a high blood pressure are just considered as outliers. A heatmap has also been made to show correlation of every feature towards one another. Any feature that has a

correlation closer to 1.00, indicates a strong positive correlation, any two features that have a correlation closer to -1.00 ,indicates a strong negative correlation, and anything else that is in the middle is considered to have no correlation.





Related Work

Sleep disorders, including conditions like sleep apnea and insomnia, significantly impact well-being, from cardiovascular issues to cognitive impairments. The critical need for precise diagnosis in sleep medicine has prompted diverse efforts, with recent focus on leveraging machine learning for classification. This exploration builds on previous work, investigating methodologies in sleep disorder classification, particularly for sleep apnea and insomnia. Examining various sources, it uncovers diverse strategies in the relentless pursuit of accurate diagnoses.

In the ScienceDirect paper "Explainable machine learning for sleep apnea prediction," the focus is on a binary classification problem related to sleep apnea. The study uses labels (1 for presence, 0 for absence) to indicate apnea episodes in recorded tuples. Emphasizing efficiency and low computational demands, the authors employ machine learning models from the scikit-learn library in Python. They partition data into training and testing sets, forming a concise yet comprehensive approach to exploring machine learning's predictive prowess in sleep apnea detection.

In "Automated Identification of Sleep Disorder Types," the study utilizes the LIME library in Python for explainable predictions from PSG data. LIME, known for simplicity, aids interpretability as a model-agnostic tool. Various Python models, including a Random Forest classifier, are assessed for classification performance using LIME.

Both methodologies are rooted in a common foundation—the utilization of MIT-BIH-PSG data. This dataset incorporates diverse signals such as ECG, invasive blood pressure, EEG, and respiration signals, complemented by annotations from healthcare professionals. These annotations play a pivotal

role in defining classes for classification tasks, specifically identifying the occurrence or absence of apnea events.

Decision Tree Modeling

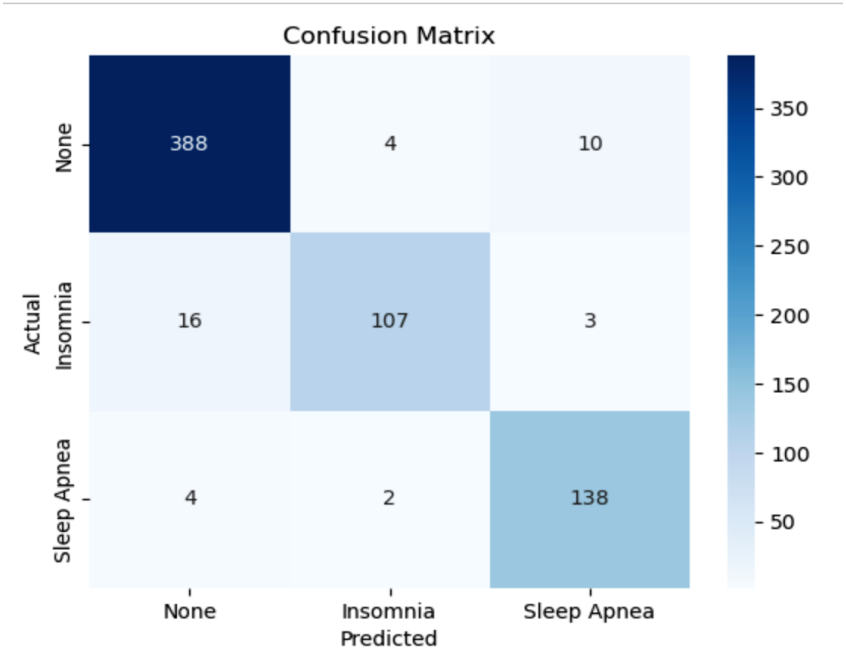
One of the algorithms used was the Decision Tree, which was implemented by German Wong. After assigning the input, X, and output, Y, and after splitting training and test set 80/20. Gridsearch was used to check which metrics worked the best. Therefore the parameters were declared with Max_depth, min_sample_split and criterion. Then the Decision Tree model was made, with the DecisionTreeClassifier(), from sci-kit learn library. Then gridsearch was run, and it resulted with the best Minimum split being 4, max level being 11, and the best algorithm being "entropy, " which measures the uncertainty. After running gridsearch, the Decision Tree was then trained, with the 80% of training set, with the best parameters that was provided.

```
#Decision Tree Model
#Test Entropy and Gini Calculations
#Applying GridSearch to check which metrics work best

parameters = {
              'max_depth': [3, 4, 5, 6, 7, 9, 11],
              'min_samples_split': [2, 3, 4, 5, 6, 7],
              'criterion': ['entropy', 'gini']
             }

#Decision Tree
model = DecisionTreeClassifier()
gridDecisionTree = RandomizedSearchCV(model, parameters, cv = 3, n_jobs = -1)
gridDecisionTree.fit(X_train, y_train)

print('Min Split: ', gridDecisionTree.best_estimator_.min_samples_split)
print('Max Nvl: ', gridDecisionTree.best_estimator_.max_depth)
print('Algorithm: ', gridDecisionTree.best_estimator_.criterion)
print('Score: ', gridDecisionTree.best_score_)
```

```
Min Split:  4
Max Nvl:  11
Algorithm:  entropy
Score:  0.9322658951848896
```

```
: #Running Decision Tree
  tree_clf = DecisionTreeClassifier(criterion = "entropy", min_samples_split = 4, max_depth = 11)
  tree_clf.fit(X_train, y_train)
```

```
: DecisionTreeClassifier(criterion='entropy', max_depth=11, min_samples_split=4)
```



