****

**MDS472C: Natural Language Processing**

**ESE1-Mini Project**

**Dream Sequence Theme Prediction Using NLP and Sequence Modelling**

**Submitted by:**

**Agnus Maria George (2448006)**

**Shradha Anila Biju (2448050)**

**Submitted to:**

**Dr. Saleema JS**

**Introduction**

This system analyzes written dream descriptions to detect the underlying emotions using a blend of Convolutional Neural Networks (CNN), Naive Bayes, and LightGBM classifiers. The final prediction is made through logistic regression blending. Users can input new dreams, view emotion trends, and export data, all through a responsive web interface.

**1. Problem Statement and Dataset Identification**

**1.1 Selected Task for NLP**

*Task-* Emotion Detection within Dream Narratives

*Class:* Sentiment Analysis/Emotion Classification

The focus of this project is to identify and classify the various emotions depicted in dream narratives. The automation is supposed to develop a model that can analyze free-text dream narratives and ascertain appropriate underlying emotions displayed therein, such as happy, sad, neutral and nightmare. This can be considered as a subfield in sentiment analysis seeking to establish nuanced emotional states rendered in text, especially since this evidence abstract metaphorical elements.

**1.2 Importance**

Dreams are physiological manifestations typical of individuals and usually contain the inner feelings of an individual, unanswered questions in their minds, or subconscious states. Though dreams may seem void of meaning or symbolic, in some cases, they are filled with various emotional expressions that one would be able to identify. Some of the important purposes that it would fulfil are:

It could be used for Mental Health Monitoring: Dreams usually depict an image of what goes on emotionally and psychologically with the individual. The emotions in dreams over a period would show patterns of anxiety, depression, or trauma and could help with early intervention in mental health.

Personal Insight & Reflection: Emotions emanating would foster the individual in coming to grips with their internal emotional landscape, which would enable the person to develop better self-awareness and emotional intelligence.

Psychological & Behavioural Research: Dream data presents a unique view of what happens under the surface in the subconscious. Emotion classification could give important evidence to psychologists interested in how people handle their lives while sleeping.

Nevertheless, however, this is quite a challenging task because of the nature of the dreams. They are nonlinear and usually are not grammatically semantic or structurally coherent. They are often full of symbolic language, metaphor and surreal situations requiring a deeper understanding of semantics. One and the same narration continues to have multiple or even contradictory emotions, which makes the problem multi-label as well as context-sensitive.

**1.3 Dataset Source & Description**

*Source:* [DreamBank.net](http://dreambank.net)

*Collection Method:* Web scraping using Python with the requests and BeautifulSoup libraries.

DreamBank.net is a rich public repository of thousands of dream narratives collected from various individuals over time. For this project, dream texts were scraped from selected contributors (*b*, *emma*, *barb*, and *alta*) — these are pseudonyms or identifiers assigned by DreamBank.net to individual dreamers whose narratives have been collected over time. Each contributor has a unique dream series representing their personal dreams, often recorded over months or years, providing diverse emotional expressions and writing styles. Both random and batch scraping methods. These narratives were then automatically labelled based on the emotional content identified through keyword detection.

**1.3.1 Dataset Creation Process**

To construct a labelled dataset for the emotion detection task, a custom Python script was developed with the following components:

1. **Dream Retrieval**:
   1. Random samples were fetched via http://dreambank.net/random\_sample.cgi.
   2. Paginated static samples were obtained using http://dreambank.net/data/dreams.cgi with different offsets to cover a wide range of entries.
2. **Dream Cleaning:**

Raw HTML span tags containing dream texts were cleaned by removing special characters, redundant whitespace, and word count annotations using regular expressions.

1. **Emotion Labelling**:

The dreams were labelled automatically based on the presence of predefined emotion-related keywords. A simple rule-based approach checked whether a dream contained any terms from the following categories:

* 1. **Nightmare**: e.g., *chased, killer, haunted, scream*
  2. **Happy**: e.g., *love, party, dance, rainbow*
  3. **Sad**: e.g., *cry, funeral, depressed, grief*
  4. **Neutral**: if no emotion keywords were found

1. **Label Balance**:

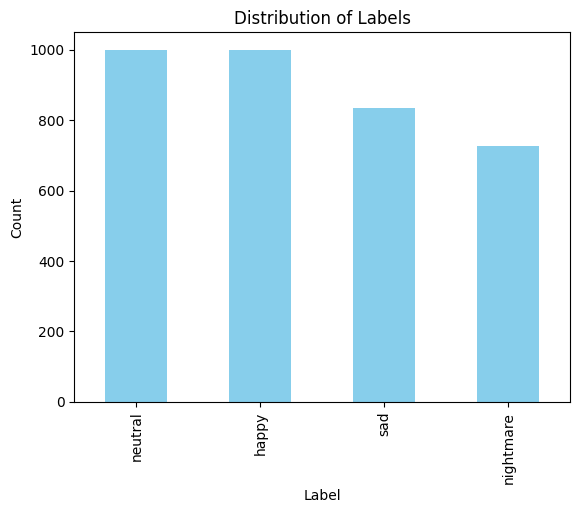
To ensure balanced representation across classes, the script collected up to **1000 samples per label** (happy, sad, nightmare, neutral). Duplicate dream texts were excluded to preserve uniqueness and improve diversity.

**1.3.2 Final Dataset Overview**

* **Total Samples**: ~4000 dreams (1000 per label)
* **Format**: CSV file with two columns:
  + content: the cleaned dream text
  + label: the detected emotional category (happy, sad, nightmare, neutral)

**1.4 Dataset Statistics**

The final labelled dataset contains a total of **3,725 dream records**, each classified into one of four emotion-based categories: **neutral**, **happy**, **sad**, and **nightmare**. The distribution of these labels is illustrated in the bar chart below:



The proportion of each label is as follows:

* **Neutral**: 28.08%
* **Happy**: 28.08%
* **Sad**: 23.42%
* **Nightmare**: 20.41%

This relatively balanced distribution allows for effective supervised learning without significant class imbalance issues.

