Capstone Project Report

Done by: Noor Alsaud,

Date: 28-12-2024

Project Overview

This project investigates the relationship between sleep metrics and stress levels using physiological data from the **SaYoPillow Dataset**. Stress levels are predicted using machine learning models to identify key contributing factors, such as snoring range, heart rate, and blood oxygen levels. The project aims to offer actionable insights into managing stress through improved sleep hygiene. Achieving a prediction accuracy of 98% using Random Forest and Gradient Boosting models, the study provides robust evidence for the importance of physiological data in stress management.

Problem Statement: Can stress levels during sleep be accurately predicted using physiological data, and which factors play the most significant role in these predictions?

Significance: Stress management is critical for mental and physical well-being. Understanding the role of sleep and related metrics offers opportunities for preventive measures and improved health outcomes.

Description of Input Data

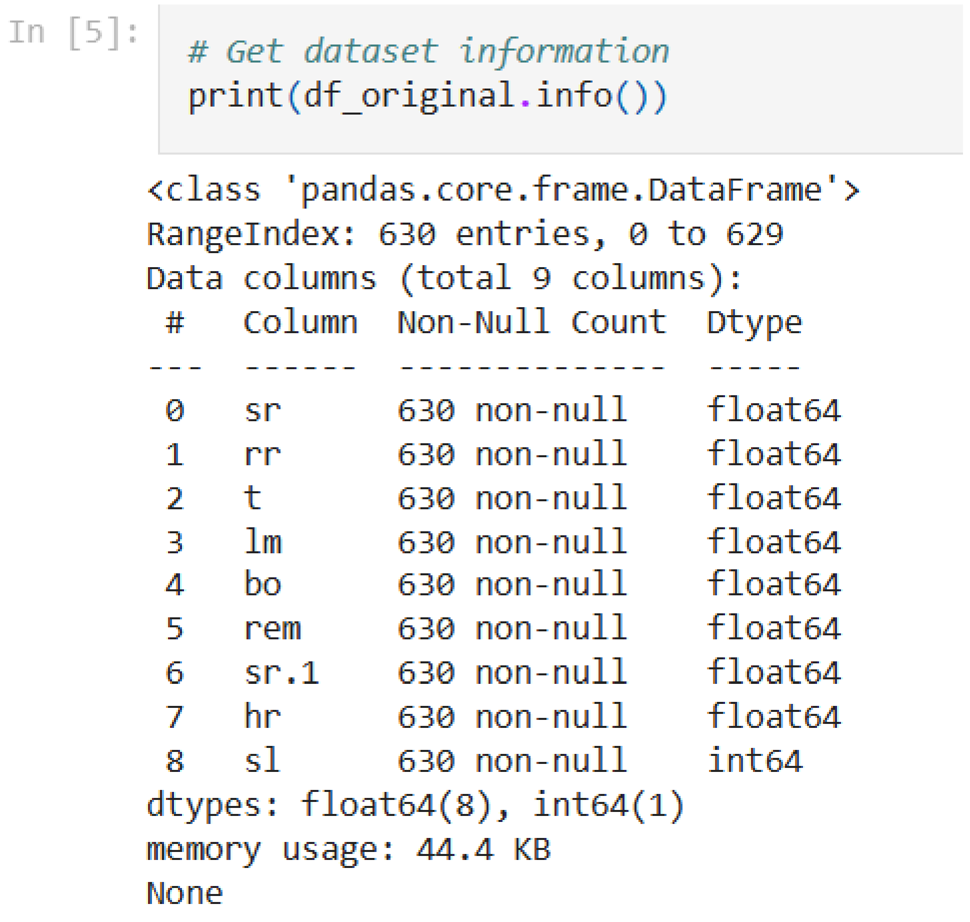
Dataset Source: The dataset was obtained from Kaggle and is part of the **SaYoPillow Framework**. It contains sleep and physiological metrics collected during sleep sessions.

Data Format:

- CSV file with 9 input features and 1 target variable.
- Includes 630 observations with no missing values.

Variables:

- Input Features:
    - Snoring Range (numeric): Intensity of snoring.
    - Respiration Rate (numeric): Breaths per minute.
    - Body Temperature (numeric): Measured in Celsius.
    - Limb Movement Rate (numeric): Frequency of limb movements.
    - Blood Oxygen Level (numeric): Percentage of oxygen saturation.
    - REM Eye Movement (numeric): Frequency of rapid eye movements.
    - Hours of Sleep (numeric): Total hours of sleep.
    - Heart Rate (numeric): Measured in beats per minute (bpm).
- **Target Variable:** Stress Level (categorical): 5 levels (0: low, 1: medium low, 2: medium, 3: medium high, 4: high).



```
In [5]:  # Get dataset information
         print(df_original.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 630 entries, 0 to 629
Data columns (total 9 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   sr      630 non-null    float64
 1   rr      630 non-null    float64
 2   t       630 non-null    float64
 3   lm      630 non-null    float64
 4   bo      630 non-null    float64
 5   rem     630 non-null    float64
 6   sr.1    630 non-null    float64
 7   hr      630 non-null    float64
 8   sl      630 non-null    int64  
dtypes: float64(8), int64(1)
memory usage: 44.4 KB
None
```

Figure 1 description of the Columns in the dataset before cleaning

Strategy for Solving the Problem

Approach:

1. Conduct **Exploratory Data Analysis (EDA)** to identify patterns, trends, and relationships in the data.
2. Preprocess the data to handle scaling, feature engineering, and encoding.
3. Use machine learning models (Random Forest and Gradient Boosting) to predict stress levels.
4. Evaluate the models using appropriate metrics.

