



Machine Learning Project

Name	Sec	BN	Code
Sara Bisheer Fikry	1	19	9210453
Menna Mohammed AbdelBaset	2	26	9211242
Rawan Mostafa Mahmod	1	18	9210423
Fatma Ebrahim Sobhy	2	1	9210799

Motivation	4
Problem Definition	4
Evaluation Metrics	4
Dataset Link	4
EDA(Exploratory Data Analysis)	5
➤ Data Information	5
➤ Data Description	5
➤ Null Values	6
➤ Distribution of classes in dataset	7
➤ Data Visualization	8
➤ Outlier Analysis	10
Feature Engineering	11
Baseline model	14
➤ ZeroR model	14
➤ Stratified model	14
SVM	15
➤ Hyperparameter Tuning	16
➤ Partial Dependence Plots	18
➤ Train-Validation Curve	19
➤ Bias-variance Analysis	19
➤ Results	20
Random Forest	20
➤ Feature Importance Plot	20
➤ Hyperparameter Tuning	22
➤ Partial Dependence Plots	23
➤ Bias Variance Analysis	23
➤ Results	23
Logistic Regression	24
➤ Feature Importance Plot	24
➤ Learning Curves Plot	25
➤ Partial Dependence Plot	26
➤ Hyperparameter Tuning	27

➤ Train-Validation Curve	28
➤ Bias-variance Analysis	28
➤ Results	29
Adaboost	29
➤ Feature Importance Plot	29
➤ Learning Curves Plot	30
➤ Partial Dependence Plot	31
➤ Hyperparameter Tuning	32
➤ Train-Validation Curve	33
➤ Results	33
Conclusion	34
Work Distribution	35

Motivation

Depression affects millions worldwide, yet early detection remains a challenge. We want to help identify patterns in survey responses.

Problem Definition

The goal is to develop a binary classification model that predicts the target class either depressed or not based on survey responses.

Evaluation Metrics

- Accuracy
- Recall
- Precision
- F1-Score

Dataset Link

- Competition Link:
<https://www.kaggle.com/competitions/playground-series-s4e11/overview>
- Dataset Link:
<https://www.kaggle.com/competitions/playground-series-s4e11/data?select=train.csv>

EDA(Exploratory Data Analysis)

➤ Data Information

we have displayed the features with its data types and non-null values

```
RangeIndex: 140700 entries, 0 to 140699
Data columns (total 20 columns):
#   Column                                          Non-Null Count  Dtype
---  -
0   id                                              140700 non-null  int64
1   Name                                           140700 non-null  object
2   Gender                                         140700 non-null  object
3   Age                                             140700 non-null  float64
4   City                                           140700 non-null  object
5   Working Professional or Student              140700 non-null  object
6   Profession                                     104070 non-null  object
7   Academic Pressure                             27897 non-null   float64
8   Work Pressure                                112782 non-null  float64
9   CGPA                                           27898 non-null   float64
10  Study Satisfaction                             27897 non-null   float64
11  Job Satisfaction                             112790 non-null  float64
12  Sleep Duration                                140700 non-null  object
13  Dietary Habits                               140696 non-null  object
14  Degree                                         140698 non-null  object
15  Have you ever had suicidal thoughts ?         140700 non-null  object
16  Work/Study Hours                             140700 non-null  float64
17  Financial Stress                             140696 non-null  float64
18  Family History of Mental Illness              140700 non-null  object
19  Depression                                    140700 non-null  int64
dtypes: float64(8), int64(2), object(10)
memory usage: 21.5+ MB
```

➤ Data Description

in this part we have displayed the important values such as maximum, minimum, average, standard deviation, 1st and 3rd quartiles for each features to explore the nature of data

<----- Data Description ----->				
	id	Age	Academic Pressure	Work Pressure \
count	140700.000000	140700.000000	27897.000000	112782.000000
mean	70349.500000	40.388621	3.142273	2.998998
std	40616.735775	12.384099	1.380457	1.405771
min	0.000000	18.000000	1.000000	1.000000
25%	35174.750000	29.000000	2.000000	2.000000
50%	70349.500000	42.000000	3.000000	3.000000
75%	105524.250000	51.000000	4.000000	4.000000
max	140699.000000	60.000000	5.000000	5.000000

	CGPA	Study Satisfaction	Job Satisfaction	Work/Study Hours \
count	27898.000000	27897.000000	112790.000000	140700.000000
mean	7.658636	2.944940	2.974404	6.252679
std	1.464466	1.360197	1.416078	3.853615
min	5.030000	1.000000	1.000000	0.000000
25%	6.290000	2.000000	2.000000	3.000000
50%	7.770000	3.000000	3.000000	6.000000
75%	8.920000	4.000000	4.000000	10.000000
max	10.000000	5.000000	5.000000	12.000000

	Financial Stress	Depression
count	140696.000000	140700.000000
mean	2.988983	0.181713
std	1.413633	0.385609
min	1.000000	0.000000
25%	2.000000	0.000000
50%	3.000000	0.000000
75%	4.000000	0.000000
max	5.000000	1.000000

