Depression Risk Prediction among Students in India

Introduction and Problem Context

I have always been very concerned about students' mental health, especially in the current situation of increasing academic pressure, lifestyle imbalance and emotional distress. My project aims to explore the possibility of depression based on students' daily life data (such as sleep time, stress, study habits and family history).

My goal is to develop a model to help institutions identify high-risk individuals early and effectively allocate mental health resources. By analyzing students' data on academic stress, sleep, diet, learning satisfaction and emotional indicators such as suicidal thoughts, I hope to reveal the patterns that lead to depression in the academic environment.

Data Overview

Link: https://www.kaggle.com/datasets/hopesb/student-depression-dataset/data

I studied a survey dataset from Indian students. It contained nearly 28,000 responses, with columns covering everything from academic performance and lifestyle habits to emotional health and mental health history. Each row represented a student, with characteristics including age, grade point average (CGPA), sleep duration, eating habits, and whether they had suicidal thoughts. All of this gave me a more complete picture of the various stressors and behaviors that may be associated with depression.

Luckily, there were no missing values in any of the major columns, so I didn't have to spend too much time cleaning the data. The only thing I removed was the "ID" column since it didn't carry any meaningful information.

Initial Observations and Stats

Before I started the visualization, I reviewed some basic data. For example, the average CGPA for all students was about 7.66 (out of 10), which seems to be quite common in Indian universities. When I looked at financial stress, academic stress, and study satisfaction, they were all rated on a scale of 0-5, and interestingly, the averages of all three hovered around 3. This suggests that most students reported moderate levels of stress and satisfaction, with nothing too extreme in between.

Distributions

To get a better understanding of the data, I plotted histograms and boxplots for these stress-related variables. The distributions for most variables were pretty well shaped, without too many weird outliers — except for CGPA, which had one or two. I decided to keep these outliers

because they might reflect real-world situations. After all, not everyone fits into a strict average. I then looked at categorical data like sleep duration and eating habits using pie charts. These distributions were fairly balanced, which is good — it means my dataset covers a wide range of behaviors, rather than being biased towards one lifestyle type.

However, when I looked at the type of degree the students were pursuing, I noticed that most of them were in high school or undergraduate. This means that the model I built might be more suitable for younger students, and less applicable to older groups like graduate students or PhDs.

Correlations to the Target: Depression

Next, I wanted to get a clearer picture of which characteristics might be most associated with depression risk, so I created a correlation matrix. The results showed that two characteristics had the strongest positive correlations with depression: academic stress had a correlation of 0.47, followed by financial stress at 0.36. These two characteristics are the highest in the matrix, but it's important to note that they are only moderately correlated at best. This suggests that these factors have some association with depression, but definitely not all. Other characteristics, such as academic satisfaction, had a smaller negative correlation of -0.17, while age had a slightly stronger negative correlation of -0.23, suggesting that younger students might be at greater risk. So while these numbers give me a good starting point, they also remind me that a simple regression model may not capture all the complexities. I'll still try to use this to establish a baseline, but I'm already considering more flexible models like decision trees or KNNs that might better handle the nonlinear relationships I suspect are present.

Then, by using a box plot, it shows that another strong indicator is age - younger students tend to be more vulnerable, which may reflect the transition stress they experience. I also looked at sleep duration, and as expected, students who slept less than 5 hours per night had higher odds of depression. Therefore, I am confident that these two features can be included in all of my models. When it comes to diet, students with unhealthy eating habits have significantly higher odds of depression. Finally, I looked at academic satisfaction, and there was no significant difference between depressed and non-depressed students. This is a bit surprising but it suggests that simply being satisfied with learning may not be enough to protect people from feeling overwhelmed or mentally unwell.

One of the strongest features was a binary indicator - suicidal thoughts. Students who reported suicidal thoughts were more likely to be depressed. This was one of the clearest dividing lines in the data. This really got me thinking about trying a decision tree based model in the future. Since some of these features act like yes/no switches, perhaps a tree model would be able to capture these features better than a simple logistic regression model.

Then, when I look at the “Family History of Mental Illness” feature, I expect it to be more strongly associated with depression, but the data reveals a very different picture. As the bar chart shows, the number of cases of depression is high for students both with and without a family history. This suggests that family history alone is not independently predictive of depression in this dataset. Therefore, while it may still play a role as part of a larger feature combination, I would not rely too much on it. This is a reminder that even features that seem important logically may not necessarily show strong predictive power in practice.