Decision Tree Evaluation

To assess the efficacy of the Decision Trees model, a 3x3 confusion matrix was employed, wherein each cell signifies the concordance between actual and predicted values. The degree of shading denotes the frequency of concordant class assignments. Review of the Confusion Matrix and Classification Report attests to the model's commendable performance across distinct classes. Precision, indicative of accurate predictions, attained noteworthy values: 0.95 for "None," 0.99 for Insomnia, and 0.90 for Sleep Apnea. Similarly, the Recall metric, reflecting the proportion of accurately identified relevant data points, exhibited commendable results: 0.97 for "None," 0.83 for Insomnia, and 0.97 for Sleep Apnea. In aggregate, the Decision Trees model demonstrated exemplary performance, boasting an overall accuracy of 95%.

```
In [51]: print('\nClassification Report:')
         print(classification_report(y_test, y_pred))
```

```
Classification Report:
              precision    recall  f1-score   support

           0       0.95      0.97      0.96       402
           1       0.99      0.83      0.91       126
           2       0.90      0.97      0.94       144

    accuracy                           0.95       672
   macro avg       0.95      0.93      0.93       672
weighted avg       0.95      0.95      0.95       672
```

Neural Network Modeling

One of the algorithms used was the Artificial Neural Network, which was implemented by Sarkis Shil-Gevorkyan. To begin, the input features, X, needed to be normalized before feeding into the algorithm, therefore we scaled it using the MinMaxScaler function in the SciKit Learn Library. Then, the target, Y, also needed to be transformed into a 2D array, which we first used the OneHotEncoder function in the Scikit Library, and to then use fit_transform() function to create the array. After transforming the X, Y, it was then split 80/20 for training and testing.

```
from sklearn.preprocessing import MinMaxScaler
sc = MinMaxScaler()

#Normalizes the features
X = sc.fit_transform(X)
X.shape
```

```
(3353, 7)
```

```
from sklearn.preprocessing import OneHotEncoder

#Transforming the Categorical Data into Numerical, and turning it into a 2D Array
ohe = OneHotEncoder()
Y = ohe.fit_transform(Y).toarray()
Y.shape
```

```
(3353, 3)
```

