Student Depression Prediction

1 DATA ANALYSIS AND PREPROCESSING

1.1 Dataset Overview:

The dataset comprises 27,901 records and 18 features, including a mix of 8 categorical and 10 numerical variables, with the target variable Depression indicating whether individuals are depressed (1) or not (0). The target distribution reveals 58.5% depressed (16,335 samples) and 41.5% not depressed (11,563 samples), indicating slight class imbalance.

1.2 Exploratory Data Analysis

1.2.1 Data Statistics

Summary statistics for numerical columns were obtained using df. describe (). for numerical columns show the Age ranges from 18 to 59 years, and the mean CGPA is 7.66 (±1.47). The Depression column is binary, with values 0 (not depressed) and 1 (depressed).

```
df.describe()
```

	id	Age	Academic Pressure	Work Pressure	CGPA	Study Satisfaction	Job Satisfaction	Work/Study Hours	Financial Stress	Depression
count	27901.000000	27901.000000	27901.000000	27901.000000	27901.000000	27901.000000	27901.000000	27901.000000	27898.000000	27901.000000
mean	70442.149421	25.822300	3.141214	0.000430	7.656104	2.943837	0.000681	7.156984	3.139867	0.585499
std	40641.175216	4.905687	1.381465	0.043992	1.470707	1.361148	0.044394	3.707642	1.437347	0.492645
min	2.000000	18.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	1.000000	0.000000
25%	35039.000000	21.000000	2.000000	0.000000	6.290000	2.000000	0.000000	4.000000	2.000000	0.000000
50%	70684.000000	25.000000	3.000000	0.000000	7.770000	3.000000	0.000000	8.000000	3.000000	1.000000
75%	105818.000000	30.000000	4.000000	0.000000	8.920000	4.000000	0.000000	10.000000	4.000000	1.000000
max	140699.000000	59.000000	5.000000	5.000000	10.000000	5.000000	4.000000	12.000000	5.000000	1.000000

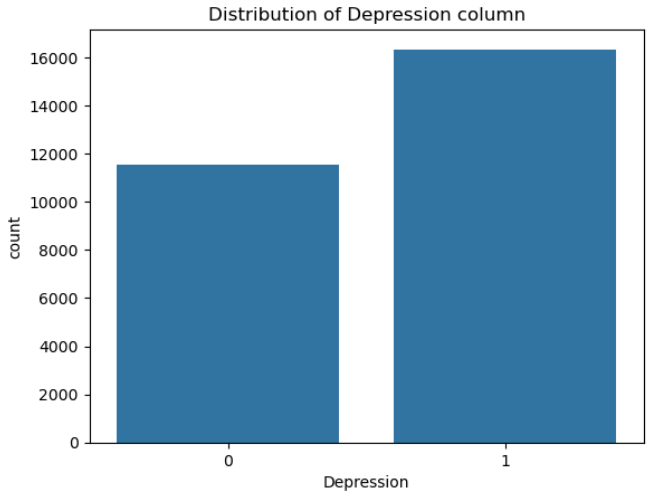
1.2.2 Checking Missing Values

Checked missing values using df.isnull(). sum (). The column Financial Stress has 3 missing values.

```
df.isnull().sum()
id                                       0
Gender                                   0
Age                                      0
City                                     0
Profession                               0
Academic Pressure                        0
Work Pressure                            0
CGPA                                     0
Study Satisfaction                       0
Job Satisfaction                         0
Sleep Duration                           0
Dietary Habits                           0
Degree                                   0
Have you ever had suicidal thoughts ?    0
Work/Study Hours                         0
Financial Stress                         3
Family History of Mental Illness         0
Depression                               0
dtype: int64
```

1.2.3 Target Variable Analysis

The target column Depression has 16,335 records labeled as 1 (Depressed) and 11,563 as 0 (Not Depressed). A bar plot visualizes this distribution.