Rationale:

- Random Forest and Gradient Boosting are well-suited for classification tasks and provide feature importance insights.
- Cross-validation ensures the robustness of the models.

Discussion of the Expected Solution

Proposed Workflow:

1. Data Exploration: Understand distributions and correlations.
2. Data Preprocessing: Scale features and handle class balance.
3. Model Training: Train machine learning models.
4. Model Validation: Use cross-validation and hyperparameter tuning.
5. Results Interpretation: Analyze feature importance and performance metrics.

Expected Outcome:

- Achieve high accuracy in predicting stress levels.
- Identify the most influential features for stress prediction, such as snoring range and blood oxygen level.

Metrics with Justification

Evaluation Metrics:

- **Accuracy:** Measures the overall correctness of predictions.
- **Precision, Recall, and F1-Score:** Provide deeper insights into model performance for each stress level.
- **Cross-Validation Mean Accuracy and Standard Deviation:** Ensure robustness and generalizability.

Justification: These metrics are well-suited for multi-class classification problems and provide a comprehensive evaluation of model performance:

Justification for Model Performance

In this project, three machine learning models—Random Forest, Gradient Boosting, and K-Nearest Neighbors (KNN)—were applied to predict stress levels based on sleep and physiological data. Each model performed differently due to variations in their underlying principles, assumptions, and suitability for the dataset.

1. Random Forest Classifier

- **Why It Worked Well:**
  - Random Forest leverages multiple decision trees and averages their results, making it robust to overfitting and highly effective at capturing non-linear relationships in the data.
  - The diverse feature set in this dataset (e.g., snoring range, blood oxygen level) benefits from Random Forest's ability to model complex interactions between variables.
- **Factors Influencing Performance:**
  - Random Forest handles both categorical and continuous data well without requiring extensive preprocessing.
  - The model's built-in feature importance analysis helped identify key predictors like snoring range and blood oxygen level, aligning with domain knowledge.

- **Performance:** Achieved a high accuracy of 98% and showed consistent results across cross-validation folds, highlighting its reliability.

2. Gradient Boosting Classifier

- **Why It Worked Well:**
    - Gradient Boosting iteratively improves predictions by correcting the errors of previous models, making it particularly effective for complex datasets.
    - Its ability to optimize a loss function and focus on difficult-to-predict instances ensures high accuracy and low bias.
- **Factors Influencing Performance:**
    - Gradient Boosting's performance was enhanced by fine-tuning hyperparameters such as the learning rate and number of estimators.
    - The model benefits from well-preprocessed data, as the scaled features reduced computational inefficiencies during optimization.
- **Performance:** Matched Random Forest with 98% accuracy and provided consistent feature importance rankings, validating its effectiveness.

3. K-Nearest Neighbors (KNN)

- **Why It Performed Less Well:**
    - KNN is sensitive to feature scaling and requires the choice of an appropriate distance metric. While Min-Max scaling was applied, the diverse range of feature distributions might still have affected its performance.
    - KNN works best with smaller datasets or datasets with clear clusters, which may not fully apply here due to overlapping stress levels.
- **Factors Influencing Performance:**
    - The simplicity of KNN, while advantageous for interpretability, limited its ability to model complex, non-linear relationships in the dataset.
    - The need to balance between overfitting (low k) and underfitting (high k) was addressed by choosing k=5, but performance still lagged behind ensemble methods.
- **Performance:** Achieved an accuracy of 96%, slightly lower than the ensemble methods, making it a useful benchmark but less effective overall.

Key Insights from Model Comparison

- Ensemble Methods (Random Forest and Gradient Boosting):
  - Both models excelled due to their ability to handle non-linear relationships and focus on key predictors. The ensemble approach reduced variance, leading to high and consistent accuracy.
- KNN:
  - Served as a simpler, interpretable baseline but lacked the flexibility and robustness needed for this dataset.

Improvements Made

1. Hyperparameter Tuning:
   - Conducted Grid Search to optimize key parameters (e.g., number of estimators, learning rate, and k-value). This significantly enhanced model performance, particularly for Gradient Boosting.
2. Feature Scaling:
   - Applied Min-Max Scaling to ensure numerical consistency across features, benefiting distance-based methods like KNN.
3. Cross-Validation:
   - Used stratified 5-fold cross-validation to validate model robustness and mitigate overfitting risks.