After transforming the input features and target, then begins creating the Neural Network model. To begin, since the model is a linear stack of layers, we assign the model to call the Sequential() function, in the Keras library. The Neural Network's input layer has 7 nodes, meaning that there are 7 features as inputs. There are also two hidden layers, with each layer being connected with 300 neurons. Both hidden layers do also have the "relu" activation function, which is a function, $f(x) = \max(0,x)$, which sets all negative values to 0. To help prevent any overfitting in the model, a dropout function was added, with a value of 0.2 given, which gave the best results. The output layer consists of 3 nodes, being the 3 classes: None, Insomnia and Sleep Apnea, and also uses softmax function, which converts the vectors into probability distributions.

```
: #Creating the Neural Network Model
model = Sequential()
# The first hidden layer will have 300 neurons
model.add(Dense(300, input_shape=(7,))) #There are 7 features being inputted
model.add(Activation('relu')) #f(x) = max(0, x), which sets all negative values to 0
model.add(Dropout(0.2)) # Used to Prevent Model from overfitting

# The second hidden layer will also have 300 neurons
model.add(Dense(300))
model.add(Activation('relu'))
model.add(Dropout(0.2))

# The final layer is output that has 3 classes
model.add(Dense(3))
model.add(Activation('softmax')) #Function that converts a vector into a probability distribution

# Summarizes the built model
model.summary()
```

```
Model: "sequential_14"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_42 (Dense)            (None, 300)               2400

 activation_42 (Activation)  (None, 300)               0

 dropout_28 (Dropout)        (None, 300)               0

 dense_43 (Dense)            (None, 300)               90300
```

After the model was created, it then had to be compiled, using the .compile() function, where the parameters consisted of “”categorical_crossentropy” to calculate the the loss and also the “adam” optimizer, which updates the weight of the neural network during training. The model was then trained with the training set of data. The training data consisted of batches of 200 tuples, with 15 epochs, meaning 200 tuples were being trained at a time 15 times. Looking at the output of the training, the first batch had a very high loss, but as it kept going to the next batch, we notice that the loss is decreasing and the accuracy is increasing. Once the accuracy and loss began to remain on a constant level, that was when it became best to stop.

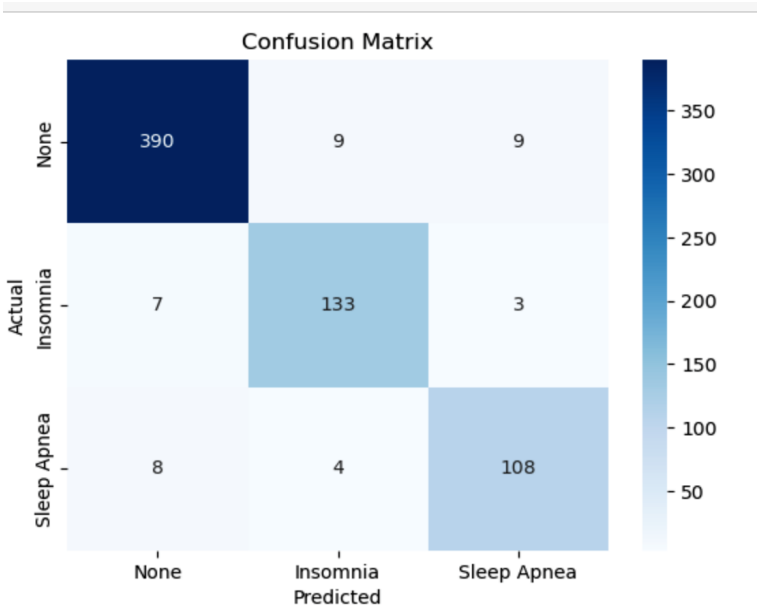
```
: #Compiling the Model
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
```

```
: #Training the Model
history = model.fit(X_train, y_train, validation_data=(X_test, y_test), batch_size = 200, epochs = 15, verbose=1)
```

```
Epoch 1/15
14/14 [==============================] - 1s 18ms/step - loss: 0.8427 - accuracy: 0.6853 - val_loss: 0.5528 - val_ac
curacy: 0.8689
Epoch 2/15
14/14 [==============================] - 0s 8ms/step - loss: 0.5284 - accuracy: 0.8494 - val_loss: 0.4242 - val_acc
uracy: 0.8703
Epoch 3/15
14/14 [==============================] - 0s 8ms/step - loss: 0.4526 - accuracy: 0.8639 - val_loss: 0.3778 - val_acc
uracy: 0.8957
Epoch 4/15
14/14 [==============================] - 0s 9ms/step - loss: 0.4130 - accuracy: 0.8799 - val_loss: 0.3443 - val_acc
uracy: 0.8957
Epoch 5/15
14/14 [==============================] - 0s 8ms/step - loss: 0.3798 - accuracy: 0.8863 - val_loss: 0.3215 - val_acc
uracy: 0.8972
Epoch 6/15
14/14 [==============================] - 0s 8ms/step - loss: 0.3596 - accuracy: 0.8893 - val_loss: 0.3025 - val_acc
uracy: 0.8987
Epoch 7/15
14/14 [==============================] - 0s 8ms/step - loss: 0.3349 - accuracy: 0.9004 - val_loss: 0.2940 - val_acc
uracy: 0.9061
Epoch 8/15
14/14 [==============================] - 0s 7ms/step - loss: 0.3252 - accuracy: 0.9098 - val_loss: 0.2782 - val_acc
uracy: 0.9225
Epoch 9/15
14/14 [==============================] - 0s 7ms/step - loss: 0.3118 - accuracy: 0.9176 - val_loss: 0.2678 - val_acc
uracy: 0.9359
Epoch 10/15
```