1.2.4 Checking Information about the data

The df.info () method shows 27,901 entries and 18 columns, with 8 numerical and 8 categorical columns. There are 3 missing values in the Financial Stress column.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 27901 entries, 0 to 27900
Data columns (total 18 columns):
 #   Column                                 Non-Null Count  Dtype
---  ------                                 --------------  -----
 0   id                                     27901 non-null  int64
 1   Gender                                 27901 non-null  object
 2   Age                                    27901 non-null  float64
 3   City                                   27901 non-null  object
 4   Profession                             27901 non-null  object
 5   Academic Pressure                      27901 non-null  float64
 6   Work Pressure                          27901 non-null  float64
 7   CGPA                                   27901 non-null  float64
 8   Study Satisfaction                     27901 non-null  float64
 9   Job Satisfaction                       27901 non-null  float64
 10  Sleep Duration                         27901 non-null  object
 11  Dietary Habits                         27901 non-null  object
 12  Degree                                 27901 non-null  object
 13  Have you ever had suicidal thoughts ?  27901 non-null  object
 14  Work/Study Hours                       27901 non-null  float64
 15  Financial Stress                       27898 non-null  float64
 16  Family History of Mental Illness       27901 non-null  object
 17  Depression                             27901 non-null  int64
```

1.2.5 Data Pre-processing and Validation

Dropping Null Values:

Dropping Null Values: Rows containing missing or null values in the dataset are removed using df.dropna(axis=0). This operation ensures that only rows with complete data across all columns are retained, which helps maintain the integrity and consistency of the dataset. By eliminating incomplete rows, the risk of introducing inaccuracies or biases during analysis or model training is minimized.

```
df = df.dropna(axis=0)

df
```

Encoding Categorical Features:

Categorical features like 'Gender' and 'City' are encoded using Label Encoding or One-Hot Encoding (pd.get_dummies ()). Numerical columns such as 'Age' are standardized with StandardScaler for consistency in scale. The dataset is split using train_test_split () to evaluate model performance. These preprocessing steps prepare the data for machine learning.

```
df_dummies = pd.get_dummies(df, columns=['Gender', 'City', "Profession",
                                         "Sleep Duration", 'Dietary Habits',
                                         'Degree', 'Have you ever had suicidal thoughts ?', 'Family History of Mental

df_dummies
```

	id	Age	Academic Pressure	Work Pressure	CGPA	Study Satisfaction	Job Satisfaction	Work/Study Hours	Financial Stress	Depression	...	Degree_MD	Degree_ME	Degree_MHM	Degr
0	2	33.0	5.0	0.0	8.97	2.0	0.0	3.0	1.0	1	...	False	False	False	
1	8	24.0	2.0	0.0	5.90	5.0	0.0	3.0	2.0	0	...	False	False	False	
2	26	31.0	3.0	0.0	7.03	5.0	0.0	9.0	1.0	0	...	False	False	False	
3	30	28.0	3.0	0.0	5.59	2.0	0.0	4.0	5.0	1	...	False	False	False	
4	32	25.0	4.0	0.0	8.13	3.0	0.0	1.0	1.0	0	...	False	False	False	

Dropping Unnecessary Columns: The column id is dropped using df_dummies. drop('id', axis=1) as it does not provide predictive value and is not required for model training.

```
df_dropped = df_dummies.drop('id', axis=1)

df_dropped
```

```
df_dropped.Depression.value_counts()
```

```
Depression
1    16335
0    11563
Name: count, dtype: int64
```

Balancing the DataFrame: The dataset is balanced by grouping by Depression and sampling 11563 rows from each category, then resetting the index. The balance is confirmed by checking the value counts, which show 11563 rows per category.

```
balanced_df.Depression.value_counts()
```

```
Depression
0    11563
1    11563
Name: count, dtype: int64
```

Correlation Matrix: The corr () method is used to calculate the correlation matrix for the numeric columns in the balanced_df. This matrix shows the relationship between the numerical features of the dataset.