➤ Null Values

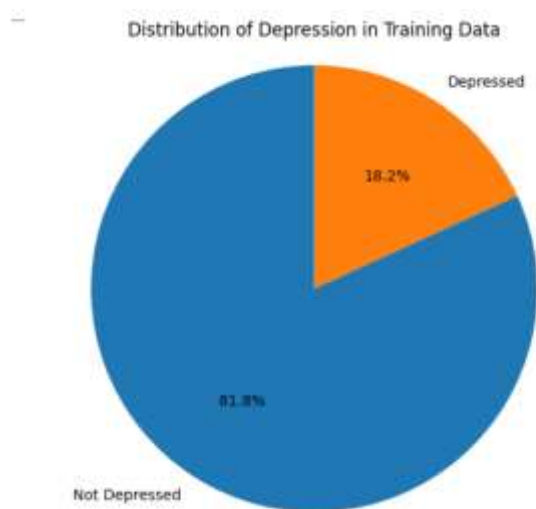
We have displayed the null values number for each feature

<----- Null Values ----->	
id	0
Name	0
Gender	0
Age	0
City	0
Working Professional or Student	0
Profession	36630
Academic Pressure	112803
Work Pressure	27918
CGPA	112802
Study Satisfaction	112803
Job Satisfaction	27910
Sleep Duration	0
Dietary Habits	4
Degree	2
Have you ever had suicidal thoughts ?	0
Work/Study Hours	0
Financial Stress	4
Family History of Mental Illness	0
Depression	0
dtype: int64	

We have handled the null values as following:

- Numerical Features: We put the median of values in the features, we have tried to put the mean instead of median but it makes no difference
- Categorical Features: We have added “Missing” labels for the null value for categorical features