Neural Network Evaluation

To evaluate the Neural Network Model, a 3x3 confusion matrix was used, where each square is the amount of data points where the actual and predicted are the same. The Darker it is, the more of the same matching classes there are between the Actual and predicted. Looking at the Confusion Matrix and Classification Report, we can see that the model performed very well for each class. The Precision, which is the measure of correct predictions, did very well, with a 0.96 for “None”, 0.91 for Insomnia, and a 0.90 for Sleep Apnea. The recall, which is the percentage of relevant data points that were correctly identified, did also do fairly well, with a 0.96 for “None”, 0.93 for Insomnia, and a 0.90 for Sleep Apnea. And overall, the Neural Network Model performed very well with an accuracy of 94%



Classification Report:

	precision	recall	f1-score	support
None	0.96	0.96	0.96	408
Insomnia	0.91	0.93	0.92	143
Sleep Apnea	0.90	0.90	0.90	120
accuracy			0.94	671
macro avg	0.92	0.93	0.93	671
weighted avg	0.94	0.94	0.94	671

Conclusion

In a comprehensive evaluation of the Decision Trees and Neural Network models, both demonstrate commendable accuracy and proficiency in handling sleep disorder classification. The Decision Trees model reveals exemplary precision, underscoring its ability to make accurate predictions with values of 0.95 for "None," an impressive 0.99 for Insomnia, and a robust 0.90 for Sleep Apnea. The Recall metric, indicating the model's capability to identify relevant data points, similarly displays strong performance, with values of 0.97 for "None," 0.83 for Insomnia, and 0.97 for Sleep Apnea. The cumulative outcome underscores the Decision Trees model's outstanding overall accuracy, which stands at an impressive 95%.

Conversely, the Neural Network model also exhibits robust precision, showcasing its capacity for accurate predictions with values of 0.96 for "None," 0.91 for Insomnia, and 0.90 for Sleep Apnea. The Recall metric corroborates these findings, with values of 0.96 for "None," 0.93 for Insomnia, and 0.90 for Sleep Apnea. The Neural Network model delivers an overall accuracy of 94%, indicating a high level of proficiency in sleep disorder classification. This performance attests to the model's effectiveness in capturing and learning from complex patterns within the data.

In summary, both the Decision Trees and Neural Network models prove to be robust contenders in the realm of sleep disorder classification. While the Decision Trees model holds a marginal lead in overall accuracy due to its performance relative to the size of our dataset, both models demonstrate great precision and recall across distinct classes, underscoring their utility in addressing the nuanced challenges associated with sleep disorder diagnoses. The choice between these models may ultimately hinge on specific application requirements and the interpretability preferences of practitioners in the field.

References:

- https://www.kaggle.com/datasets/uom190346a/sleep-health-and-lifestyle-dataset
- ScienceDirect - Explainable machine learning for sleep apnea prediction
- Automated Identification of Sleep Disorder Types Using Triplet Half-Band Filter and Ensemble MAchine Learning Techniques with EE