Summary on EDA

This EDA gave me a better understanding of what areas to focus on. Features such as academic stress, sleep, diet, financial pressure, age, and suicidal ideation seem particularly valuable. On the other hand, some features, such as learning satisfaction and family history, may not be that important. I plan to start with logistic regression as a baseline, and then try KNN and decision trees to see if they can capture more subtle patterns. Ultimately, my goal is to build a system that can not only accurately predict the risk of depression, but also be easy to explain and ethical.

Logistics Regression Model

Since my goal is to predict whether a student has depression or not — the outcome is either “yes” or “no” — logistic regression is a natural starting point for me. Not only is it easy to implement, but it also gives interpretable results: I can actually see to what extent each feature (such as academic stress or suicidal thoughts) increases or decreases the chances of having depression.

First Try – Keep It Simple

I started with a super simple model using just one feature: Academic Pressure. It was one of the strongest predictors from my EDA, so I wanted to see how well it could perform on its own. After splitting the data 80/20 and training the model, I got a training accuracy of ~72.8% and a test accuracy of ~73.0%. That’s a solid baseline—better than random guessing and even better than just picking the majority class, which had a baseline of about 58.6%. This told me that Academic Pressure alone could capture something meaningful about the likelihood of depression. But I knew I could push this further by adding more features that came up during EDA.

Second Try – Add Strong Predictors

I brought in more features that stood out earlier: Financial Stress, Age, Sleep Duration, Dietary Habits, and Suicidal Thoughts. I also spent time cleaning and encoding these columns by mapping "Yes/No" to 1/0, “one-hot encoding” categorical values like sleep and diet, and then building pipelines for both numeric and categorical features. With the complete pipeline in place, I retrained the model and the test accuracy jumped to 83.1%. This is a huge improvement over the simple single-feature version. I also looked at the report and both precision and recall looked

higher, meaning that the model was doing a pretty balanced job of capturing recall without generating too many false positives.

Third and Fourth Tries – Fine-Tuning Features
After that, I got curious about what happens if I slightly tweak the feature set? So I ran another version with CGPA and Study Satisfaction added in (they weren't strongly correlated, but I figured they might still help). Accuracy rose slightly again to 83.4%. After that, I added all the features in the dataset (except ID), including work/school time and family history. This version became my most complete logistic model: accuracy reached 83.9%, precision reached 85.1%, and recall reached 87.7% (the highest among them). I like that the recall remains high, which means that the model is able to identify students who are truly depressed well.

Overall, I was impressed with how far logistic regression could go. Starting from a simple one-variable model at 73% accuracy, I was able to climb to almost 84% just by smartly adding features. But I also noticed some limitations. Even though performance was solid, logistic regression assumes a linear relationship between features and the log-odds of the outcome. And with this kind of psychological data, I started to suspect there were nonlinear patterns that the model wasn't quite capturing. That's why I decided to try more flexible models like KNN, decision trees, and eventually neural networks to see if they could uncover patterns that logistic regression might be missing.

KNN
Now, I wanted to explore a model that doesn't assume a linear relationship between the predictors and the target. That's where K-Nearest Neighbors (KNN) came in. The idea behind KNN is pretty intuitive: when a new student comes in, the model looks at similar students in the dataset (their "neighbors") and predicts their depression status based on what those nearby students experienced. This makes a lot of sense in a mental health context. Depression can be very nuanced, and it's possible that certain combinations of features, like high academic pressure, low sleep, and unhealthy diet, tend to "cluster" together. KNN can pick up on those patterns more naturally than a linear model.

First KNN Try
To be fair, I used the same set of features as my best logistic regression model: e.g. academic stress, financial stress, sleep duration, suicidal tendencies, etc.Then, I built a similar pipeline of scaling, encoding, and preprocessing, then ran GridSearchCV to tune the n_neighbors hyperparameter. I tested different k values from 1 to 29 and found that k = 25 gave the best performance. With this, I got a test accuracy of 83.2%, which is similar to my logistic regression model. But more importantly, the recall improved to 87.7%, meaning it also did a better job of capturing true cases of depression.