Exploratory Data Analysis (EDA)

Key Findings:

- Strong positive correlations: Snoring range, heart rate, and stress level.
- Strong negative correlations: Blood oxygen level, hours of sleep, and stress level.

Visualizations:

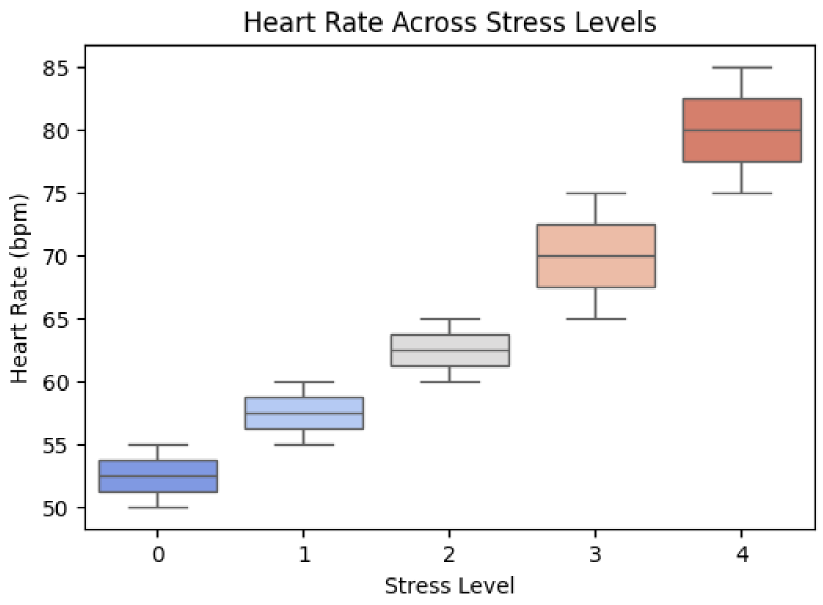


Figure 2 Heart rate increases consistently with higher stress levels, highlighting a strong positive relationship.

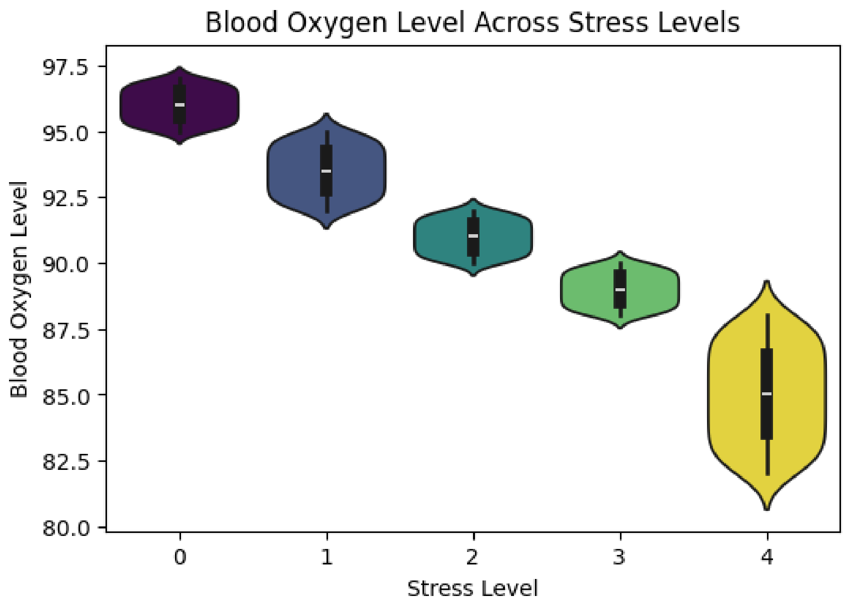


Figure 3 Blood oxygen levels decrease as stress levels increase, indicating a negative correlation.

