Project Report: Mental Health Treatment Prediction in the Tech Industry

Author: Shubham Verma
Deployment URL: mental-health-treatment-in-tech-industry.onrender.com

1. Project Overview

This project aims to predict whether an employee in the tech industry will seek treatment for a mental health condition based on various factors from a workplace survey. By leveraging machine learning, the goal is to create an accessible tool that can provide insights into mental health trends and encourage proactive support within the tech community.

The project encompasses a full data science pipeline, including data cleaning, exploratory data analysis, feature engineering, model training, and hyperparameter tuning. The final, optimized model is deployed as an interactive web application using Flask, allowing users to input their information and receive a real-time prediction.

2. Data Science Pipeline

The project followed a structured data science workflow, from raw data to a predictive model.

2.1. Data Cleaning and Preprocessing

The initial dataset contained inconsistencies, missing values, and irrelevant columns that needed to be addressed. The key steps performed in the data_cleaning.ipynb notebook were:

- **Handling Missing Values:** Null values in columns like self_employed and work_interfere were filled with the mode (most frequent value) to preserve data integrity.
- **Correcting Data Types:** Columns were converted to appropriate data types for analysis.
- **Removing Redundant Columns:** Columns like Timestamp, state, and comments were dropped as they were not relevant for the prediction task.
- **Standardizing Categorical Data:** Inconsistent values within categorical features were standardized (e.g., mapping different representations of gender to a uniform set).

2.2. Exploratory Data Analysis (EDA)

EDA was performed in the exploratory_analysis.ipynb notebook to uncover patterns and relationships in the data. Key insights included:

- **Target Variable Distribution:** The dataset showed a near 50/50 split between employees who sought treatment and those who did not, indicating a well-balanced dataset for classification.
- **Feature Relationships:** Visualizations revealed strong correlations between seeking treatment and factors like having a family history of mental illness, the availability of mental health benefits, and the perceived consequences of discussing mental health at

work.

2.3. Feature Engineering

To improve model performance, new features were created and existing ones were encoded in the feature_engineering.ipynb notebook:

- **Age Grouping:** A new categorical feature, Age_Group, was created to group employees into age brackets, which can be more informative than a continuous age variable.
- **Label Encoding:** All categorical features were converted into numerical format using label encoding, making them suitable for machine learning algorithms.

3. Model Training and Hyperparameter Tuning

Several classification models were evaluated to find the best performer for this task, as documented in model_training.ipynb.

- **Models Tested:** Logistic Regression, Random Forest, XGBoost, and CatBoost Classifier.
- **Hyperparameter Tuning:** GridSearchCV was used to systematically search for the optimal hyperparameters for the top-performing models.
- **Final Model Selection:** The RandomForestClassifier was ultimately chosen as the final model due to its high accuracy and robust handling of categorical features. After tuning, the model achieved a high level of performance on the test set. *(Note: The final train.py script uses a RandomForestClassifier with optimized parameters from the notebook, which is also a strong choice)*.

The best parameters found for the RandomForestClassifier model were:

- bootstrap : True
- max_depth : 20
- min_samples_leaf : 2
- min_samples_split : 5
- n_estimators : 200

The trained model was serialized and saved as trained_model.pkl for use in the application.

4. Prediction Pipeline and Database Integration

A modular and automated pipeline was built to handle data and predictions efficiently.

- **Data Storage:** The cleaned and processed dataset was uploaded to a **MongoDB Atlas** cloud database. The helpers.py script includes logic to check if the data already exists, preventing re-uploads on subsequent runs.
- **Automated Training:** The train.py script connects to MongoDB, retrieves the data, trains the model, and saves the .pkl file. The helpers.py script orchestrates this process, ensuring a trained model is always available before the application starts.
- **Prediction Module:** The predict.py script contains functions to load the saved trained_model.pkl file and make predictions on new input data.

5. Web Application and Deployment

A user-friendly web application was developed using **Flask** to make the model accessible to end-users.

- **User Interface:** An HTML form (form.html) was created to allow users to input their information for all 23 features. The form is styled with CSS to provide a clean, full-screen, dashboard-like experience.
- **Backend Logic (app.py):** The Flask application handles both GET requests (to display the form) and POST requests (to process submitted data). It takes the user's input, structures it into the correct format, and uses the loaded model to return a "Yes" or "No" prediction. The app also includes error handling for invalid input.
- **Deployment:** The application was deployed to the cloud using **Render**, a free hosting service. The deployment process involved:
  1. Creating a requirements.txt file to list all necessary Python packages.
  2. Modifying app.py for a production environment by removing the development server block.
  3. Pushing the entire project to a GitHub repository.
  4. Connecting the repository to Render and configuring the build and start commands (pip install -r requirements.txt and gunicorn src.app:app).

The final, live application is accessible at the provided URL, successfully meeting the project's deployment requirements.

6. Conclusion

This project successfully demonstrates an end-to-end machine learning workflow, from raw data to a deployed web application. The final model can accurately predict whether a tech employee is likely to seek mental health treatment, serving as a valuable proof-of-concept for data-driven mental health awareness tools in the workplace.