Predicting Mental Health Treatment in the Tech Industry

Welcome to the Mini-Hackathon, organized by PWSkills.This bi-monthly Hackathon fosters innovation and technical expertise in data-driven solutions. This edition focuses on leveraging machine learning to address **mental health challenges in the tech industry**, aiming to develop accurate and interpretable models that can predict whether an individual has sought treatment for mental health issues based on survey responses..

1. Problem Statement

In tech workplaces, mental health is increasingly recognized as crucial yet stigmatized. The goal of this hackathon is to build a model that **predicts whether an individual has sought treatment for a mental health condition** based on their responses to survey questions about demographics, workplace culture, attitudes, and support systems. The model should be accurate, interpretable, and useful for informing policy or workplace intervention.

2. Dataset — Explanation

Dataset:
https://drive.google.com/file/d/1oudxpap1iR8Xg7GBpAzB6KVPzIHA5RD4/view?usp=drive_link

This survey captures responses from individuals in the tech sector about mental health, workplace environment, and attitudes.

Key features:

- **Demographics**: Age, Gender, Country, State (for US respondents)
- **Work status**: Self-employed or not; company size (number of employees)
- **Workplace and support environment**:
    - Remote work ≧ 50% time
    - Whether working for a tech company
    - Knowledge of benefits / care options for mental health from employer
    - Whether employer has wellness programs or resources
    - Anonymity protections, leave policies for mental health
- **Attitudes / Consequences:**
    - Willingness to discuss mental health with coworkers, supervisors, or in interviews
    - Perceived negative consequences of disclosing mental health issues
    - Physical vs mental health consequences perceptions
- **Target Variable:** treatment — whether the respondent has sought treatment for a mental health issue.
- **Data Format & Size:**
- ~1,200+ survey respondents.
- Mix of categorical, ordinal, Boolean features; some free text/comments.
- Some missing values / noisy inputs (e.g. in Age, Gender, etc.).

3. Tasks & Deliverables

1. **Data Exploration & Preprocessing**
    - Understand feature distributions, missingness, outliers.
    - Clean and standardize features (e.g. harmonize gender responses, correct ages).
    - Encode categorical variables appropriately.
2. **Feature Engineering & Selection**
    - Engineer new features if helpful (e.g. age groups, combining related features).
    - Select features that have predictive power while ensuring interpretability.
3. **Modeling**
    - Train classification models to predict `treatment` (Yes / No).
    - Compare multiple algorithms (e.g., Logistic Regression, Random Forest, Gradient Boosting Machines, possibly simpler models for interpretability).
    - Hyperparameter tuning.
4. **Interpretability / Explainability**
    - Identify potential biases (e.g. by gender, country, etc.).
5. **Deployment**
    - Build a small interface or dashboard to input features and see predicted probability.
    - Visualizations or summaries that could help non-technical stakeholders understand the results.

5. Tech Stack & Tools

Here are recommended tools and libraries:

- **Language**: Python
- **Notebook env**: Jupyter, Colab
- **Data processing & analysis**: Pandas, NumPy
- **Visualization**: Seaborn, Matplotlib, Plotly
- **ML Algorithms**: Scikit-learn, XGBoost, LightGBM, CatBoost
- **Deployment / UI**: Streamlit or Flask / FastAPI for prototype dashboard

6. Submission Guidelines & Structure

Participants must submit their work via GitHub or Google Drive to ensure accessibility and ease of evaluation. Submissions should include the components listed below, organized in a clear folder structure to support both experimentation and deployment of a Flask-based application. Additionally, participants are encouraged to deploy their application on one of the specified platforms (AWS, Render, Vercel, or Hugging Face Spaces) to demonstrate real-world applicability.

Submission Components

- **Code:** A well-organized script or Jupyter Notebook containing the complete solution (data preprocessing, model training, prediction), alongside modular code for a Flask-based application.
- **Model Output:** A .csv file with predictions on the test set (if provided) in the specified format.
- **Documentation:** A detailed report (see Section 6) explaining the approach, methodology, and findings.
- **README:** A file explaining how to run the code and deploy the application, including dependencies, instructions, and platform-specific details.
- **Deployment Link:** A working URL to the deployed Flask application (hosted on AWS, Render, Vercel, or Hugging Face Spaces), demonstrating model predictions.

Folder Structure Suggestion

```
project_root/
├── Notebooks/
│   ├── data_cleaning.ipynb
│   ├── exploratory_analysis.ipynb
│   ├── feature_engineering.ipynb
│   ├── model_training.ipynb
├── src/
│   ├── __init__.py
│   ├── data/
│   │   ├── preprocess.py
│   ├── models/
│   │   ├── train.py
│   │   ├── predict.py
│   ├── utils/
│       ├── helpers.py
│   ├── app.py     # for web / dashboard if any
├── data/
│   ├── raw/       # original data
│   ├── processed/  # cleaned / processed data
├── models/
│   ├── trained_model.pkl  # or appropriate serialization
├── requirements.txt
├── README.md
├── report.pdf (or .md)
├── predictions.csv
├── (optional) deployment_link.txt or URL in README
```

7. Evaluation Criteria

Submissions will be evaluated based on the following criteria, ensuring a balance between technical performance, code quality, and practical deployment.

Accuracy: The model's predictive performance on the test set.
Code Quality:

- Clarity: Code should be well-commented, modular, and easy to understand, with clear separation of concerns (e.g., preprocessing, training, prediction).
- Reproducibility: Code must run without errors when executed with the provided instructions in the README.md, including local execution and deployment setup.
- Efficiency: Efficient use of computational resources, with optimized preprocessing, model training, and prediction steps suitable for deployment.

Deployment:

- Working Link: A mandatory working URL to the deployed Flask application (hosted on AWS, Render, Vercel, or Hugging Face Spaces), allowing evaluators to test predictions.
- Functionality: The deployed application must correctly serve predictions based on input features, demonstrating integration of the trained model.
- Documentation: The quality and clarity of the participant's report (see Section 6), including a clear explanation of the methodology and deployment process.
- Innovation: Creative approaches to feature engineering, model selection, visualization of results, or deployment strategies that enhance the solution's effectiveness or usability.

8. Documentation Requirements

Participants are required to submit a detailed report (PDF or Markdown) alongside their code. The documentation should provide a clear narrative of the approach and findings, accessible to both technical and non-technical audiences. The report should include:
Introduction: A brief overview of the problem and its significance in the context of disease prediction and the scope of your solution.
Methodology:

- Data preprocessing steps (e.g., handling missing values, encoding categorical variables).
- Feature engineering techniques (e.g., creating interaction terms, scaling features).
- Model selection and justification (e.g., why you chose a specific algorithm).
- Hyperparameter tuning process (if applicable).
- Document the procedure you have followed for deployment.

Results:

- Model performance (accuracy on validation/test sets).
- General insights derived from the data or model.

Discussion:

- Challenges faced and how they were addressed.
- Limitations of the approach and potential improvements.
- Real-world implications of the solution, especially regarding early diagnosis.

Conclusion: A summary of key findings and takeaways.

Formatting Guidelines:

•Use clear headings and subheadings.
•Include visualizations to support your findings.
•Keep the report concise (recommended: 4–5 pages).