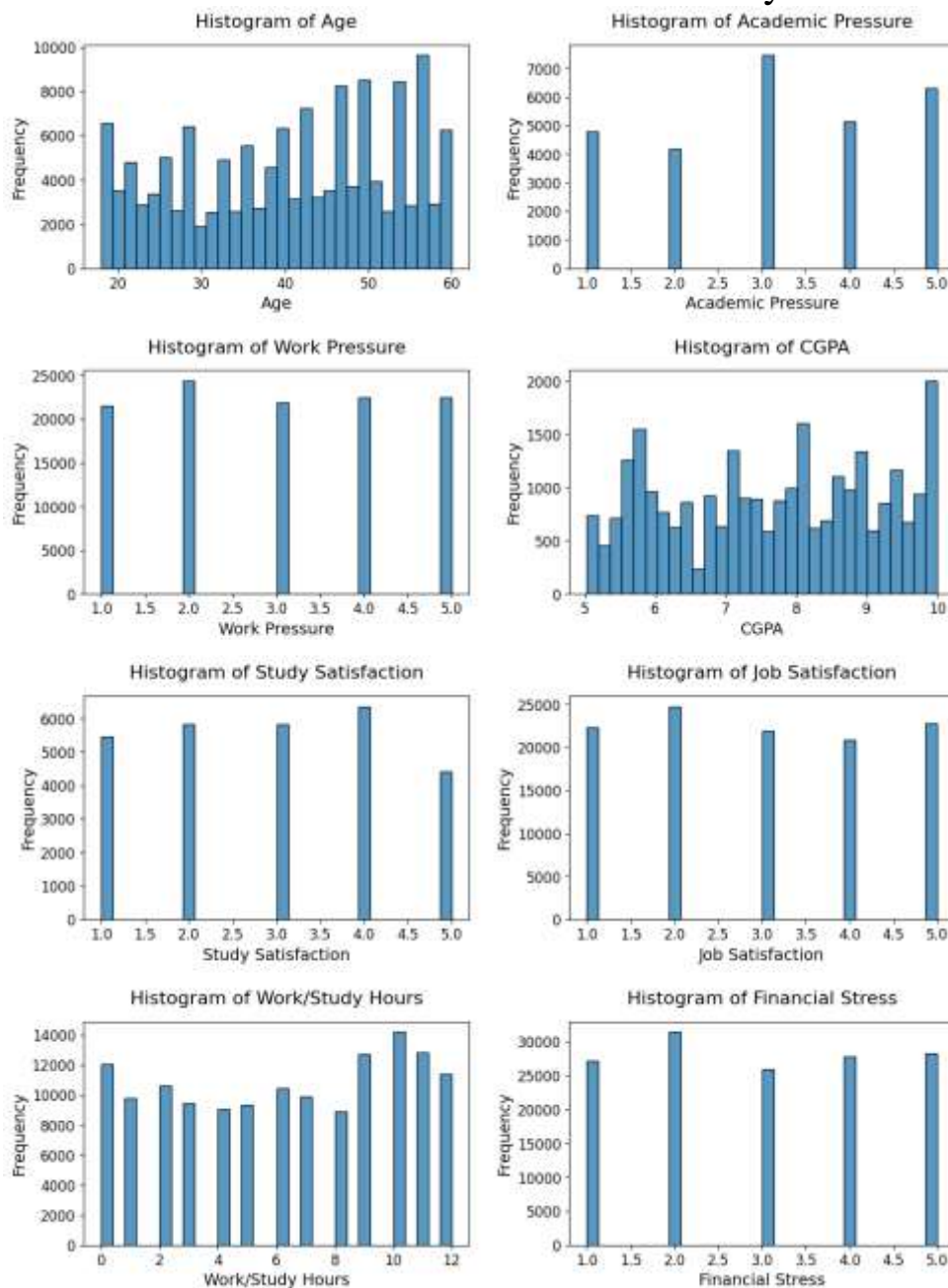
➤ Distribution of classes in dataset



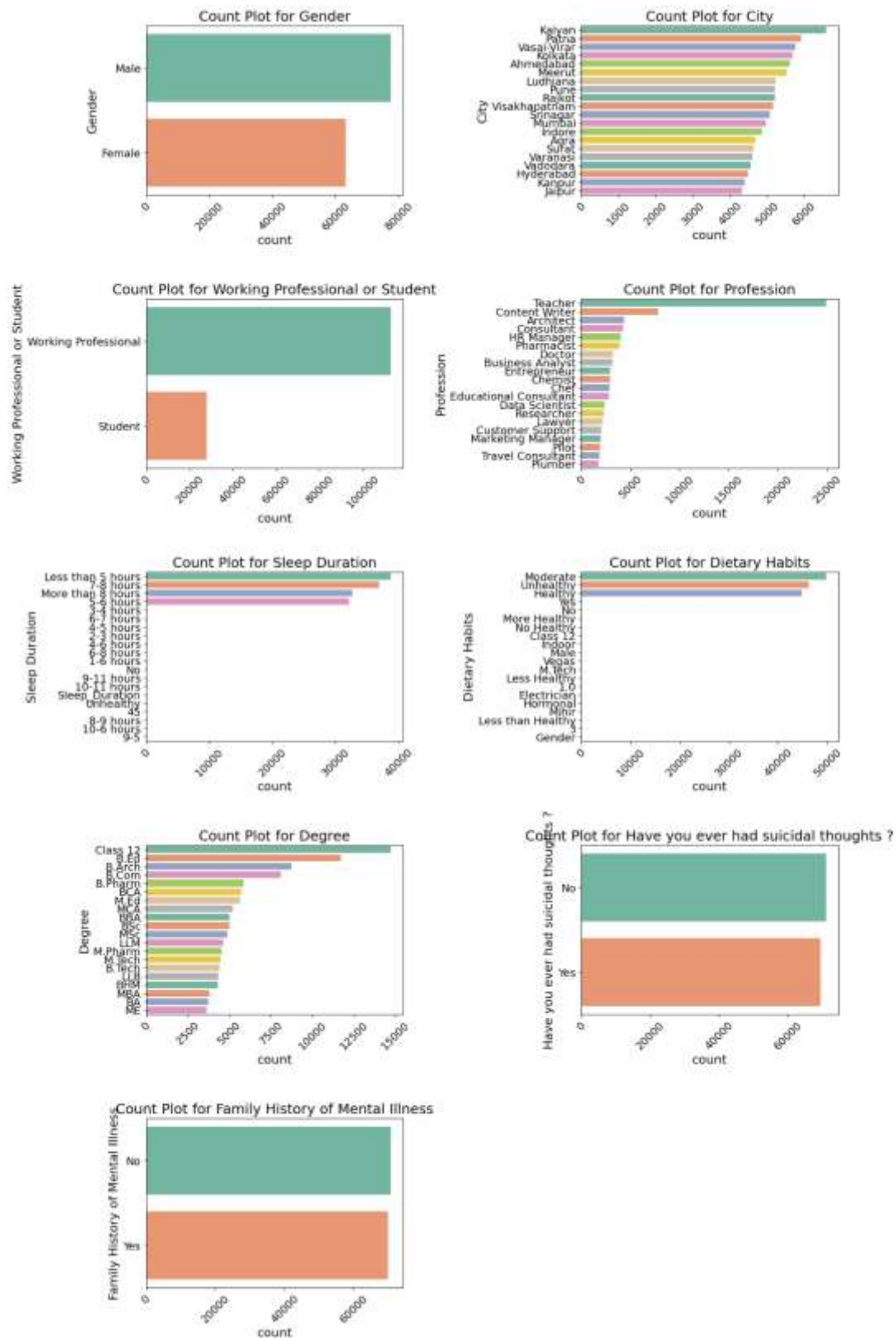
and because the two classes are imbalanced, we take metrics like recall, precision, and F1-Score into consideration.

➤ Data Visualization

First, We have made histograms for numerical features to see the distribution of values of each feature visually