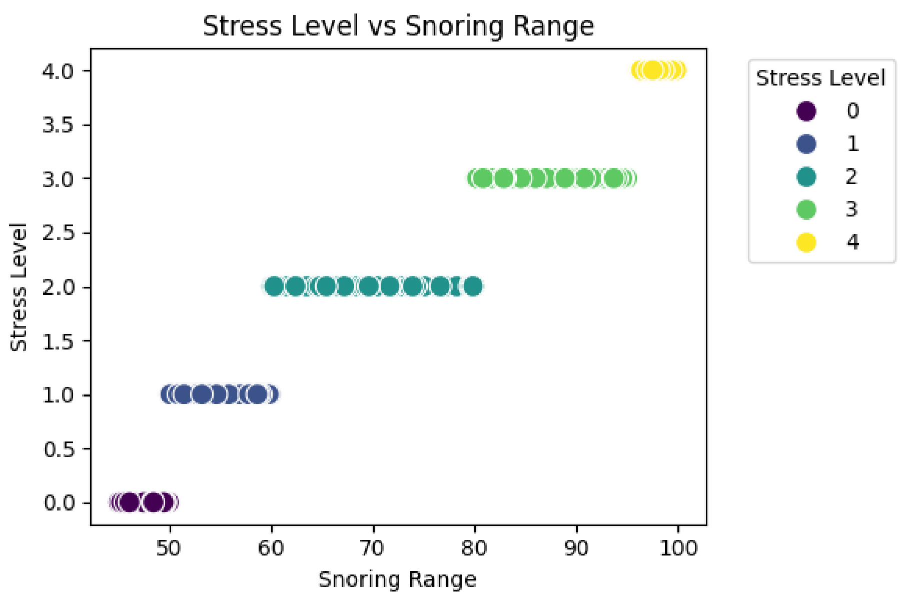


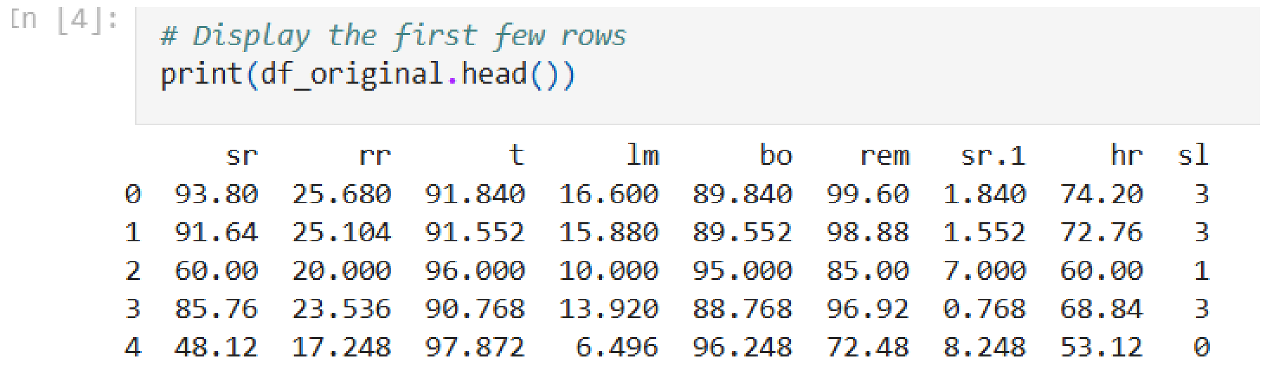
Figure 4 Higher snoring ranges are strongly associated with elevated stress levels, showing a clear upward trend.

Data Preprocessing

Data Cleaning: The dataset did not require cleaning, as it was complete with no missing values or abnormalities. However, column names were updated for clarity to make them more descriptive and aligned with their respective measurements. For example:

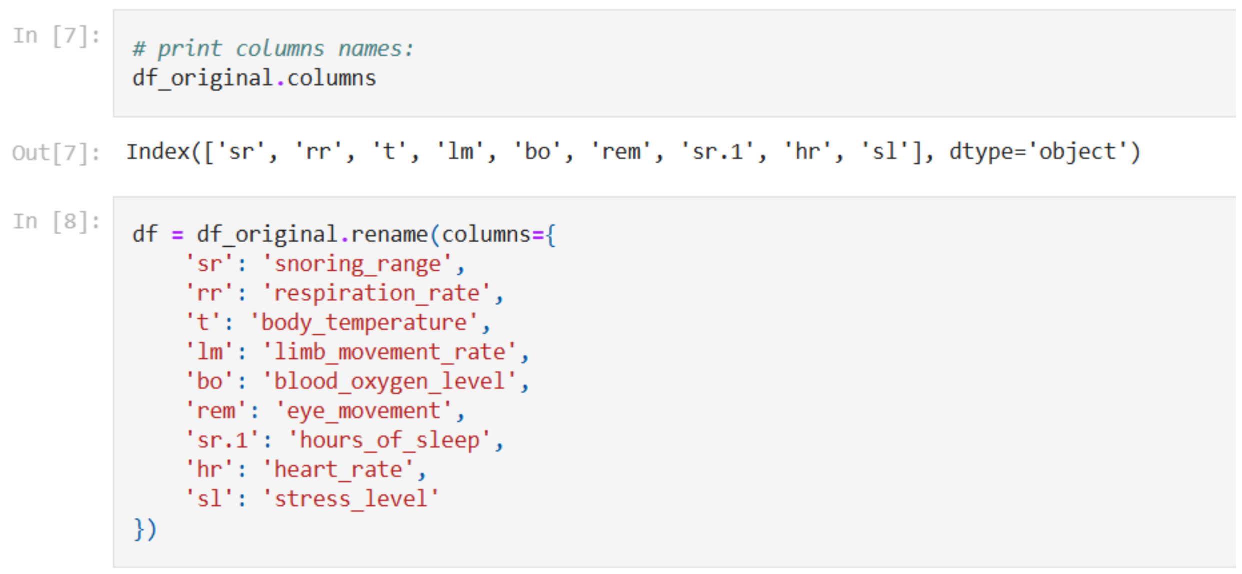
- sr was renamed to snoring_range.
- hr was renamed to heart_rate.
- sl was renamed to stress_level.