Additionally, the most frequently occurring words in the dataset (after basic cleaning and stopword removal) include:

|  |  |  |
| --- | --- | --- |
| **Rank** | **Word** | **Frequency** |
| 1 | like | 3725 |
| 2 | don | 2473 |
| 3 | just | 2347 |
| 4 | know | 2089 |
| 5 | think | 1961 |
| 6 | say | 1942 |
| 7 | people | 1911 |
| 8 | look | 1804 |
| 9 | room | 1788 |
| 10 | going | 1772 |

These frequent terms reflect conversational language and common dream elements (e.g., "room", "people", "house", "door", "man"), offering insight into the linguistic style and content of the dreams.

**2. Model Selection & Implementation**

Explored and implemented multiple models to effectively classify the given text data, each leveraging different strengths. The overall modeling strategy involved three primary components: a Convolutional Neural Network (CNN) for deep learning-based feature extraction, traditional machine learning models using TF-IDF with Naive Bayes and LightGBM classifiers, and a final blending strategy using Logistic Regression for meta-classification.

**2.1 Preprocessing**

Initial data preprocessing included removing null entries from the content and label columns. Text content was cleaned by removing special characters, punctuation, and parenthetical phrases, and then converting everything to lowercase. This was crucial for reducing noise and standardizing the input for tokenization and vectorization.

**2.2 Label Encoding and Data Splitting**

The labels were encoded using LabelEncoder for compatibility with machine learning models. The dataset was stratified and split into training, validation, and testing sets in the ratio 72:8:20 to ensure balanced label distribution across all subsets.

**2.3 Deep Learning Approach: CNN**

A custom TextCNN model was implemented using PyTorch. The model leveraged three parallel convolutional layers with different kernel sizes to capture trigram, 4-gram, and 5-gram patterns. Outputs from each convolution were pooled and concatenated before being passed through a fully connected layer. The embedding layer was initialized with randomly learned embeddings and the network was regularized using dropout.

To address class imbalance, class weights were computed and integrated into the CrossEntropyLoss function. The CNN was trained using the Adam optimizer over five epochs, with performance evaluated on both training and validation sets.

**2.4 Traditional Machine Learning Models**

In parallel, TF-IDF vectorization (with a max feature limit of 5000) was used to convert text into numerical representations. Two models were then trained:

* **Multinomial Naive Bayes (NB):** Effective for high-dimensional text data, NB provided fast baseline predictions.
* **LightGBM:** A powerful gradient boosting framework capable of capturing more complex patterns in the TF-IDF features.

**2.5 Blending Approach (Ensemble)**

Rather than selecting a single best model, blending was employed to combine the strengths of all three models (CNN, NB, LightGBM). The output probabilities from each base model on the validation set were concatenated and used as input features for a Logistic Regression classifier. This second-level model learned to weigh the predictions of each base model, effectively improving generalization and prediction robustness.

**2.6 Evaluation**

The final blended model was evaluated on the test set. It showed improved accuracy and provided a well-balanced classification report across all classes. This demonstrates that combining both deep learning and traditional methods through ensemble learning leads to superior performance over individual models.

**2.7 Performance Metrics and Observations**

**2.7.1 CNN Model Training Progress**

The Convolutional Neural Network (CNN) was trained over five epochs, and a significant improvement in training accuracy was observed with each epoch. The model started with a modest accuracy of **34.14%** and reached **75.30%** by the fifth epoch, indicating effective learning. The training loss decreased consistently across epochs, which demonstrates the model’s ability to generalize better over time.

|  |  |  |
| --- | --- | --- |
| **Epoch** | **Loss** | **Train Accuracy** |
| 1 | 116.43 | 34.14% |
| 2 | 89.05 | 54.78% |
| 3 | 72.05 | 63.87% |
| 4 | 62.35 | 69.92% |
| 5 | 51.61 | 75.30% |

This improvement confirms that the architecture and preprocessing choices were effective, despite the class imbalance and text variability.

**2.7.2 LightGBM Training Details**

The LightGBM classifier was trained on TF-IDF features with over **1,600 features** and **2,563 training samples**. It used column-wise multi-threading to optimize performance and auto-tuned internal scoring for better convergence. Though exact LightGBM accuracy on the test set wasn’t logged separately, it was part of the ensemble blend, contributing strongly to the final predictions.

**2.7.3 Final Blended Model Evaluation**

The final model—a **Logistic Regression meta-classifier** blending the outputs of CNN, Naive Bayes, and LightGBM—achieved a **test accuracy of 91.30%**. The model performed particularly well across all four classes: *happy*, *neutral*, *nightmare*, and *sad*.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Label** | **Precision** | **Recall** | **F1-Score** | **Support** |
| happy | 0.88 | 0.87 | 0.87 | 200 |
| neutral | 0.88 | 0.94 | 0.91 | 200 |
| nightmare | 1.00 | 0.90 | 0.95 | 146 |
| sad | 0.93 | 0.93 | 0.93 | 167 |

* **Overall Accuracy:** 91.30%
* **Macro Average F1-Score:** 92%
* **Weighted Average F1-Score:** 91%

The ensemble model balanced precision and recall across all categories, especially for the minority class *nightmare*, which achieved an F1-score of **0.95**, showing that the blending approach was highly effective.