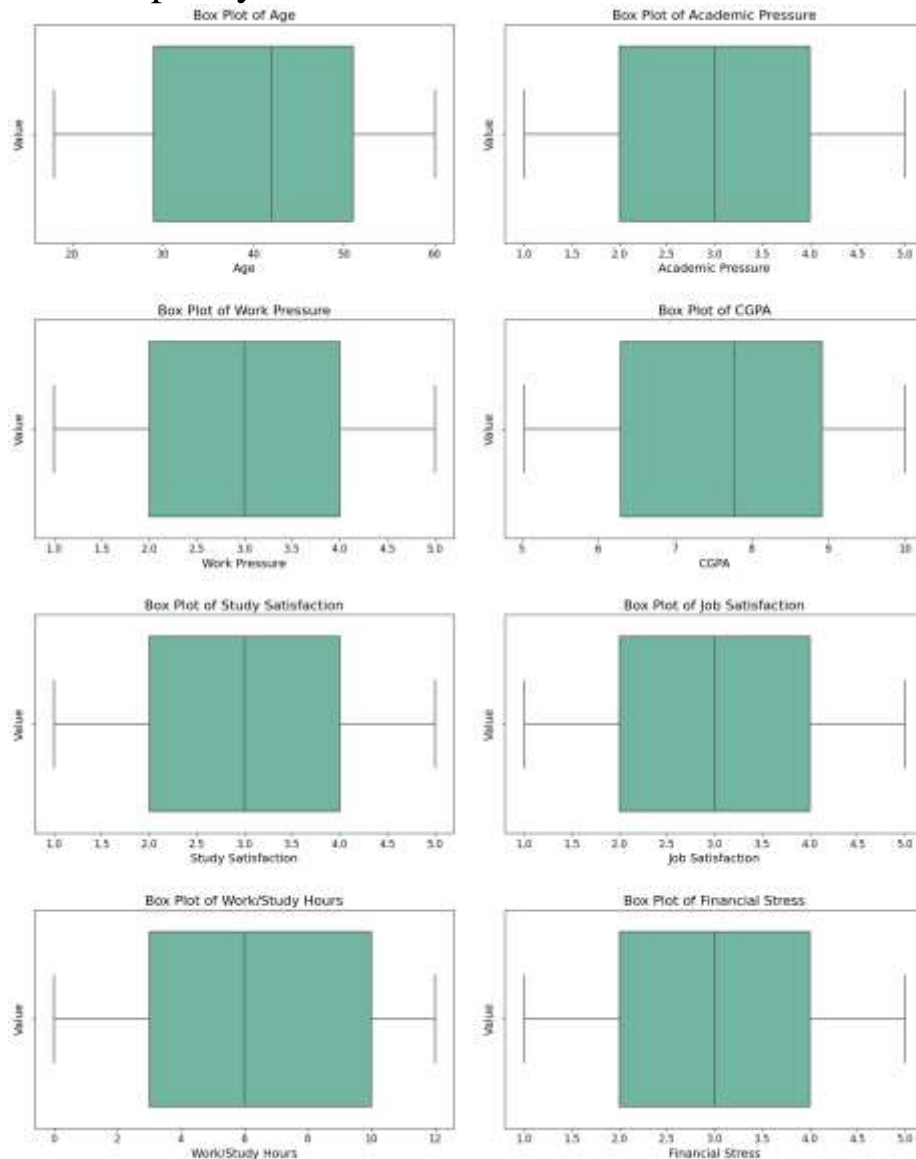


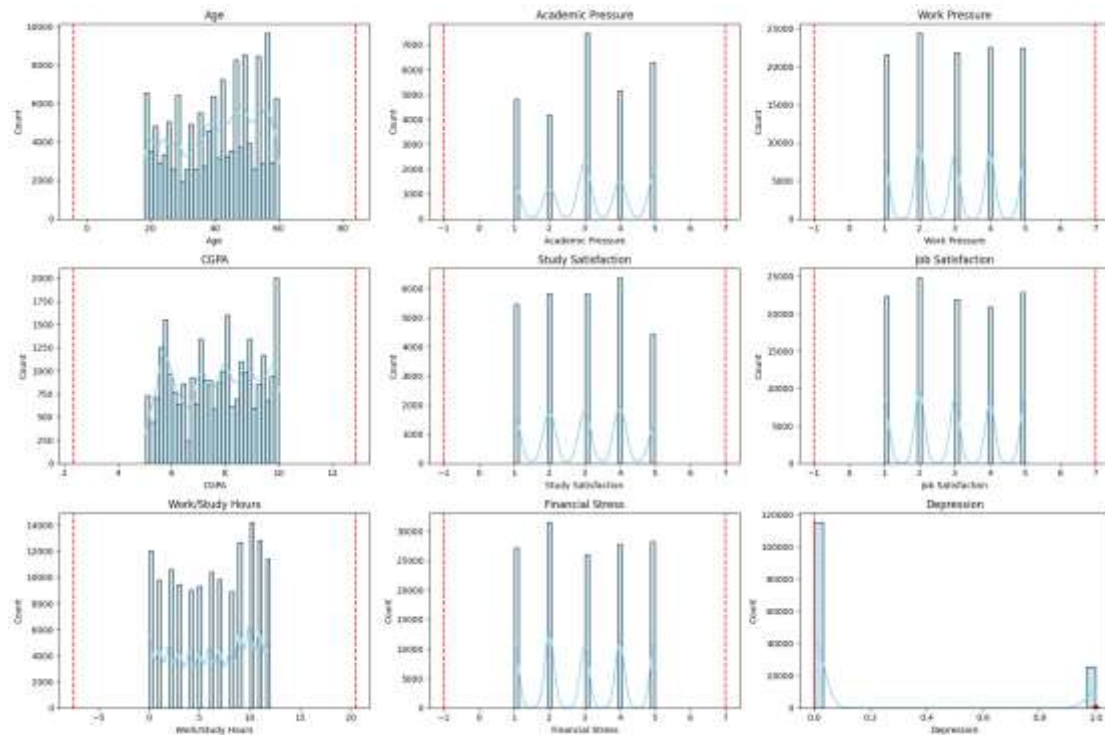
Then we made count plot for categorical features to see the distribution of each features categories