```
# Calculate the correlation matrix
corr_matrix = balanced_df.corr()

# Print the correlation matrix
corr_matrix
```

2 Modeling:

2.1.1 Feature extraction

Dataset Split:

X represents the features, which includes all columns except the target variable "Depression," while y is the target variable, the "Depression" column. The dataset is then split into training (80%) and testing (20%) sets using train_test_split for model evaluation.

Standardizing Features for Supervised and Unsupervised Learning:

In supervised learning, the training data (X_train) is standardized using StandardScaler with fit_transform () to learn the scaling parameters, while the test data (X_test) is scaled using transform () based on the training set's statistics. For unsupervised learning, the entire dataset (X) is standardized using transform () to ensure uniform feature scaling without fitting new parameters, maintaining consistency across both phases.

```
# Standerdizing feature for Supervised Learning phase
scaler = StandardScaler()

X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Standerdizing feature for UN-Supervised Learning phase
X_scaled_USP = scaler.transform(X)
```

Training Random Forest Classifier for Feature Selection

It trains a RandomForestClassifier on scaled data to determine feature importance for predicting the target variable. The model computes and prints importance scores, where higher values indicate more impactful features, and near-zero scores suggest minimal relevance.

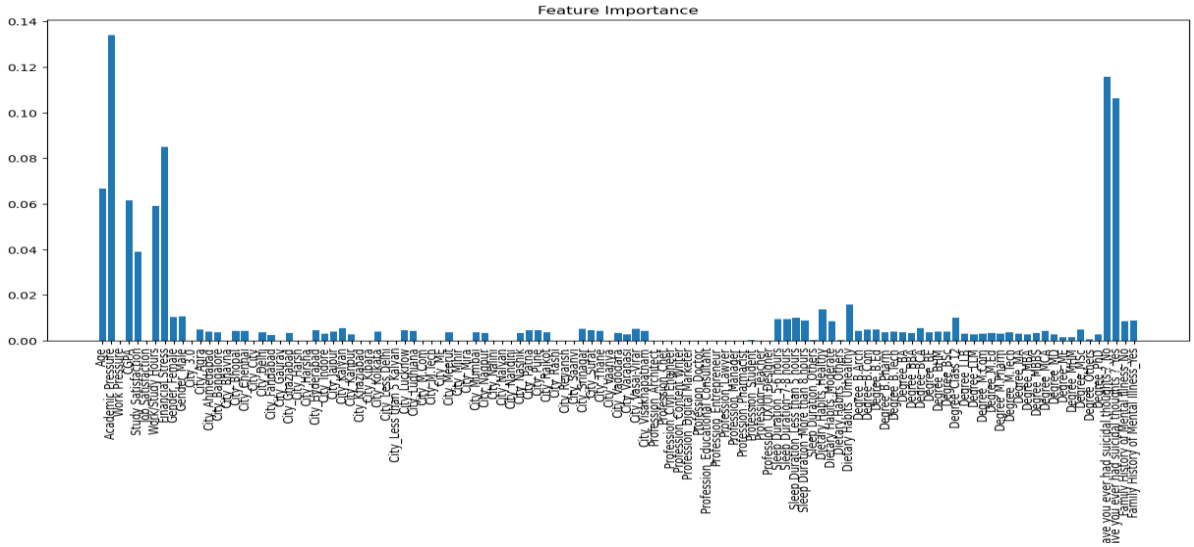
Creating DataFrame for Feature Importance

A Data Frame is than created to display feature importances alongside their column names, then sorts the features by importance in descending order. It prints the sorted Data Frame to highlight the most significant features.

