Predicting Student Depression Using Machine Learning

Research Question:

What personal, academic, and lifestyle-related factors best predict depression among university students, and how accurately can machine learning models identify students at risk based on these features ?

Amit Dvash, Yuval Livshits, Guy Korenfeld, Tom Badash, Einav Cohen

May 31, 2025

1 Methodology

In this project, we aimed to predict depression levels among students using machine learning models. The dataset included responses from 27,901 students, each represented by 18 attributes, including demographic details (e.g., age, gender, city), academic and work-related pressures, CGPA, study and job satisfaction, sleep duration, dietary habits, degree pursued, suicidal thoughts, work/study hours, financial stress, family history of mental illness, and a binary indicator of depression status (1 for depressed, 0 for not depressed).

To assess which modeling approach would provide the most reliable prediction, we implemented and compared several machine learning models:

1. Random Forest Classifier
2. XGBoost Classifier
3. Neural Network (Keras Sequential Model)
4. Ensemble Model (Manual Soft Voting of RF + XGBoost + NN)

Problem-Solving Pipeline

Data Analysis

To better understand the structure and potential issues in the dataset, we performed exploratory data analysis (EDA) across all features.

- **Missing Values:** The dataset is largely complete, with only three missing entries in the *Financial Stress* column, which were imputed based on the mode.
- **Categorical Features:** Several categorical variables showed wide variation in cardinality. For instance:

  - *City* had over 50 unique values.
  - *Degree* covered a diverse range of academic programs.
  - *Gender*, *Family History of Mental Illness*, and *Suicidal Thoughts* were binary or low-cardinality features.

- **Target Variable Distribution:** Approximately 60% of students were labeled as depressed, indicating a moderate class imbalance. This was considered when selecting models and evaluation metrics.

- **Demographic Insights:**
  - The dataset included a broad age range, with most students between 18 and 35.
  - Gender distribution was moderately imbalanced, with a higher number of male students.
  - Cities such as Kalyan, Srinagar, and Hyderabad had the highest student representation.

- **Feature Correlations with Depression:** We found strong and meaningful associations between depression and several key variables:
  - *Academic Pressure:* Depression prevalence increased sharply with higher academic pressure ratings.
  - *Financial Stress:* Students reporting high financial stress were far more likely to experience depression.
  - *Dietary Habits:* Those with unhealthy diets had a significantly higher likelihood of being classified as depressed.
  - *Study Satisfaction:* Students reporting low satisfaction with their academic experience were more likely to report depression.
  - *Family History of Mental Illness:* Students with a family history showed elevated rates of depression.
  - *Suicidal Thoughts:* Among the strongest predictors — students who reported suicidal thoughts overwhelmingly belonged to the depressed class.

Data Preprocessing

We performed a series of preprocessing steps to clean and standardize the dataset for machine learning models.

- **Missing Values:** The only column with missing values was *Financial Stress*, which had three missing entries. These were imputed using the column's mean value to preserve distributional integrity.

- **Sleep Duration:** Originally stored as textual ranges (e.g., "5-6 hours"), this column was mapped to numeric midpoints:
  - "Less than 5 hours" $\rightarrow$ 4.5
  - "5-6 hours" $\rightarrow$ 5.5
  - "7-8 hours" $\rightarrow$ 7.5

  - “More than 8 hours” → 9

  Rows with missing sleep duration values were removed.

- **Dietary Habits:** The categorical values “Healthy”, “Moderate”, and “Unhealthy” were mapped to scores of 3, 2, and 1 respectively. Missing entries were dropped.
- **Binary Variables:**
  - The columns *Suicidal Thoughts* and *Family History of Mental Illness* were originally in “Yes”/“No” format and were mapped to 1 and 0 respectively.
  - *Gender* was encoded as 1 for male and 0 for female.
- **City Encoding:** The *City* column was encoded using frequency counts to preserve ordinal representation while retaining model compatibility.
- **Degree Encoding:** The wide range of academic degrees was categorized into three levels:
  - “Higher Secondary” for students reporting “Class 12”
  - “Graduated” for Bachelor’s level programs (e.g., BSc, B.Tech)
  - “Post Graduated” for Master’s or higher programs (e.g., MSc, M.Tech, PhD)

  These categories were then mapped to ordinal values: 1, 2, and 3 respectively.

Feature Engineering

To enhance model performance and incorporate domain knowledge, we performed feature engineering on several attributes:

- **Lifestyle Score:** We created a composite feature called *Lifestyle Score* to represent a student’s overall well-being. It combines three health-related variables:
  - **Sleep Duration** (positive weight): Indicates sleep health.
  - **Dietary Habits** (positive weight): Reflects nutritional quality.
  - **Financial Stress** (negative weight): A known stressor linked to mental health.

  The formula used:

  $$\text{Lifestyle Score} = 0.4 \cdot \text{Sleep Duration} + 1.5 \cdot \text{Dietary Habits} - 0.8 \cdot \text{Financial Stress}$$

- **Final Dataset Validation:** After all preprocessing and feature engineering steps, we verified that the final dataset contained no missing values across any column.

Model Building

Three primary machine learning models were trained and evaluated:

- **Random Forest:** A tree-based ensemble model trained using stratified data split and optimized with grid search over key parameters like tree depth, number of estimators, and feature subset selection.
- **XGBoost:** A gradient-boosted tree model trained with random search over learning rate, subsample ratio, max depth, and number of estimators. Evaluation metrics demonstrated strong generalization capacity.
- **Neural Network:** A Keras-based sequential deep learning model with dense layers, dropout regularization, and binary cross-entropy loss. The model used early stopping to prevent overfitting and was trained on scaled input data.
- **Ensemble:** We implemented a manual soft voting ensemble combining predicted probabilities from Random Forest, XGBoost, and the Neural Network. The average of the predicted probabilities from all three models yielded a final prediction.