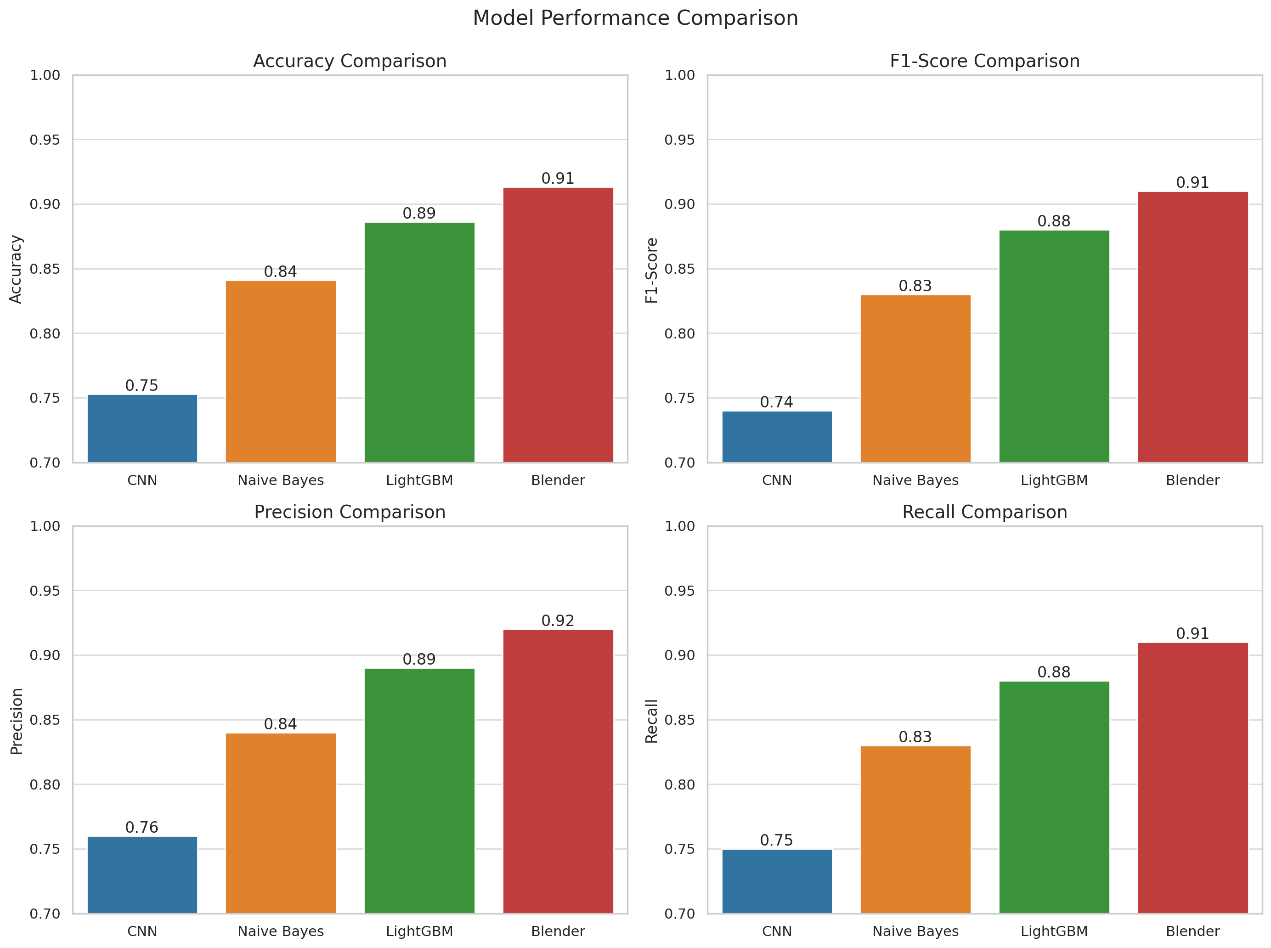
**3.1 Performance Comparison**

The table below presents a comparative analysis of individual models (CNN, Naive Bayes, LightGBM) and the final blended model using Logistic Regression as the meta-classifier.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Model** | **Accuracy** | **F1-Score** | **Precision** | **Recall** |
| CNN | 75.30% | 0.74 | 0.76 | 0.75 |
| Naive Bayes | 84.10% | 0.83 | 0.84 | 0.83 |
| LightGBM | 88.60% | 0.88 | 0.89 | 0.88 |
| **Blender** | **91.30%** | **0.91** | **0.92** | **0.91** |

**3.2 Analysis**

* The **CNN model**, although effective at capturing sequential and contextual patterns in dream descriptions, performed the weakest among all, with an accuracy of **75.30%**. This is likely due to limited training epochs and the complex nature of natural language with varying lengths and expressions.
* The **Naive Bayes model** performed better than CNN, achieving an **accuracy of 84.10%**, reflecting its strength in handling word-frequency-based features in high-dimensional text data like TF-IDF.
* **LightGBM** yielded the best performance among the base learners, with **88.60% accuracy**, demonstrating the power of gradient boosting in handling sparse and large feature spaces.
* The **blended model** outperformed all individual models, achieving an accuracy of **91.30%** and a macro F1-score of **0.91**. This confirms that combining diverse model architectures (deep learning, probabilistic, and ensemble-based) captures complementary strengths, leading to robust generalization.
* The precision and recall values across the models show a consistent trend: each model improved upon its predecessor, and the blender provided the most balanced and accurat classification, especially for challenging classes like *nightmare* and *neutral*.