```
Feature Importance:
                                    Feature  Importance
1                          Academic Pressure    0.134104
113   Have you ever had suicidal thoughts ?_No    0.115808
114  Have you ever had suicidal thoughts ?_Yes    0.106217
7                          Financial Stress    0.084823
0                                       Age    0.066667
..                                      ...         ...
32                        City_Less Delhi     0.000000
36                             City_M.Com     0.000000
37                            City_M.Tech     0.000000
57                            City_Vaanya     0.000000
70                      Profession_Lawyer     0.000000

[117 rows x 2 columns]
```

Finally, a bar plot is generated to visually represent the feature importances, helping to identify which features most influence the model's predictions.



Dataset Splitting for selected features and Feature Scaling

It splits the dataset into training and test sets (80%/20%) using selected features and corresponding labels. It then standardizes the selected features with StandardScaler, ensuring they have a mean of 0 and standard deviation of 1. The scaler is applied to both the training and test data for consistent feature scaling across the model.

```
# Split data into training and test sets
X_train_sel_features, X_test_sel_features, y_train_sel_features, y_test_sel_features = train_test_split(
    X_sel_features, y, test_size=0.2, random_state=42
)

X_train_sel_features
```

	Age	Academic Pressure	CGPA	Study Satisfaction	Work/Study Hours	Financial Stress	Gender_Female	Gender_Male	City_Agra	City_Ahmedabad	...	Degree_BSc	Degree_Class 12
20151	21.0	1.0	9.36	5.0	8.0	1.0	False	True	True	False	...	False	False
915	28.0	5.0	6.78	2.0	0.0	3.0	True	False	False	False	...	False	False
2728	18.0	2.0	9.98	5.0	9.0	1.0	False	True	False	False	...	False	True
22929	25.0	5.0	5.87	5.0	8.0	2.0	True	False	False	False	...	False	False
5564	27.0	4.0	9.44	5.0	5.0	4.0	True	False	False	False	...	False	False
...	...	...	...	...	...	...	...	...	...	...	...	...	...
11964	28.0	5.0	8.69	2.0	10.0	5.0	True	False	False	False	...	True	False
21575	20.0	3.0	8.69	4.0	11.0	5.0	False	True	False	False	...	False	True
5390	23.0	5.0	8.63	5.0	8.0	2.0	True	False	False	False	...	False	False
860	25.0	3.0	8.04	2.0	8.0	2.0	False	True	False	False	...	False	False
15795	28.0	3.0	6.89	4.0	12.0	3.0	False	True	False	False	...	False	False

18500 rows × 59 columns

3 Evaluation

3.1 XGBoost Model Training and Performance Evaluation

We have trained XGBoost classifier using the scaled selected features and evaluates its performance. The model is fitted on the training data, and predictions are made on the test set. The predicted labels are compared to the actual labels, and the results are summarized through accuracy, a confusion matrix, and a classification report.

```
# Initialize the XGBoost classifier
xgb_model = XGBClassifier(use_label_encoder=False, eval_metric='logloss')

# Train the model
xgb_model.fit(X_train_scaled_sel_features, y_train_sel_features)

# Make predictions
y_pred = xgb_model.predict(X_test_scaled_sel_features)
```

```
print(f'Actual : {y_test_sel_features[:10]}')
print(f'Predicted: {y_pred[:10]}')
```