Second KNN Try – Prioritizing Recall
Since recall was my top priority (I'd rather have a few false alarms than miss students who actually need help), I adjusted my model to focus more on that metric. I used StratifiedKFold for better class balance and switched my GridSearchCV scoring from accuracy to recall. Again, the best k ended up being 25, and the recall improved even more, up to 88.2%. That means out of all the students who truly were depressed, my model correctly identified 88.2% of them. Precision also held up pretty well at 84.1%, which means the model wasn't just flagging everyone randomly—it was still making thoughtful, targeted predictions. So far, this is the best model.

Comparing to Logistic Regression
Comparing with my best logistic model (which had recall around 87.7% and precision at 85.1%), KNN had slightly better recall (88.2%). The accuracy of KNN was also really solid at about 83.3%, just a little under my logistic model. But what sealed the deal for me was the false negative rate—in other words, how often the model missed someone who was actually depressed. KNN had fewer false negatives, which is crucial in this kind of problem. Missing someone in need is a much bigger issue than flagging someone who turns out to be okay. Thus, it is clear that the second try KNN model is the best.

Decision Tree Models
After trying out logistic regression and KNN, I became interested in decision trees. One thing I really like about this model is its interpretability. Rather than just giving a prediction, it actually shows the path that led to the decision. This kind of transparency is crucial in sensitive fields like mental health, where understanding why the model made a certain prediction is crucial.

First Tree
For my first try, I built a basic decision tree using the same features. I limited the max depth to 4 just to prevent overfitting and keep the model easy to read. The performance was okay: Train accuracy was about 82.4% and test accuracy was around 81.1%. So I wasn't overfitting, but also wasn't outperforming KNN or logistic regression. Still, what really caught my attention was the recall, which hit 88.1%—that's pretty solid and close to what KNN gave me.

I also checked the feature importance, and sure enough, the tree heavily relied on just a few variables—Suicidal Thoughts, Academic Pressure, and Financial Stress made up nearly all of the decision power. Other features barely mattered, which was interesting to see.

Second Tree
To improve the model, I used GridSearchCV to tune the max_depth and min_samples_leaf parameters. This allowed the model to find the sweet spot between being too shallow (underfitting) and too deep (overfitting). The best combo turned out to be a depth of 3 with a single minimum sample per leaf. The model is simpler while almost keeping the same recall rate

at 88.1%. However, the overall test accuracy and precision went a bit lower (dropped from 83% to 81%). Overall, even the recall was nice, the model still lagged a bit behind KNN in general performance.

Third Try
In my third attempt, I tried something more advanced—RandomizedSearchCV. I widened the search space and added more hyperparameters like max_features and class_weight to help with class imbalance. I also kept the scoring focused on recall, since that's what matters most in identifying depressed students. This tuning worked—recall shot up to 97.5%, which is incredibly high. The model was catching nearly every single depressed student. But here's the catch: precision dropped to about 63%, and accuracy also fell to around 65%. This told me the model was overdoing it and it flags way too many students as depressed. So while it had the highest recall, the trade-off wasn't worth it.

Overall, the decision tree gave me solid insights into which features mattered most and how predictions were made. It was easy to visualize and helped confirm what I saw in my EDA but when it came to balancing recall, precision, and overall performance, it just couldn't beat my tuned KNN model.

Neural Network Model
After running models like logistic regression, KNN, and decision trees, I was curious if neural networks could do better — especially since they are designed to capture complex, nonlinear patterns in data. Mental health is not black and white, and I suspected there might be some interactions between features that simple models can't capture.