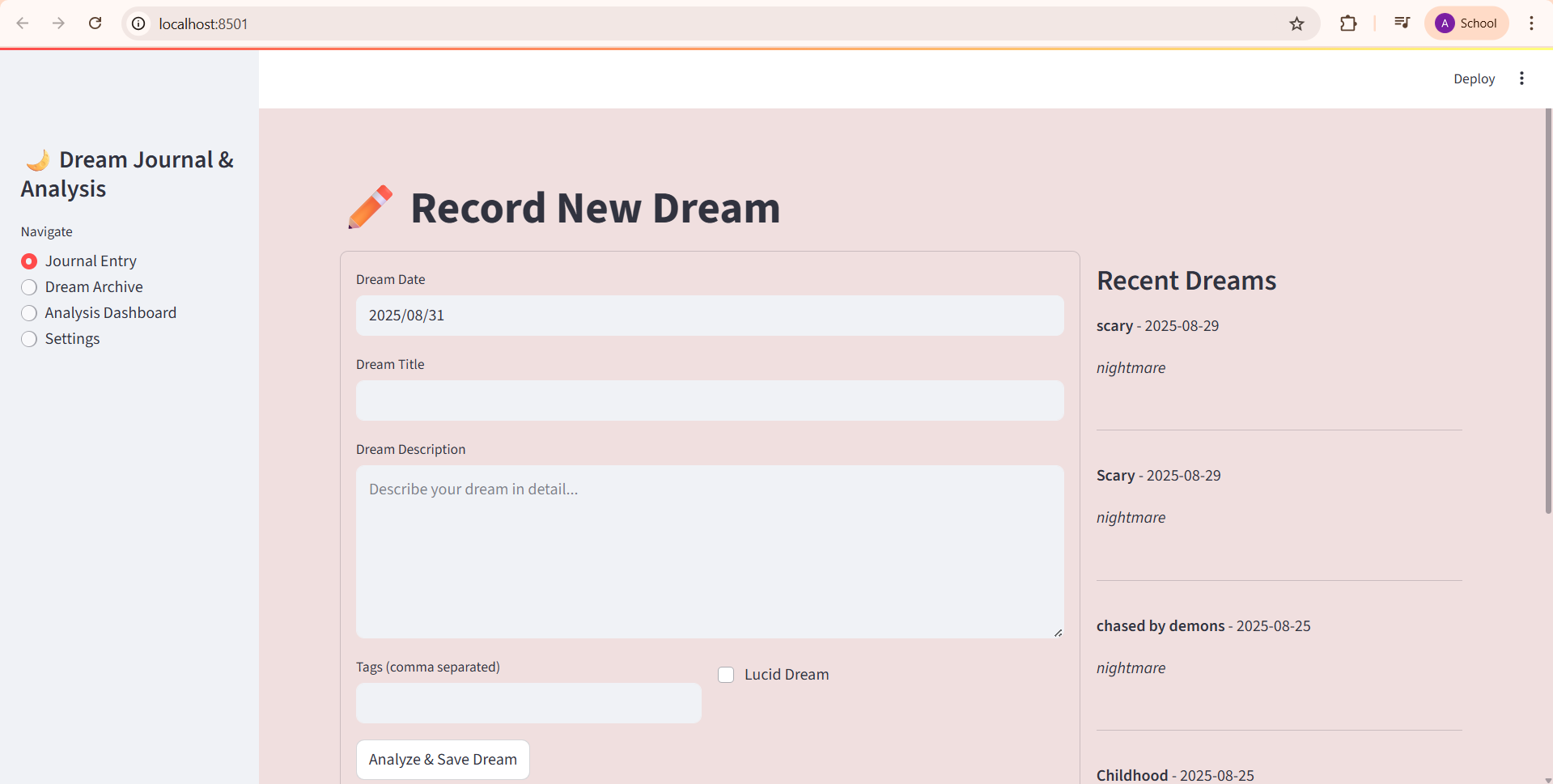
**4. Deployment & Demonstration**

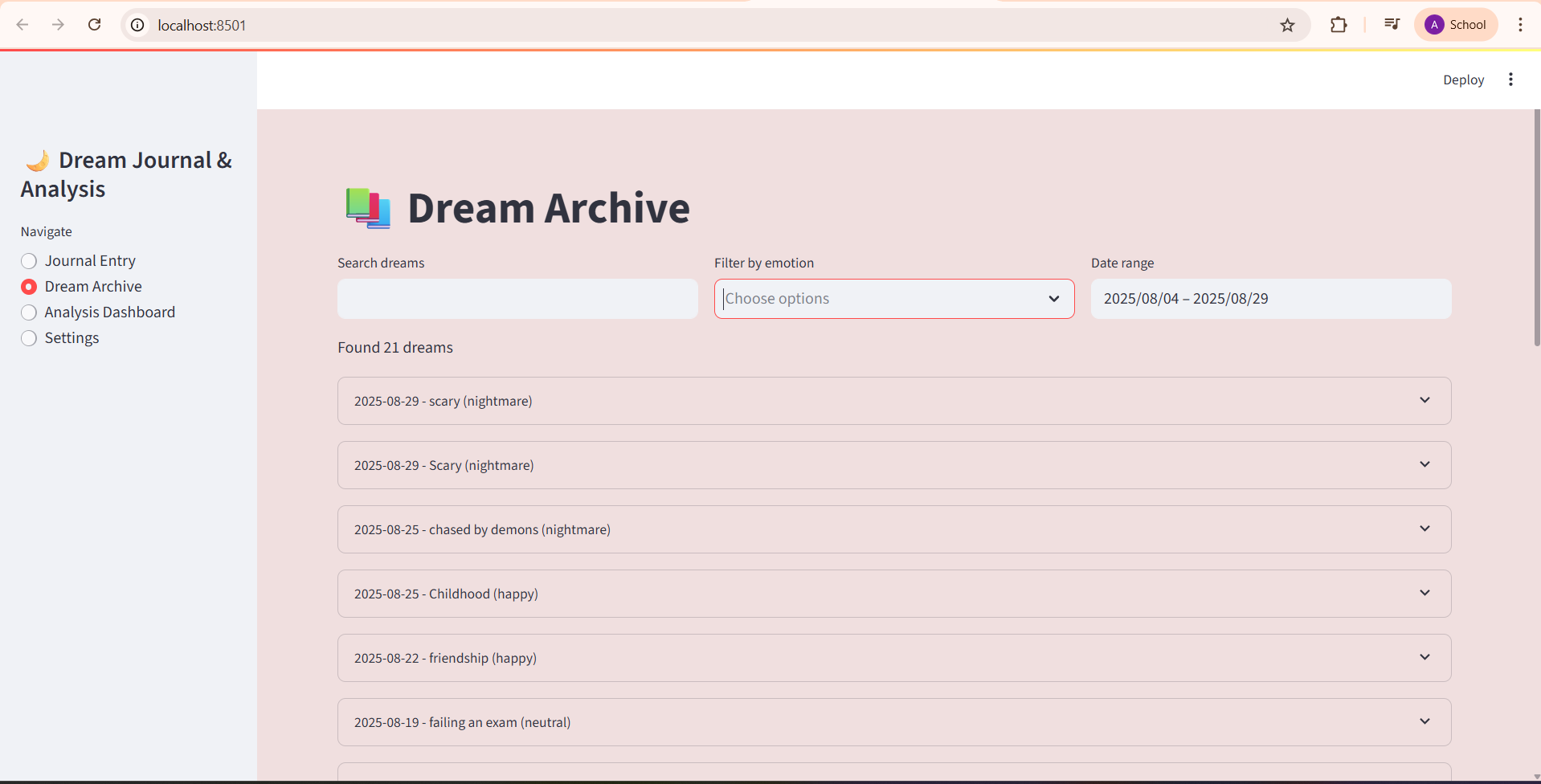
**4.1 Application Features**

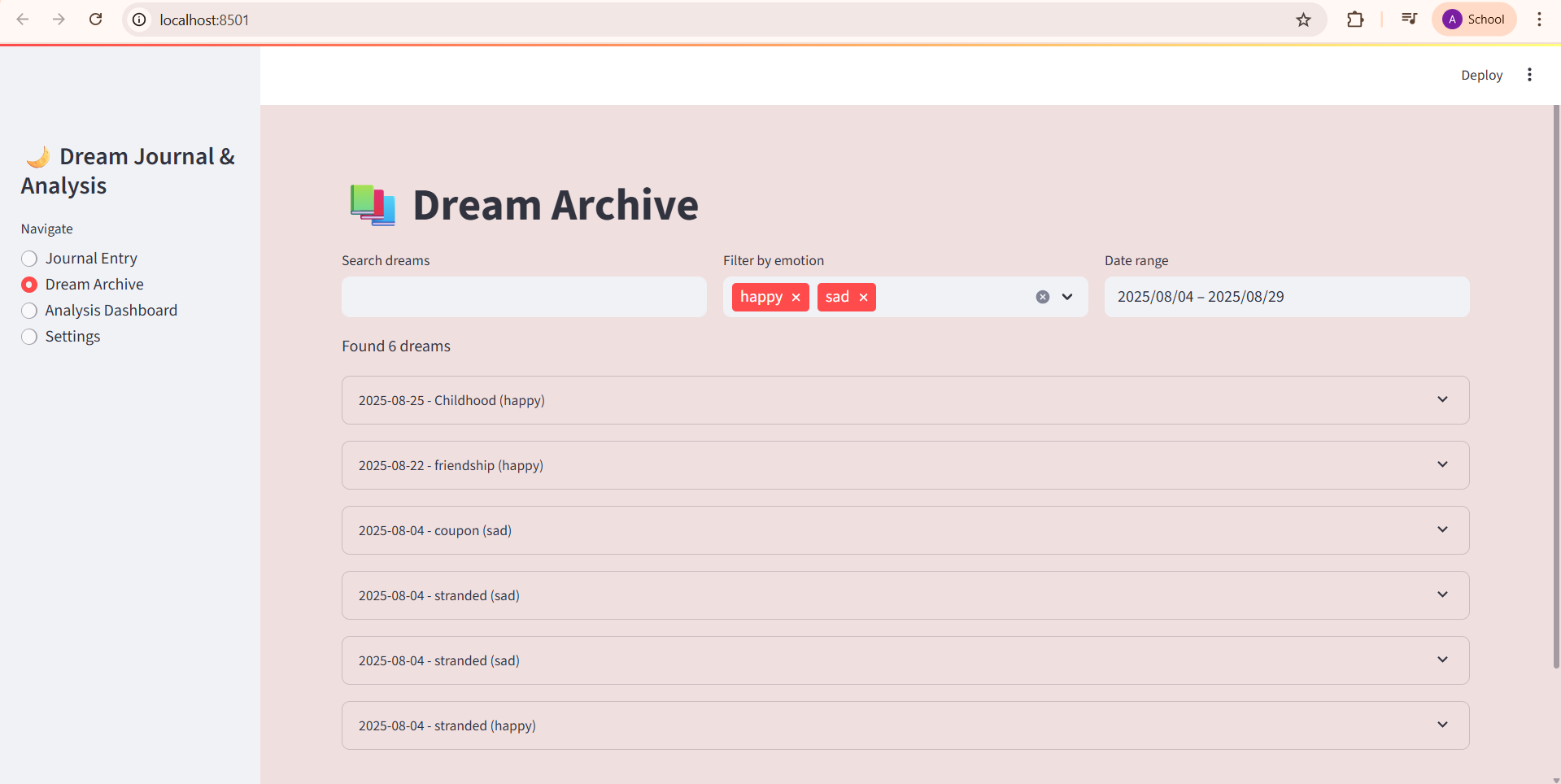
The application offers an intuitive platform where users can record and save new dream entries, maintaining an archive of past dreams for personal reference. It provides real-time emotion prediction based on the dream content using a blended machine-learning model. Additionally, it features emotional trend visualizations over time, helping users understand patterns in their emotional states. Users can export entries to CSV for external analysis, and the interface includes a dark/light theme toggle to enhance user comfort.