```
Actual : 16842    1
3615     0
18229    1
1398     0
4540     0
1512     0
12548    1
12731    1
11622    1
11320    0
Name: Depression, dtype: int64
Predicted: [1 0 1 0 0 0 1 1 1 0]
```

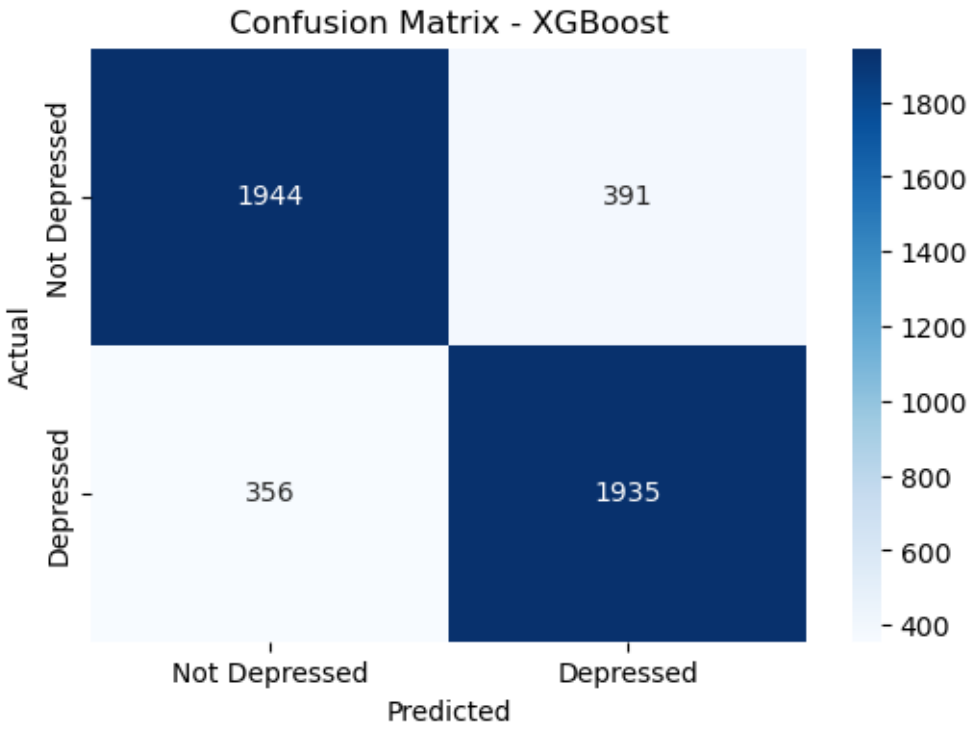
3.2 Confusion matrix and classification report

The confusion matrix visually represents the true positives, true negatives, false positives, and false negatives, while the classification report provides precision, recall, F1-score, and support for each class. Additionally, a heatmap of the confusion matrix helps interpret the model's performance in predicting "Depressed" and "Not Depressed" cases. The overall accuracy of the model is 83.85%, indicating good predictive capability.

```
XGBoost Model
Accuracy: 0.8385214007782101
Classification Report:
              precision    recall  f1-score   support

           0       0.85      0.83      0.84      2335
           1       0.83      0.84      0.84      2291

    accuracy                           0.84      4626
   macro avg       0.84      0.84      0.84      4626
weighted avg       0.84      0.84      0.84      4626
```



3.3 Random Forest Model Training and Performance Evaluation

We also trained second model a Random Forest Classifier model with 2000 trees using the scaled selected features. Predictions are made on the test set, and the model's performance is evaluated using accuracy, confusion matrix, and a classification report. The confusion matrix visualizes the actual vs. predicted labels, distinguishing between "Not Depressed" and "Depressed." The classification report provides precision, recall, and F1-score for each class, showing an overall accuracy of 84.57%.

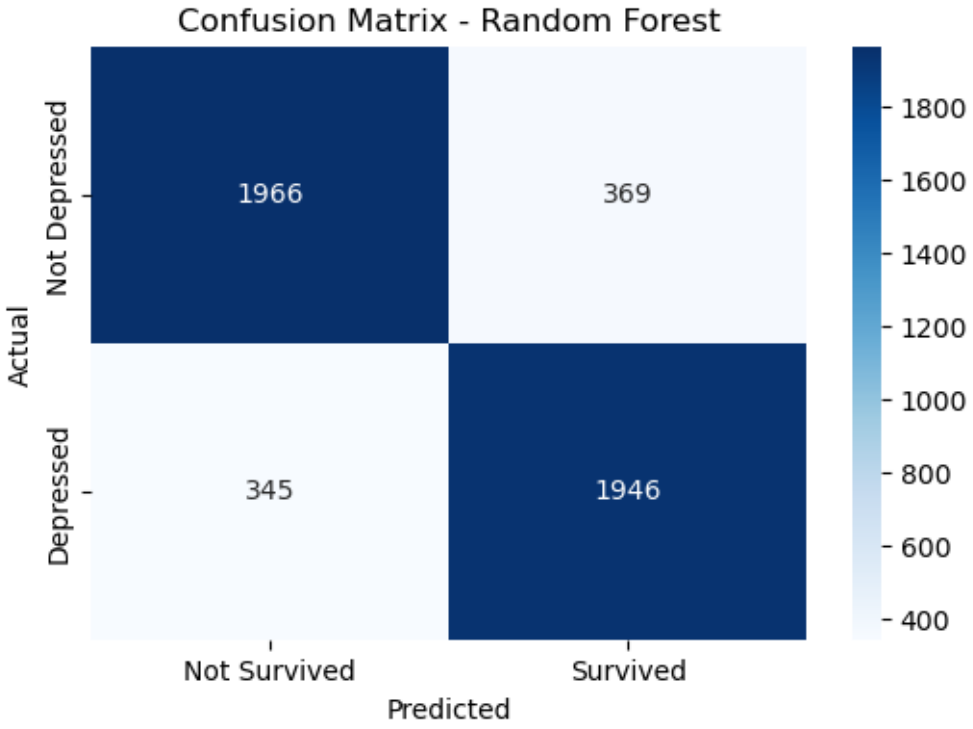
3.4 Classification Report