➤ Outlier Analysis

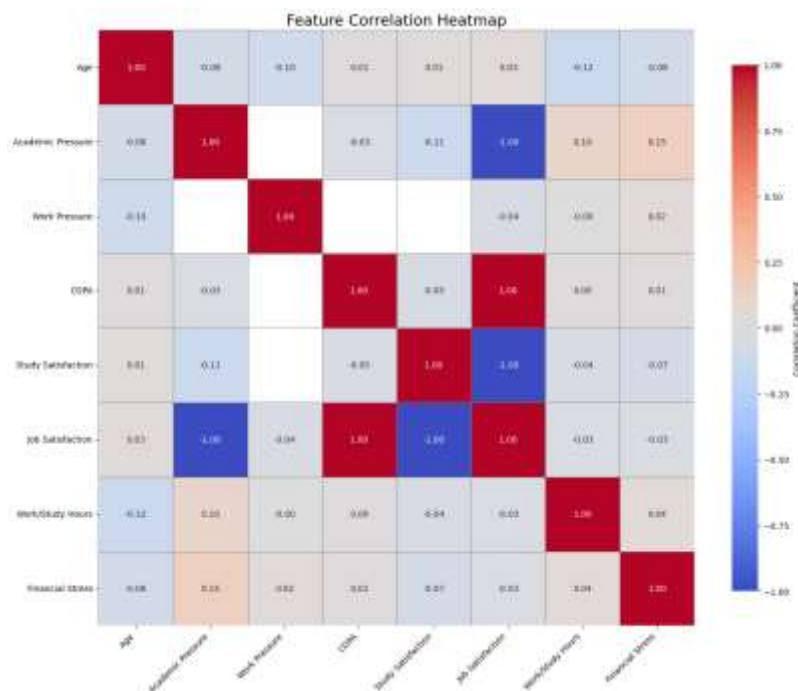
We have made box plots for the features to make sure that there is no plenty of outliers





Feature Engineering

We have calculated a correlation matrix for the numerical features to see the relation between them and if there is any redundant information any feature introduce and here is what we got:



The graph shows a high correlation between the job satisfaction and CGPA

We then removed one of them which is CGPA but it has no effect between removing it and preserving it and here is the result of AdaBoost before remove CGPA and after:

Before:

AdaBoost Model Performance:

Classification Report:

	precision	recall	f1-score	support
0	0.96	0.97	0.96	22986
1	0.84	0.82	0.83	5154
accuracy			0.94	28140
macro avg	0.90	0.89	0.90	28140
weighted avg	0.94	0.94	0.94	28140

After:

AdaBoost Model Performance:

Classification Report:

	precision	recall	f1-score	support
0	0.96	0.97	0.96	22986
1	0.84	0.82	0.83	5154
accuracy			0.94	28140
macro avg	0.90	0.89	0.90	28140
weighted avg	0.94	0.94	0.94	28140

it has no difference literally in all metrics so we decided to preserve CGPA for the rest of the models

➤ Adding alternative Features

- We have tried to add alternative features for example the age features we made Age_groups instead of Age as a pure number to be more descriptive
- We have made a new feature which is Total_Stress(is the summation of three features that are about stress)

```
train_data['Total_Stress'] =  
train_data['Academic Pressure'] +  
train_data['Work Pressure'] +  
train_data['Financial Stress']
```

and here is the result of adaboost after the new feature space

```
Test Performance:
      precision    recall  f1-score   support

     0       0.94      0.96      0.95     22986
     1       0.80      0.75      0.77      5154

 accuracy      0.92     28140
 macro avg      0.87     28140
 weighted avg    0.92     28140

Test Accuracy: 0.9198
```

It is worst than the results that we got before the new feature space so kept the original feature space

- We tried to replace the columns of job and study satisfaction with one columns as when one exists the other is null and replaced it with this equation

```
train_data['Total_Sat']=train_data['Study Satisfaction']+train_data['Job Satisfaction']
```

but what we got that the accuracy didn't change and here is the result of adaboost after replacing columns with this new column

```
Test Performance:
      precision    recall  f1-score   support

     0       0.96      0.96      0.96     22986
     1       0.83      0.81      0.82      5154

 accuracy      0.93     28140
 macro avg      0.89     28140
 weighted avg    0.93     28140

Test Accuracy: 0.9348
```

Baseline model

We have tried to use baseline model to see how our models improve, we have tried the following:

➤ ZeroR model

zeroR is a simple model that chooses the most frequent class as the prediction and here is the results that we got out of it:

Classification Report:					
	precision	recall	f1-score	support	
0	0.82	1.00	0.90	22986	
1	0.00	0.00	0.00	5154	
accuracy			0.82	28140	
macro avg	0.41	0.50	0.45	28140	
weighted avg	0.67	0.82	0.73	28140	

➤ Stratified model

It is a model that predicts according to the training class distribution but it randomly predicts, but following the same proportions as the classes appear in the training set. And here is the result of it:

