Self-Analysis Mental Health Model

Overview

This project aims to develop a Self-Analysis Mental Health Model that predicts potential mental health cond

Features

- Multi-class classification model for mental health condition prediction.
- Dynamic user input collection.
- Label encoding to handle categorical data.
- Handles unseen values in categorical inputs gracefully.
- Model interpretability using SHAP/LIME (if implemented).
- A simple UI or CLI for user interaction.

Project Structure

- mental_health_model.pkl: Trained machine learning model
- label_encoders.pkl: Encoded categorical variables
- class_labels.pkl: Mapping for predicted classes
- predict_mental_health.py: Inference script for making predictions
- mental_health_ui.py: UI implementation using Streamlit
- model_testing.ipynb: Jupyter Notebook for testing
- README.md: Project documentation
- data/: Directory for datasets
  - raw_data.csv: Original datasets used
  - processed_data.csv: Preprocessed datasets
- reports/: Reports and analysis
  - model_performance.pdf: Evaluation results
  - LLM_experimentation.pdf: Findings from LLM-based explanations (if any)

Installation & Setup

Prerequisites

Ensure you have the following installed:

- Python 3.8+
- Required dependencies in requirements.txt

Installation

pip install -r requirements.txt

How to Use

Command-Line Interface (CLI)

To run a prediction using the command-line script:

```
python predict_mental_health.py
```

The script will prompt you for inputs and return the predicted mental health condition.

UI with Streamlit

To run the UI:

```
streamlit run mental_health_ui.py
```

This will launch a web interface where users can enter inputs and receive predictions.

Model Development Process

1. Data Collection: Used publicly available datasets such as "Mental Health in Tech Survey" and "WHO Me
2. Data Preprocessing: Cleaned and normalized data, handled missing values, and performed feature engi
3. Feature Selection: Identified relevant features for prediction.
4. Model Training: Compared models (Random Forest, XGBoost, Logistic Regression, etc.) and selected th
5. Evaluation: Used accuracy, precision, recall, F1-score, and ROC-AUC metrics.
6. Interpretability: Utilized SHAP/LIME for explaining predictions (if implemented).
7. Deployment: Provided CLI and UI for user interaction.

Evaluation Metrics

- Accuracy: Measures overall correctness.
- Precision & Recall: Evaluates class-specific performance.
- F1-score: Balances precision and recall.
- ROC-AUC: Measures model performance in distinguishing between classes.

LLM Experimentation (Optional)

- Implemented an LLM-based approach to provide natural language explanations for predictions.
- Suggests potential coping mechanisms based on the predicted condition.

Video Demonstration

A 5-10 minute video walkthrough covering:

- Code structure and workflow
- Model training and testing
- Sample predictions on test inputs
- How to use the CLI/UI

Contributing

Feel free to submit issues or contribute by making a pull request. Suggestions for improving the model's ac

License

This project is open-source and licensed under the MIT License.

Contact

For any inquiries or support, reach out via GitHub Issues.