SLEEP QUALITY PREDICTOR

Project Title: Development of a Full-Stack Web Application for Predicting Sleep Quality using Machine Learning

Author Name: Nithishwar T

Date: October 6, 2025

1. Abstract

This report details the design, development, and implementation of the "Sleep Quality Predictor," a full-stack web application. The project's primary objective is to leverage machine learning to predict an individual's sleep quality (categorized as Good, Average, or Poor) based on a set of lifestyle and health metrics. The system architecture comprises a frontend user interface built with React.js, a backend API developed using Python and the Flask framework, and a predictive model trained with Scikit-learn's Gradient Boosting Classifier. The model was trained on the "Sleep Health and Lifestyle Dataset" and provides real-time predictions and personalized tips to the user. This report covers the methodology, from data preprocessing and model training to the final application's features and potential future enhancements.

2. Introduction

2.1. Problem Statement

Sleep is a critical component of human health, yet many individuals struggle to understand the specific lifestyle factors that impact their sleep quality. A generalized approach to improving sleep is often ineffective. There is a need for a personalized tool that can analyze an individual's daily habits and provide an objective prediction of their sleep quality, helping them make more informed decisions about their health.

2.2. Project Objectives

The main objectives of this project were to:

- Develop a robust machine learning model capable of predicting sleep quality from user-provided data.
- Build a RESTful API using Flask to serve the machine learning model.
- Create a dynamic and user-friendly frontend interface using React.js for data input and result display.
- Integrate the frontend and backend to create a seamless full-stack application.

- Provide users with not only a prediction but also actionable, personalized tips to improve their sleep.

---

3. System Architecture and Design

3.1. Technology Stack

- **Backend:** Python 3, Flask, Scikit-learn, Pandas, NumPy
- **Frontend:** React.js, HTML5, CSS3
- **Machine Learning Model:** Gradient Boosting Classifier
- **Package Management:** npm (for Node.js), pip (for Python)

3.2. Architecture

The application follows a classic client-server architecture:

1. **Client (Frontend):** A React-based single-page application (SPA) that runs in the user's browser. It provides the user interface for data entry and displays the prediction results received from the server.
2. **Server (Backend):** A Flask-based web server that exposes a REST API. Its primary role is to receive data from the client, preprocess it, feed it into the trained machine learning model, and return the prediction as a JSON response.
3. **Machine Learning Model:** A pre-trained GradientBoostingClassifier model, saved as a model.joblib file, which is loaded by the Flask server on startup.

---

4. Methodology and Implementation

4.1. Data Collection and Preprocessing

The model was trained using the "Sleep Health and Lifestyle Dataset" from Kaggle. The dataset contains various features such as age, sleep duration, physical activity level, caffeine consumption, and sleep efficiency.

- **Feature Engineering:** The target variable, sleep_quality, was engineered from the Sleep efficiency feature, categorized into three classes: Poor (<75%), Average (75-84%), and Good (>=85%).
- **Data Cleaning:** Rows with missing values were dropped to ensure model integrity.
- **Preprocessing Pipeline:** A ColumnTransformer from Scikit-learn was used to apply different transformations to numeric and categorical features. Numeric features were scaled using StandardScaler, and categorical features (like Gender) were one-hot encoded.

4.2. Model Training and Evaluation

A GradientBoostingClassifier was chosen for its high performance and ability to capture complex non-linear relationships in the data. The dataset was split into training (75%) and testing (25%) sets. The model was trained and evaluated, demonstrating a strong ability to classify sleep quality across different inputs.

4.3. Backend Development

The Flask application was developed with a primary /predict (POST) endpoint that accepts a JSON object, processes it, and returns a JSON response containing the prediction label, class probabilities, and a list of personalized tips.