Classification Report:					
	precision	recall	f1-score	support	
0	0.82	0.82	0.82	22986	
1	0.19	0.19	0.19	5154	
accuracy			0.70	28140	
macro avg	0.51	0.51	0.51	28140	
weighted avg	0.70	0.70	0.70	28140	

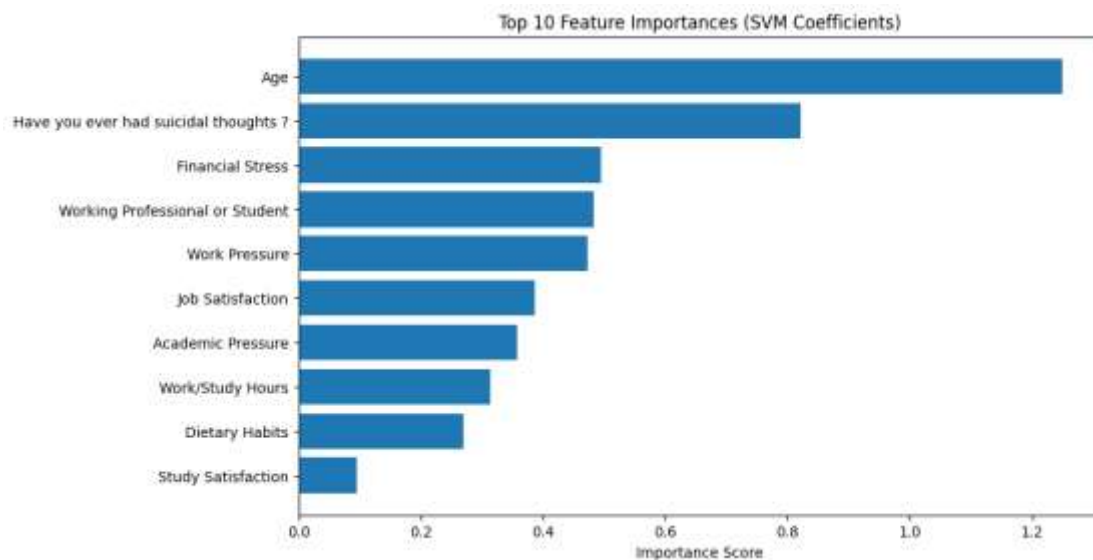
As we take F1 score, recall, precision, and accuracy as our metric we want to improve from the results that we got from baseline models above.

SVM

SVMs work by finding the hyperplane that best separates the data into different classes. The optimal decision boundary is the hyperplane that maximizes the margin between the two classes. In the case where the data is not linearly separable, SVMs use a kernel trick to map the data into a higher-dimensional space where the data can be linearly separated.

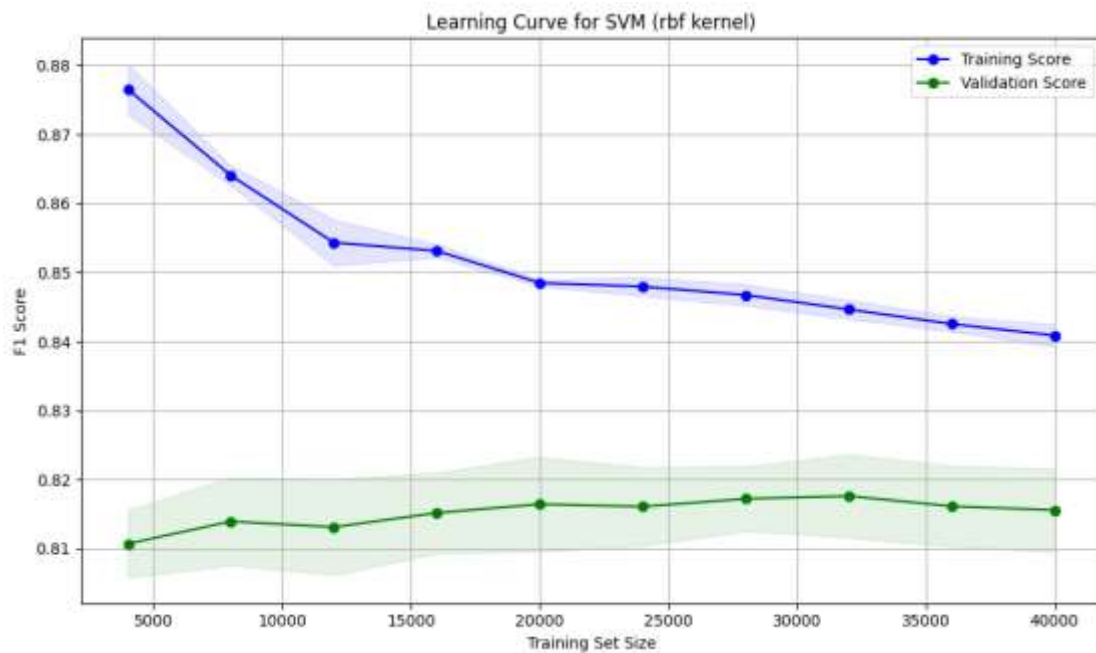
➤ Feature Importance Plot

A feature importance plot shows the importance of each feature in the model. It can be used to identify the most important features and to understand the impact of each feature on the model's predictions



From the plot we can conclude that the first two features “Age” and “Have you ever had suicidal thoughts ?” are the most important and the following three feature are almost equal.