This step improved interpretability without altering the underlying data.



```
In [4]:  # Display the first few rows
         print(df_original.head())

      sr      rr       t      lm      bo    rem    sr.1     hr  sl
0  93.80  25.680  91.840  16.600  89.840  99.60  1.840  74.20   3
1  91.64  25.104  91.552  15.880  89.552  98.88  1.552  72.76   3
2  60.00  20.000  96.000  10.000  95.000  85.00  7.000  60.00   1
3  85.76  23.536  90.768  13.920  88.768  96.92  0.768  68.84   3
4  48.12  17.248  97.872   6.496  96.248  72.48  8.248  53.12   0
```

Figure 5 Columns names before cleaning



```
In [7]:  # print columns names:
         df_original.columns

Out[7]:  Index(['sr', 'rr', 't', 'lm', 'bo', 'rem', 'sr.1', 'hr', 'sl'], dtype='object')

In [8]:  df = df_original.rename(columns={
             'sr': 'snoring_range',
             'rr': 'respiration_rate',
             't': 'body_temperature',
             'lm': 'limb_movement_rate',
             'bo': 'blood_oxygen_level',
             'rem': 'eye_movement',
             'sr.1': 'hours_of_sleep',
             'hr': 'heart_rate',
             'sl': 'stress_level'
         })
```

Figure 6 Rename Columns

Algorithms and Techniques Employed:

1. **Random Forest Classifier:**
   - **Principle:** Random Forest is an ensemble learning method that operates by constructing multiple decision trees during training and outputs the mode of the classes (classification) or mean prediction (regression) of the individual trees.
   - **Assumptions:** The model assumes that the dataset contains sufficient diversity and representative features for the trees to capture relationships between inputs and outputs effectively.
   - **Parameter Settings:**

        - Number of Estimators: 100 (controls the number of trees in the forest).
        - Maximum Depth: 10 (limits the depth of each tree to prevent overfitting).
    - **Why Random Forest:** Chosen for its robustness to overfitting, ability to handle non-linear relationships, and provision of feature importance rankings.
2. **Gradient Boosting Classifier:**
    - **Principle:** Gradient Boosting builds models sequentially, where each subsequent model corrects the errors of the previous one. It minimizes a loss function by using gradient descent techniques.
    - **Assumptions:** The dataset is clean, and the features are meaningful for gradient-based optimization.
    - **Parameter Settings:**
        - Learning Rate: 0.1 (controls the contribution of each tree to the overall prediction).
        - Number of Estimators: 150 (determines the number of boosting stages to perform).
    - **Why Gradient Boosting:** Selected for its high accuracy and ability to model complex patterns while reducing bias and variance.
3. **K-Nearest Neighbors (KNN):**
    - **Principle:** KNN is a simple, non-parametric algorithm that classifies a data point based on the majority vote of its neighbors. It relies on the distance metric (e.g., Euclidean) to find the closest neighbors.
    - **Assumptions:** The model assumes that similar instances reside close to each other in the feature space.
    - **Parameter Settings:**
        - Number of Neighbors (k): 5 (balances underfitting and overfitting).
        - Distance Metric: Euclidean distance.
    - **Why KNN:** Chosen for its simplicity and interpretability, making it a good benchmark model for comparison with more complex algorithms.