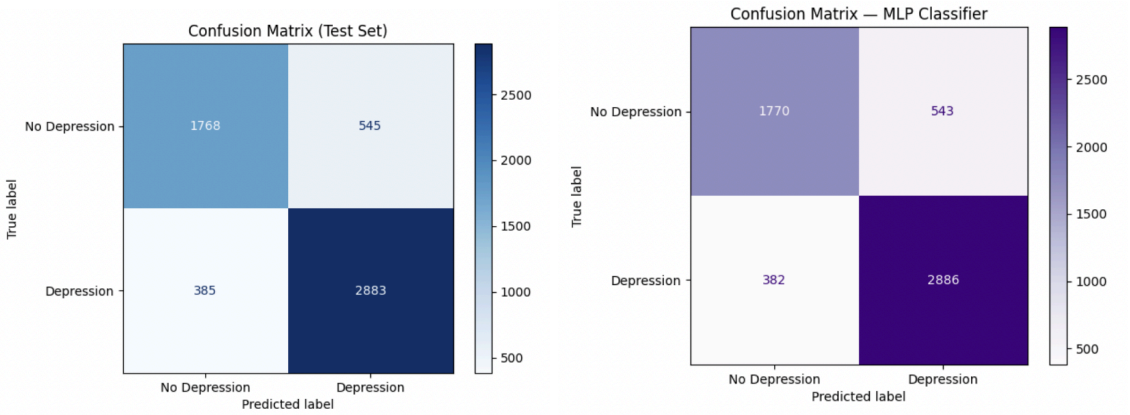
First Try - with PyTorch
After testing traditional models, I wanted to challenge myself and build a neural network using PyTorch. I think depression prediction might benefit from a more flexible model. So, instead of using a pre-built classifier, I constructed and trained a network using tensors and modules from PyTorch. Once I preprocessed the data (scaling, encoding, etc.), I converted everything to PyTorch tensors and trained the model over 100 epochs. I also monitored the loss decreasing over time. After training, I evaluated it on the test set and got: Test Accuracy: 84.3%; Test Precision: 86.0%; Test Recall: 87.5%. These results were very strong, and the model actually outperformed KNN on precision and accuracy. However, looking at recall, my KNN pipeline acctually can give me a stronger recall at 88.2% with precision aims at 84.1% but having minimal code. So I had to ask myself: is PyTorch worth it for this task? In this case, probably not: the performance gain was small, and the added benefit in precision wasn't the most important thing in this case. Thus, I'll still think KNN model is the best. But if I work with more complex inputs, like raw text responses from surveys or journal entries, I'd definitely consider PyTorch again.

Second Try - with MLPClassifier

Later, I explored a different neural network approach using scikit-learn's MLPClassifier. (I learned this from my tutor) The main benefit here was simplicity—I didn't need to worry about manual training loops, optimizers, or tensor conversions. After training the model and running it on the same test set, it performed slightly better than the PyTorch one in some areas: **Test Accuracy: 83.4% (higher than KNN of 83.3%); Test Precision: 84.2% (higher than KNN of 84.1%); Test Recall: 88.3% (higher than KNN of 88.2%)**. All metrics performed better than using PyTorch and my past KNN model (all a bit higher).

Which Model Wins?

Now, all these models are left with 2 models to compare. They're very similar. (See pics below) Left is KNN and the right is neuro network.



Although both the KNN and MLP neural network models performed incredibly well—and honestly, their metrics were shockingly close:

Details to compare:

- KNN Recall: 88.2%
- MLP Recall: 88.3%
- KNN Precision: 84.1%
- MLP Precision: 84.2%
- KNN Test Accuracy: 83.3%
- MLP Test Accuracy: 83.4%

So yes, the difference was less than 0.1% in each metric. If I'm only looking at numbers, I might say they're equally good. But what nudged me toward choosing the MLP neural network as my final model came down to 2 things:

1. Handling Complex Patterns: MLP is a flexible model that can capture non-linear interactions between features, something that's very relevant when trying to model something as complex and nuanced as mental health. Even if my current dataset is tabular and not too large, this flexibility might be why it squeezed out that extra performance.
2. Scalability for Future Use: If I ever scale this project, by adding more features, raw text data, or increasing sample size, MLPs are better positioned to handle that growth. The performance advantage may seem small now, but it could become more significant as the problem gets more complex.

Summary

In conclusion, after I explored multiple models to predict student depression, including logistic regression, KNN, decision trees, and neural networks, the MLP neural network outperformed across all metrics, making it my final model of choice. For next steps, I'd like to experiment with larger datasets, include more diverse inputs like text-based survey responses, and test ensemble methods like PyTorch neural network or Random Forest. By integrating these additional improvements, I aim to build a more robust and interpretable model that not only performs well but also offers meaningful insights for universities, counselors, and policymakers looking to support student wellness. Also, if I continue to include more complex data inside, using a neural network might be a better fit.