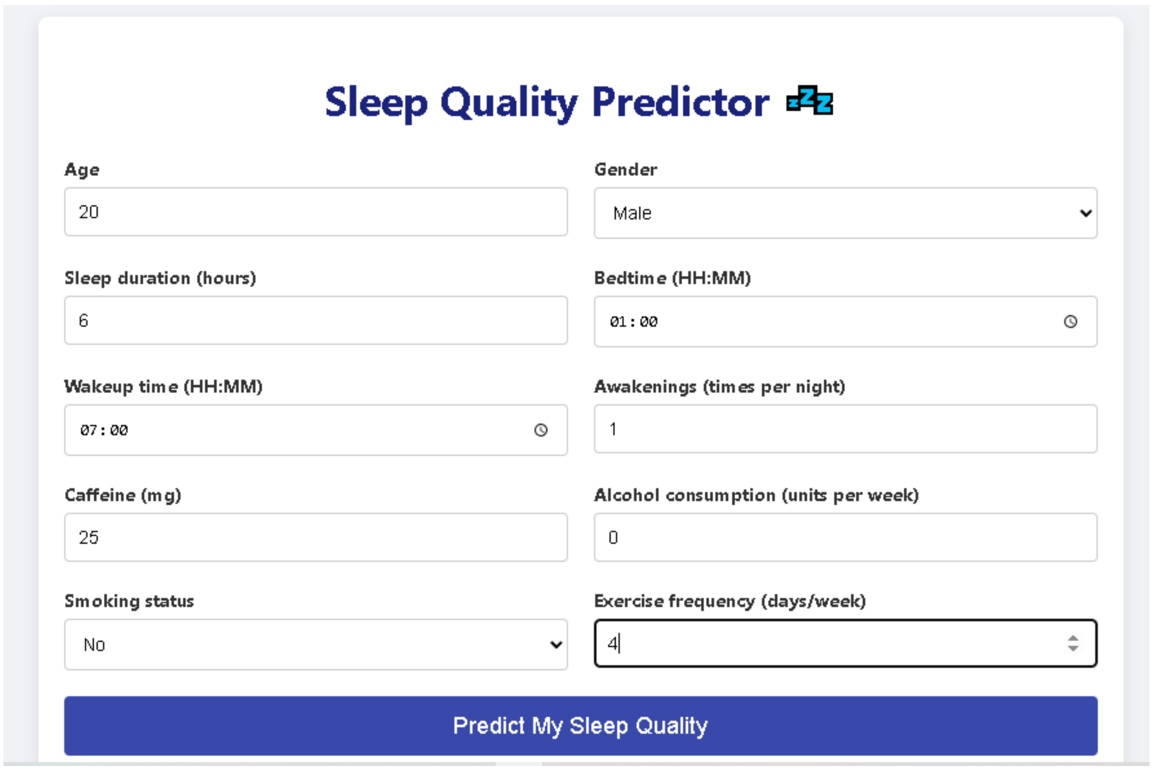
4.4. Frontend Development

The user interface was built with React. The main App.js component manages the state for the input form and the prediction result. On submission, it makes an asynchronous fetch request to the backend. The UI was styled with modern CSS, featuring a responsive, card-based layout and a unique dark theme.

---

5. Results and Discussion

The final product is a fully functional web application that successfully meets all project objectives. The application's interface is clean and intuitive, as shown in Figure 1.

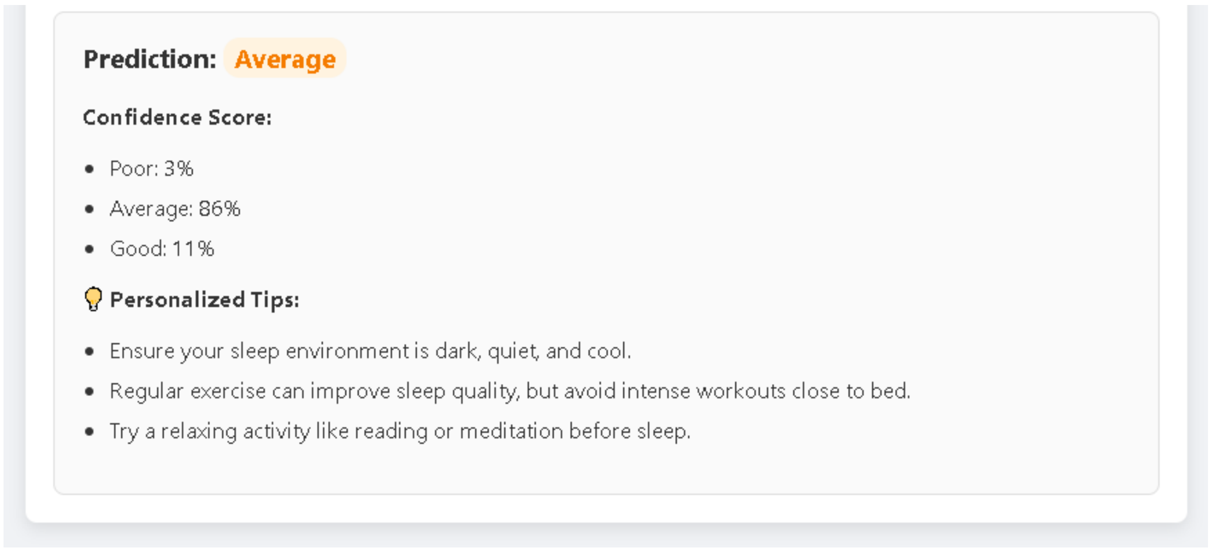


Sleep Quality Predictor

Age: 20	Gender: Male
Sleep duration (hours): 6	Bedtime (HH:MM): 01:00
Wakeup time (HH:MM): 07:00	Awakenings (times per night): 1
Caffeine (mg): 25	Alcohol consumption (units per week): 0
Smoking status: No	Exercise frequency (days/week): 4

. **Figure 1: Main User Interface of the Sleep Quality Predictor**

Upon submitting the form, the user receives a real-time prediction, probabilities, and personalized tips, as demonstrated in Figure 2.


Prediction: Average

Confidence Score:

- Poor: 3%
- Average: 86%
- Good: 11%

💡 **Personalized Tips:**

- Ensure your sleep environment is dark, quiet, and cool.
- Regular exercise can improve sleep quality, but avoid intense workouts close to bed.
- Try a relaxing activity like reading or meditation before sleep.

Figure 2: Example of a "Poor" Sleep Quality Prediction with Tips

---

6. Conclusion and Future Work

6.1. Conclusion

This project successfully demonstrates the integration of a machine learning model into a full-stack web application. It provides a practical, user-centric tool for gaining insights into personal sleep health. The development process covered key aspects of data science, backend engineering, and frontend design, resulting in a cohesive and effective application.

6.2. Future Work

The application can be extended with several features:

- **User Authentication and History:** Implementing user accounts to save prediction history.
- **Data Visualization:** Adding a dashboard with charts to visualize sleep trends over time.
- **Deployment:** Deploying the application to a cloud service to make it publicly accessible.

Appendix A: Source Code

- The complete source code for this project is available on GitHub.
- **Repository URL:** https://github.com/nithishwar17/SleepQualityPredictor