```
Random Forest Model
Accuracy: 0.8456549935149157
Classification Report:
              precision    recall  f1-score   support

           0       0.85      0.84      0.85      2335
           1       0.84      0.85      0.84      2291

    accuracy                           0.85      4626
   macro avg       0.85      0.85      0.85      4626
weighted avg       0.85      0.85      0.85      4626
```

3.5 Confusion Matrix



3.6 Model Validation

validates the performance of the XGBoost and Random Forest models using cross-validation. The cross_val_score function splits the dataset into 5 folds, trains the model on 4 folds, and evaluates it on the remaining fold, repeating this process for all folds. It calculates the cross-validation scores for both models and outputs the scores for each fold along with the mean cross-validation score. This ensures the robustness of the models by assessing their performance on different data splits.

```
Cross-validation scores for XGBoost model is : [0.83290099 0.83502703 0.82897297 0.82745946 0.83005405]
Mean CV score: 0.8308829015786214

Cross-validation scores for Random Forest model is : [0.83463035 0.83567568 0.83545946 0.83610811 0.83372973]
Mean CV score: 0.8351206646335051
```

4 Interpretation

Model Comparison:

- The Random Forest model slightly outperformed the XGBoost model in terms of accuracy and cross-validation scores, indicating it may be better at capturing underlying patterns in the dataset.
- Both models demonstrate balanced precision, recall, and F1-scores, making them effective for predicting both "Depressed" and "Not Depressed" classes.

Model Stability:

- Cross-validation scores for both models are consistent with their evaluation accuracies, confirming their robustness and stability across different data splits. This suggests the models are not overfitting and can generalize well to unseen data.

Class Balance:

- The balanced performance across both classes highlights that the models are not biased towards one class, which is critical for fair and accurate predictions in sensitive tasks like depression detection.

Practical Application:

- The high accuracy and reliability of both models make them suitable for real-world deployment in identifying depression. The slight edge of the Random Forest model suggests it could be the preferred choice if computational efficiency and interpretability are also considered.

Future Improvements:

- Incorporating additional features or fine-tuning hyperparameters further could help improve both models' performance. Exploring ensemble approaches combining both XGBoost and Random Forest could leverage their strengths for even better outcomes.

Conclusion:

Both XGBoost and Random Forest models exhibit high accuracy and consistent performance in predicting depression, with Random Forest slightly outperforming XGBoost in terms of accuracy and cross-validation stability. The balanced precision, recall, and F1-scores across both classes highlight their effectiveness in handling the dataset. These results indicate that both models are reliable tools for identifying depression, offering valuable potential applications in early detection, mental health monitoring, and supporting timely and informed interventions