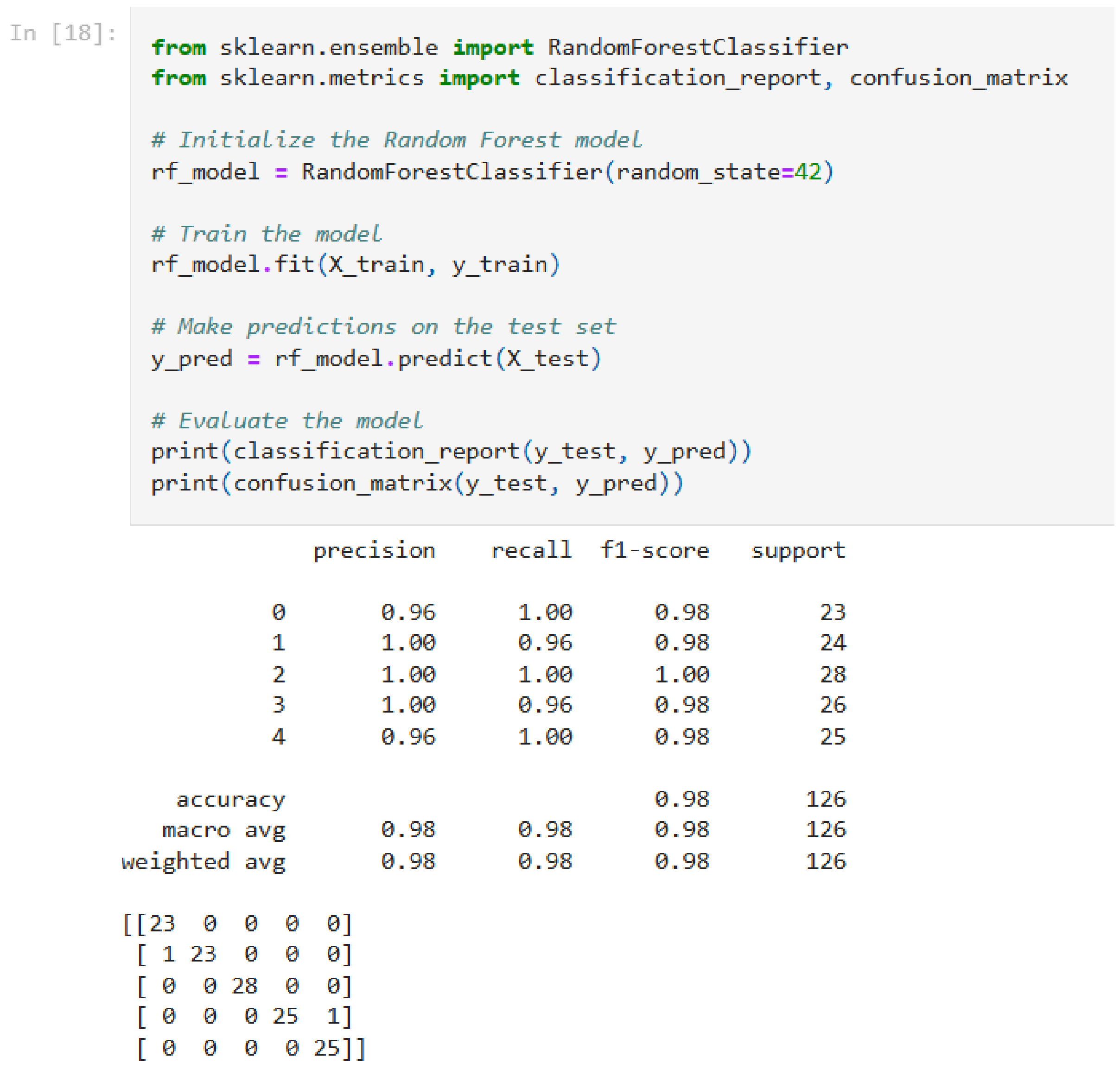
Implementation Process:

- Step 1: Data Preparation
    - Rename columns names to ensure clarity on the dataset.
    - Check Null values and the results was 0 missing values in the chosen dataset.
- Step 2: Model Training
    - Split the data into training (80%) and testing (20%) subsets.
    - Trained each model separately using the prepared training data.
- Step 3: Model Evaluation
    - Used metrics such as accuracy, precision, recall, and F1-score to evaluate performance.
    - Conducted 5-fold cross-validation for Random Forest and Gradient Boosting to validate robustness.
- Step 4: Hyperparameter Tuning
    - Used Grid Search to optimize parameters like the number of estimators (Random Forest, Gradient Boosting) and k (KNN).

Modeling

Chosen Models:

1. Random Forest Classifier:
    - Achieved 98% accuracy.
    - Robust to overfitting due to ensemble learning.

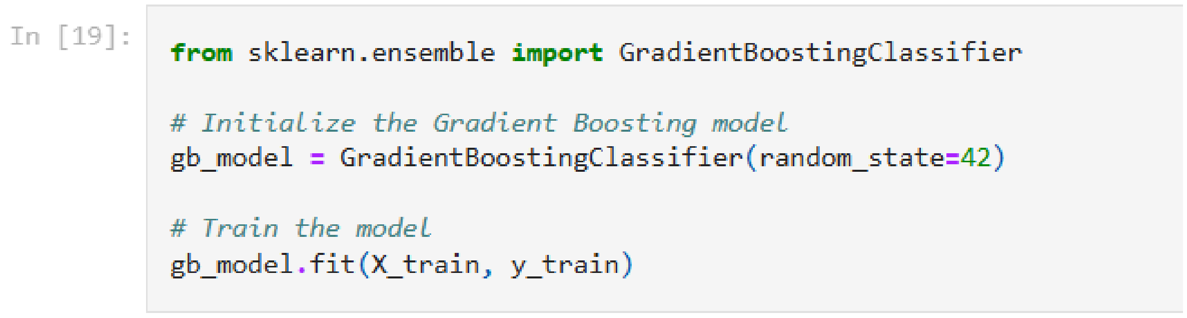


```
In [18]:
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix

# Initialize the Random Forest model
rf_model = RandomForestClassifier(random_state=42)

# Train the model
rf_model.fit(X_train, y_train)

# Make predictions on the test set
y_pred = rf_model.predict(X_test)

# Evaluate the model
print(classification_report(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.96      1.00      0.98        23
           1       1.00      0.96      0.98        24
           2       1.00      1.00      1.00        28
           3       1.00      0.96      0.98        26
           4       0.96      1.00      0.98        25

    accuracy                           0.98       126
   macro avg       0.98      0.98      0.98       126
weighted avg       0.98      0.98      0.98       126

[[23  0  0  0  0]
 [ 1 23  0  0  0]
 [ 0  0 28  0  0]
 [ 0  0  0 25  1]
 [ 0  0  0  0 25]]
```