➤ Learning Curves Plot



The validation F1 score is slightly lower, suggesting the model generalizes well but may have a small gap between training and validation performance. As the training set size increases, both scores stabilize without significant divergence, implying the model is not overfitting and benefits from more data.

➤ Hyperparameter Tuning

We have used grid search to find our best parameters and here is the parameters that we have finetuned

```
# Define the hyperparameter grid for SVM
param_grid = {
    'C': [0.1, 1, 10],          # Regularization parameter
    'gamma': [0.1, 1, 10],      # Kernel coefficient
    'kernel': ['linear', 'rbf'], # Kernel type
}
```

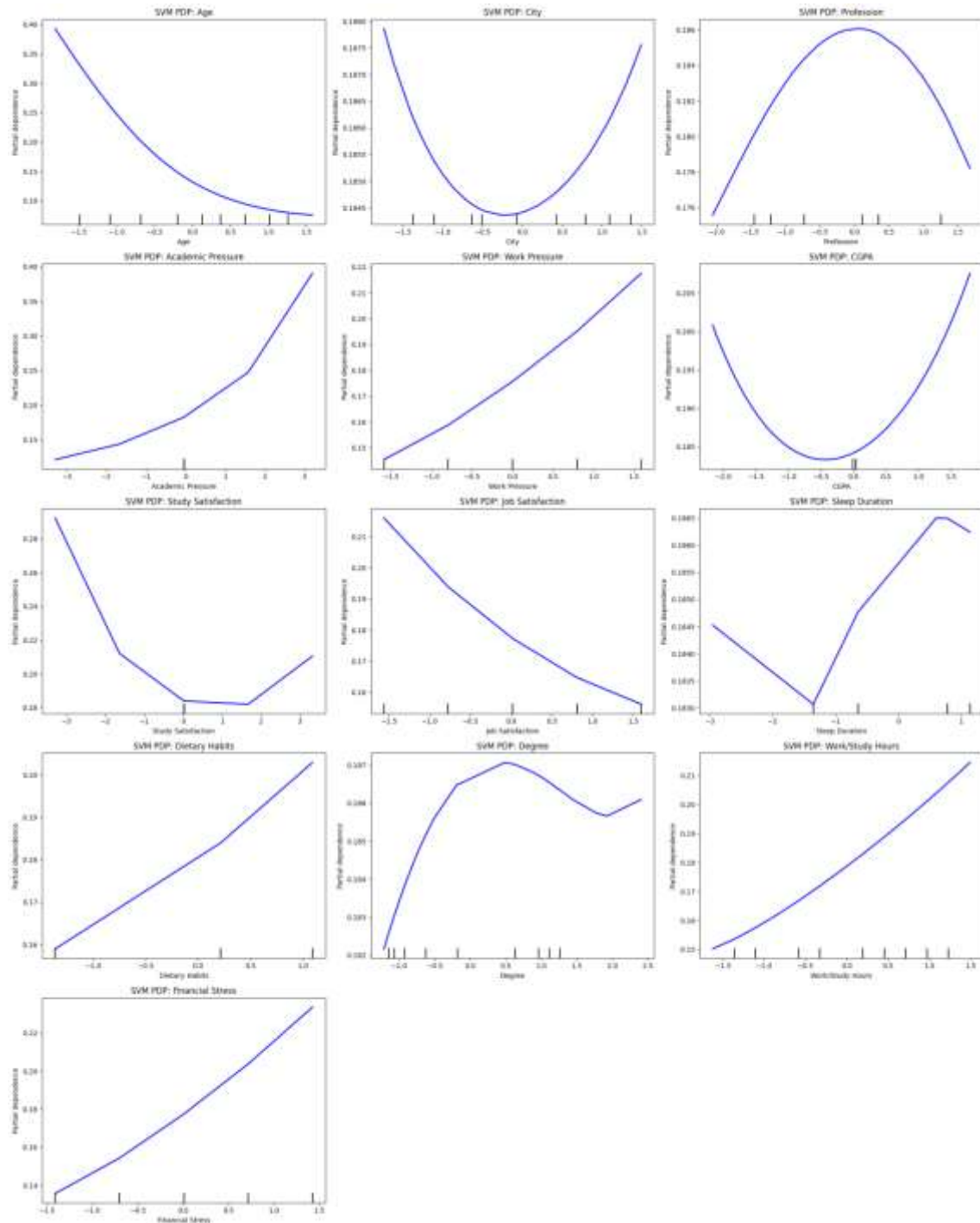
And we got that the best parameters are

Fitting 5 folds for each of 18 candidates, totalling 90 fits