Parameters Tuning

Each model underwent hyperparameter tuning using appropriate strategies:

- **Random Forest:** GridSearchCV with 3-fold cross-validation was applied to tune the number of estimators, tree depth, minimum samples per split/leaf, and max features.
- **XGBoost:** RandomizedSearchCV was used with 20 iterations across hyperparameters including learning rate, subsample, column sampling, tree depth, and regularization term (γ).
- **Neural Network:** Architectural tuning included experimenting with layer sizes, dropout rates, and optimizer configurations. Early stopping was used to halt training once validation loss plateaued.

Pipeline Automation

To ensure reproducibility and maintainability, the entire ML pipeline was built to support:

- Stratified splitting for consistent class distribution.
- Scalable training on structured NumPy/Pandas input.
- Unified API for fitting, predicting, and evaluating across all models.
- Seamless integration of preprocessing, model training, evaluation, and ensembling in a single, reusable notebook structure.

Why This is the Best Solution

1. The pipeline is **modular**, allowing individual components like data cleaning, model tuning, and evaluation to be updated independently.
2. It is fully **reproducible**, with fixed random seeds and consistent splits.
3. The structure is highly **scalable**, with support for large datasets and multiple modeling strategies.
4. Our approach **outperforms baseline models** by combining strengths of tree-based and deep learning models using ensemble voting.
5. The final ensemble model is **deployment-ready**, offering a robust balance between performance and efficiency.

Software/System Implementation

- **Programming Language:** Python 3.9.13
- **Environment:** Visual Studio Code.
- **Version Control:** Codebase managed via GitHub.

Engineering Attributes

- **Dependencies:** Listed in `requirements.txt`
- **Reproducibility:** Fixed seeds and ensured consistent data splits.
- **Modularity:** Although implemented in a single notebook, the code was organized into clearly separated sections: data preprocessing, feature engineering, model training, and evaluation — enabling easy maintenance and readability.
- **Scalability & Efficiency:** The notebook was tested on a dataset with nearly 28,000 samples and ran efficiently using standard hardware, demonstrating practical scalability for medium-sized real-world datasets.

2 Evaluation

We evaluated our models using both performance metrics and interpretability methods to compare model effectiveness and gain insights into feature contributions.

Evaluation Metrics

We used the following key metrics for model comparison:

- **Accuracy:** Overall correctness of the model.
- **F1-Score:** Harmonic mean of precision and recall; especially important in slightly imbalanced datasets.

- **ROC-AUC:** Area under the ROC curve, showing true positive vs. false positive rates.
- **Precision-Recall AUC:** Better reflects performance on imbalanced data.
- **Confusion Matrices:** Provide a breakdown of TP, FP, FN, and TN to understand the types of prediction errors.

The neural network achieved the highest AUC (0.93), followed by the ensemble model (0.91). XGBoost and Random Forest models both achieved 0.80–0.82 AUC.

- **Neural Network:** Highest accuracy and F1-score (0.85), strong recall, and lowest false negatives.
- **Ensemble (Soft Voting):** Balanced performance with strong precision and recall (accuracy 0.82).
- **Random Forest:** Solid baseline, precision 0.84, recall 0.79, accuracy 0.85.
- **XGBoost:** Similar performance to RF with slightly lower recall.

ROC and Precision-Recall Curves

The ROC and PR curves reinforce the neural network's superiority in distinguishing between classes. The ensemble model also demonstrates high separability, indicating value in combining model predictions.

Feature Importance (Permutation-Based)

We computed permutation-based feature importances to understand which variables contributed most to model predictions.

- Across all models, **Suicidal Thoughts** was the most predictive feature.
- **Lifestyle Score**, derived from sleep, diet, and stress, ranked highly in RF and XGBoost.
- **Academic Pressure** had particularly high importance in the Neural Network model.
- **Family History of Mental Illness**, **Financial Stress**, and **Study Satisfaction** also had meaningful influence across models.

Neural Network Wrapper for Interpretability

To enable permutation importance with the Keras model, we implemented a custom wrapper class `PredictAsBinary` that conforms to the `BaseEstimator` and `ClassifierMixin` interfaces from `sklearn`. This allowed us to extract reliable importance values while preserving the NN's probabilistic output and threshold-based prediction logic.

System-Level Observations

In addition to traditional evaluation metrics, we assessed system-level performance to understand each model's practicality and efficiency in real-world deployment.

- **Training Time:**
  - Random Forest: 3 minutes and 42 seconds
  - XGBoost: 1 minute and 19 seconds
  - Neural Network: 1 minute and 3 seconds

  The neural network benefited from early stopping and vectorized training, while XGBoost offered a good balance between speed and performance. Random Forest was the most computationally intensive during hyperparameter tuning.

- **Inference Speed:** All models returned predictions for over 5,000 samples in under a second, enabling real-time or near real-time inference capability.

- **Model Size:** All trained models were relatively lightweight and suitable for deployment in resource-constrained environments. The neural network had slightly higher memory usage due to its dense architecture.

- **Reproducibility:** Fixed random seeds and stratified splits were applied across all models, ensuring consistent and repeatable results across runs.

- **Modularity and Maintainability:** Despite being implemented within a single notebook, the project structure allowed clean transitions between preprocessing, model training, and evaluation. This streamlined experimentation and comparison across multiple models.

- **Pipeline Efficiency:** End-to-end integration of feature engineering, model fitting, evaluation, and visualization was achieved in a unified, reusable structure—enabling efficient iterations and extensibility.