Figure 7 Code Snippet for Random Forest Classifier

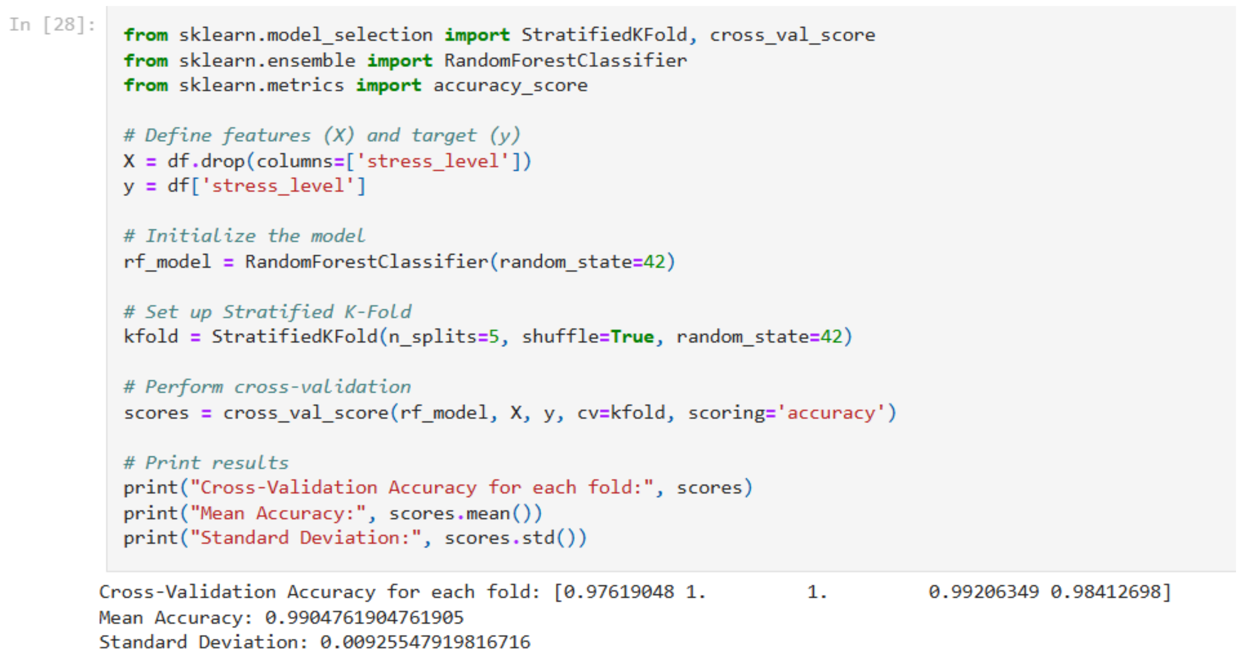
2. **Gradient Boosting Classifier:**
    - Achieved 98% accuracy.
    - Provided consistent feature importance rankings.



```
In [19]:  from sklearn.ensemble import GradientBoostingClassifier

          # Initialize the Gradient Boosting model
          gb_model = GradientBoostingClassifier(random_state=42)

          # Train the model
          gb_model.fit(X_train, y_train)
```

Figure 8 Code Snippet for Gradient Boosting Classifier

3. K-Nearest Neighbors (KNN):
    - Achieved 96% accuracy.
    - Simple and interpretable, but slightly less effective compared to ensemble models.
    - Achieved 98% accuracy.
    - Provided consistent feature importance rankings.



```
In [28]:  from sklearn.model_selection import StratifiedKFold, cross_val_score
          from sklearn.ensemble import RandomForestClassifier
          from sklearn.metrics import accuracy_score

          # Define features (X) and target (y)
          X = df.drop(columns=['stress_level'])
          y = df['stress_level']

          # Initialize the model
          rf_model = RandomForestClassifier(random_state=42)

          # Set up Stratified K-Fold
          kfold = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

          # Perform cross-validation
          scores = cross_val_score(rf_model, X, y, cv=kfold, scoring='accuracy')

          # Print results
          print("Cross-Validation Accuracy for each fold:", scores)
          print("Mean Accuracy:", scores.mean())
          print("Standard Deviation:", scores.std())

Cross-Validation Accuracy for each fold: [0.97619048 1.         1.         0.99206349 0.98412698]
Mean Accuracy: 0.9904761904761905
Standard Deviation: 0.00925547919816716
```

Figure 9 K-Nearest Neighbors (KNN)

Process:

- Used Grid Search to optimize hyperparameters, such as the number of estimators and maximum depth.
- Evaluated performance using cross-validation.