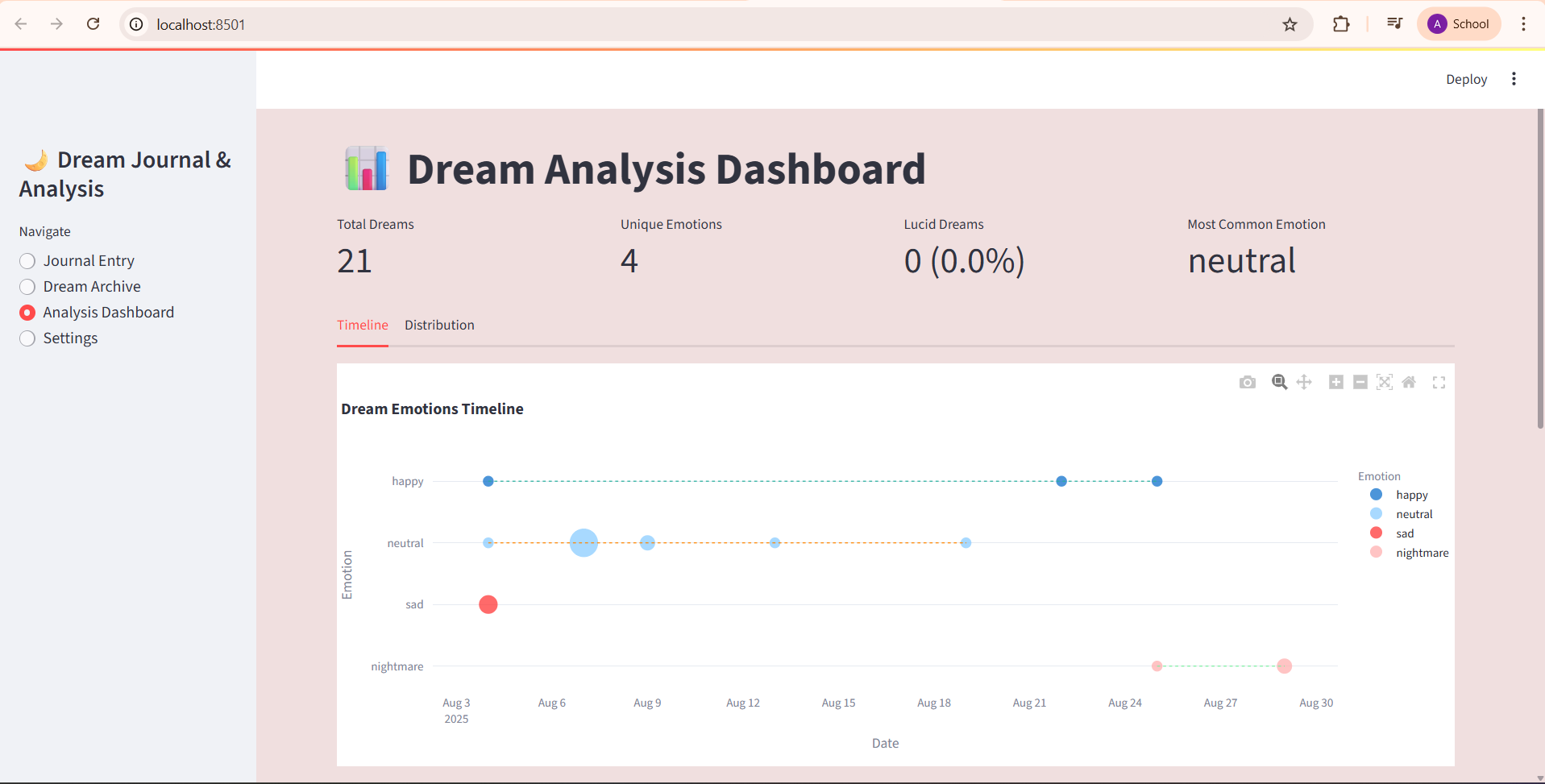
**4.2 Interface**

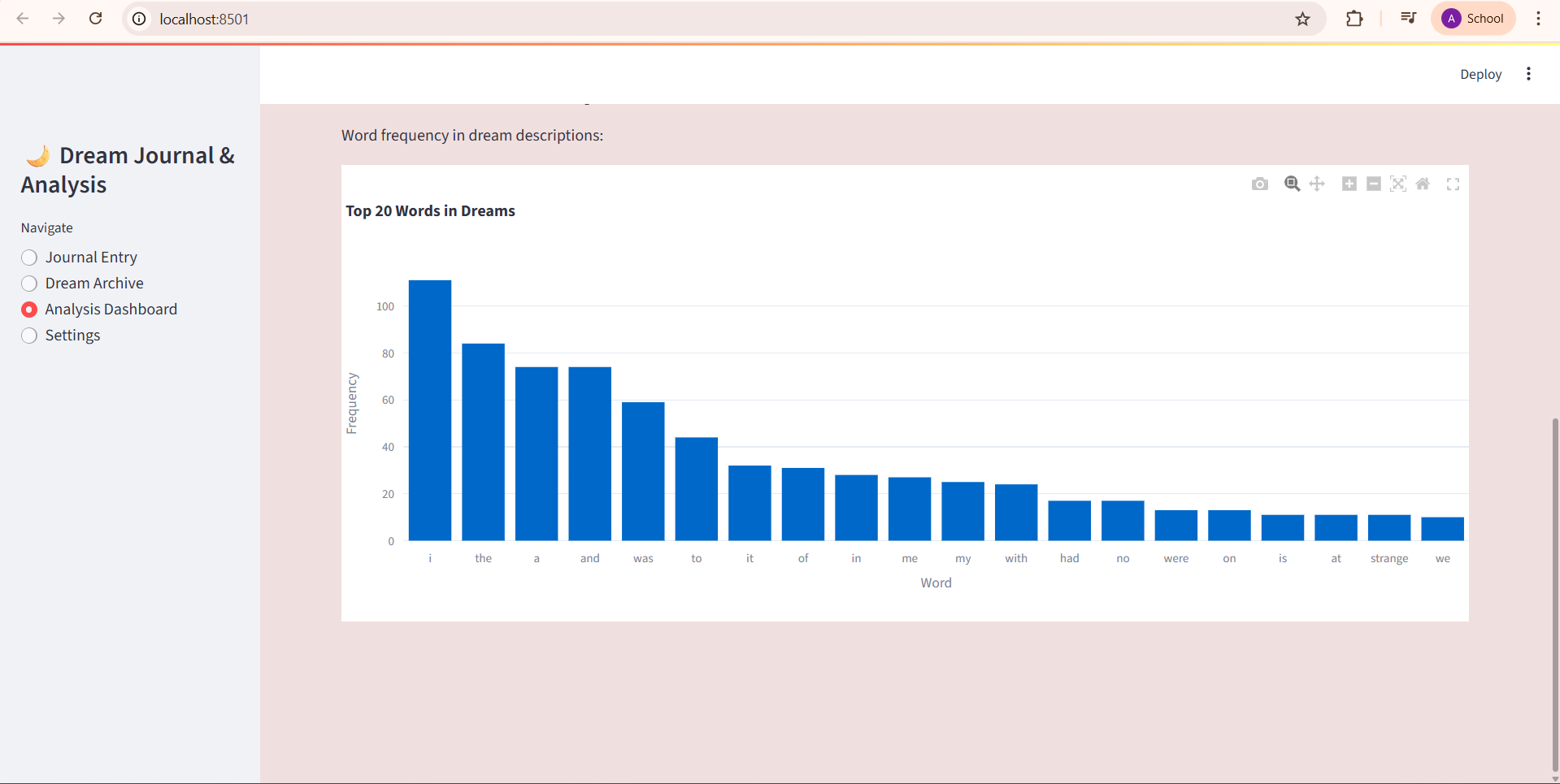
The front end is built using Streamlit and is designed to be responsive and user-friendly. The interface gracefully handles invalid or empty input, ensuring a smooth experience. After each prediction, it displays the top 3 most probable emotions, offering transparency and insight into the model’s decision-making process.

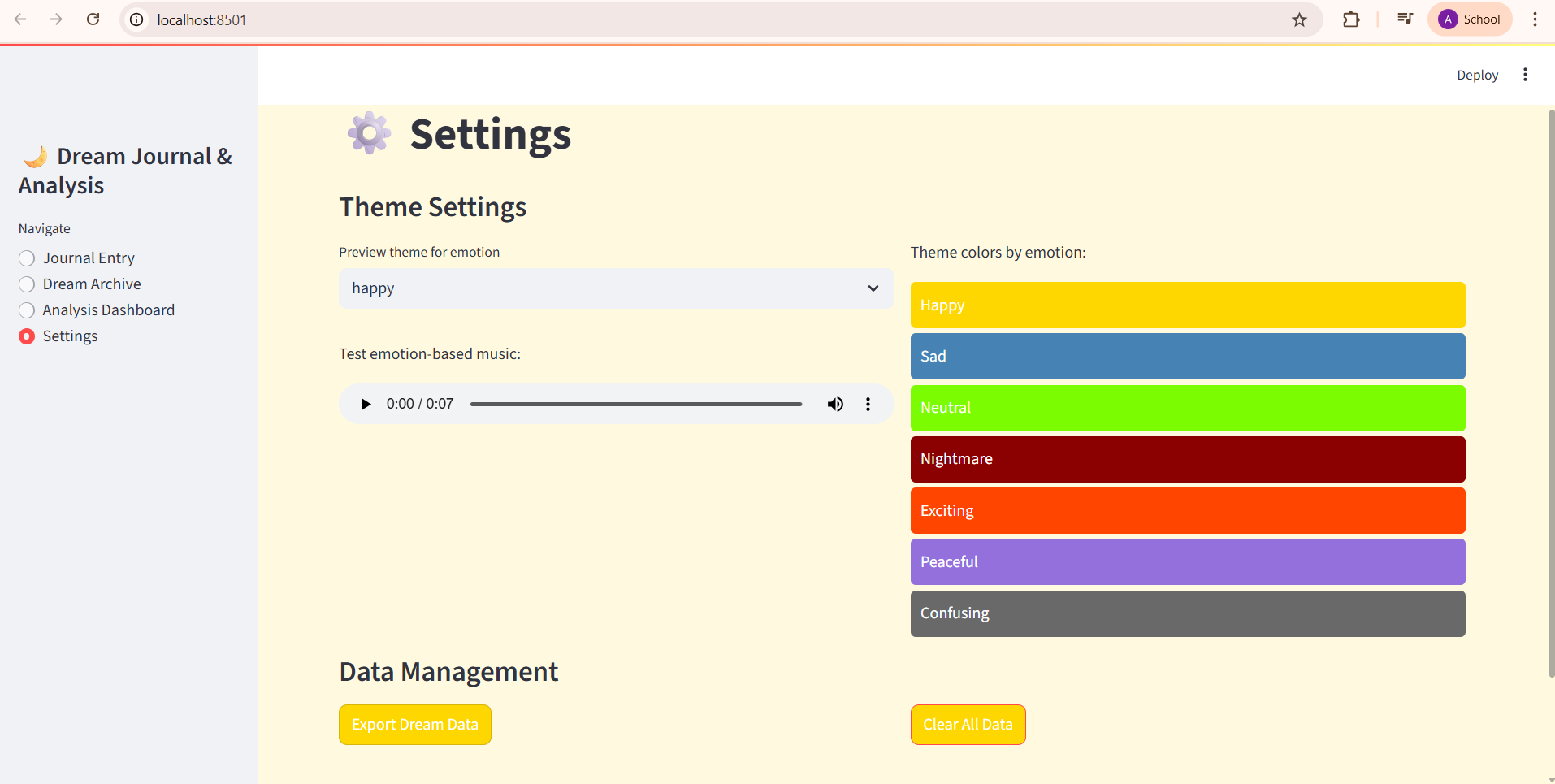


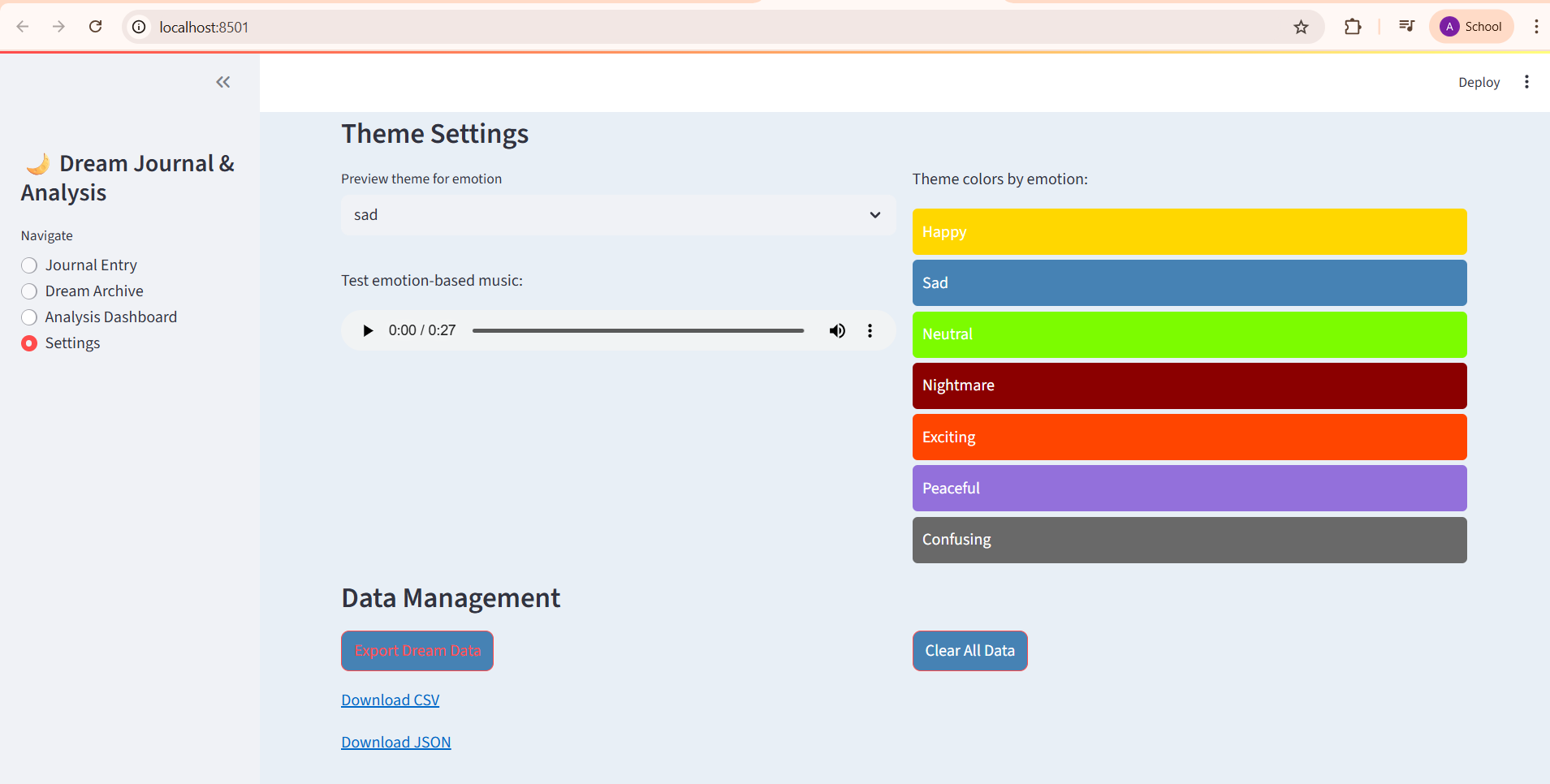












**5. Conclusion & Future Work**

The dream emotion classification system developed in this project demonstrated strong performance, with the blended model achieving an accuracy of **91.3%**. By combining CNN, Naive Bayes, and LightGBM classifiers, the system was able to effectively predict emotions such as happy, sad, neutral, and nightmare from user-submitted dream texts. Additional features, including dream archiving, real-time prediction, and emotion trend visualization, enhanced the overall utility of the application.

However, emotion interpretation remains subjective, and model predictions may not always align with a user's perceived experience. For instance, a dream classified as "neutral" by the system might feel "sad" or "confusing" to the user. To improve alignment with individual emotional perceptions, future work can explore the integration of **Reinforcement Learning (RL)**. By incorporating user feedback as a reward signal, the model could dynamically adjust and learn more personalized emotion mappings over time. This approach may enhance the system’s adaptability, making it more context-aware and user-centric in long-term usage scenarios.

Future developments could also focus on multilingual support, speech-to-text input, and integration with mental wellness platforms to broaden the application’s reach and impact.