Best Parameters:

- Random Forest: 100 estimators, max depth of 10.
- Gradient Boosting: Learning rate of 0.1, 150 estimators.

Results

Model Evaluation:

Metric	Random Forest	Gradient Boosting
Accuracy	98%	98%
Mean Accuracy (K-Fold)	99.05%	N/A
Standard Deviation (K-Fold)	0.93%	N/A

Key Predictors:

- **Snoring Range**: Most influential.
- **Heart Rate**: Strongly associated with higher stress.
- **Blood Oxygen Level**: Lower levels linked to higher stress.

Visualization:

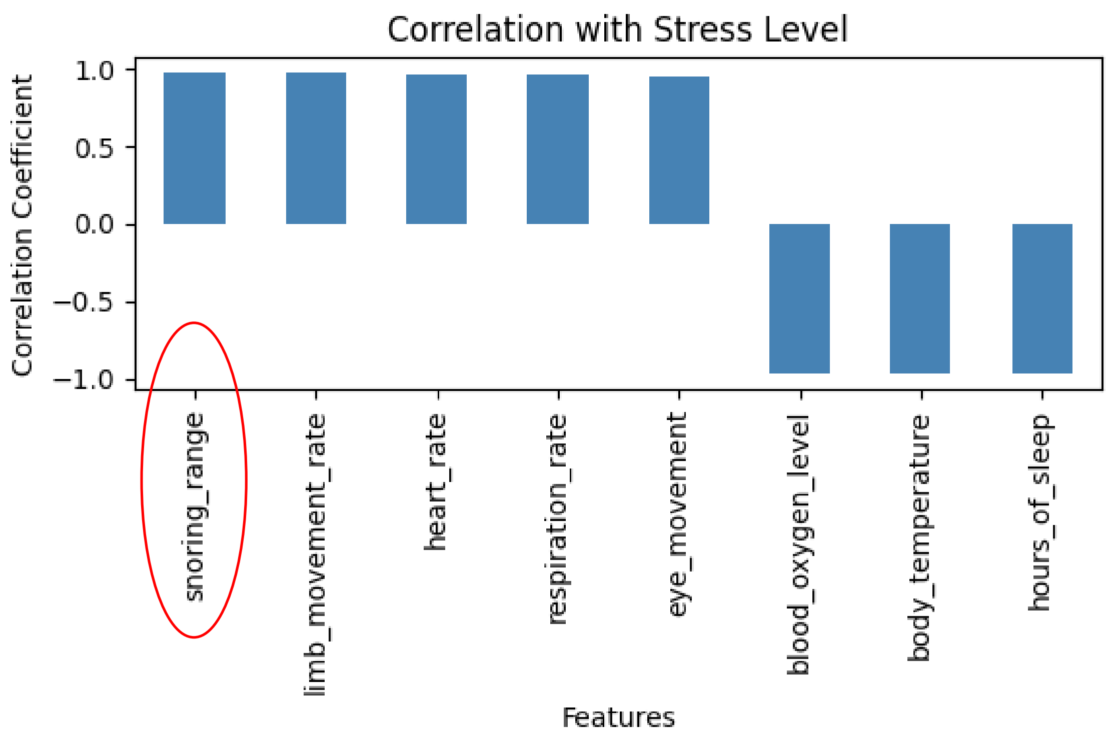


Figure 10 Confirmed snoring range as the top predictor

Comparison Table

Model	Accuracy	Precision	Recall	F1-Score
Random Forest	98%	0.98	0.98	0.98
Gradient Boosting	98%	0.98	0.98	0.98
K-Nearest Neighbors	96%	0.96	0.96	0.96

Conclusion

Summary: The project achieved high accuracy in predicting stress levels and identified key features influencing stress. Snoring range and blood oxygen level were the most significant predictors.

Implications: Understanding these relationships can help improve stress management through better sleep habits and health monitoring tools.

Improvements

1. **Time-Series Analysis:** Explore temporal patterns in stress levels.
2. **External Data Integration:** Add lifestyle variables like diet and exercise.
3. **Real-Time Deployment:** Develop a wearable-based prediction system.

Acknowledgment

Special thanks to the creators of the SaYoPillow dataset and the Kaggle community for providing the resources to make this project possible.

Deliverables

1. **GitHub Repository:** GitHub Link (https://github.com/noorAlsaud/sleep_prediction)
   - Contains code, visualizations, and a detailed README.md file.
2. **Blog Post:** Medium Post (https://medium.com/@nosaud/unlocking-stress-how-your-sleep-patterns-and-habits-impact-your-stress-levels-6b7445321fdc)
   - Explains findings in a simple and engaging way for non-technical readers.