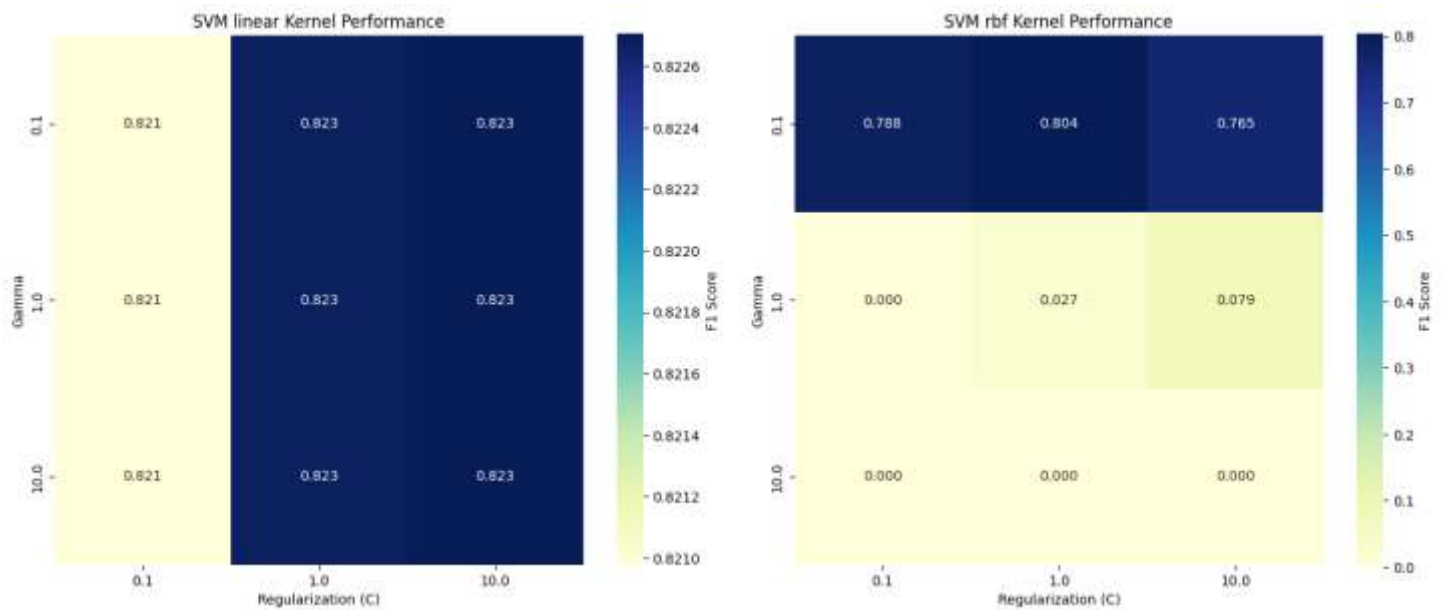
Best Parameters: {'C': 10, 'gamma': 0.1, 'kernel': 'linear'}
 Best F1 Score: 0.8227069005201078

➤ Partial Dependence Plots

we have plotted the partial dependence between each feature and the output class (Depression) to see the relation between them

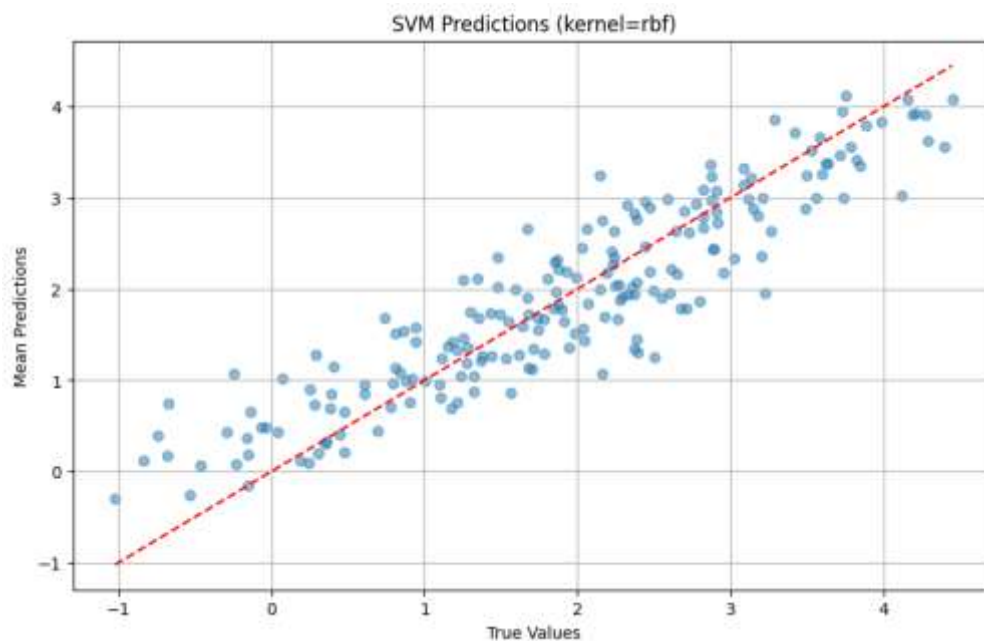


➤ Train-Validation Curve



➤ Bias-variance Analysis

- Accuracy: 0.9344
- Bias: 0.0656
- Variance: 0.0137



➤ Results

We found that all the kernels got almost the same results with no big difference, So we chose the linear kernel as our kernel because it is the simplest one and there is no need to add complexity when it didn't show any difference and to avoid overfitting

```
Results for linear kernel:
```

```
Accuracy: 0.9362
```

```
Training time: 722.01 seconds
```

```
Classification Report:
```

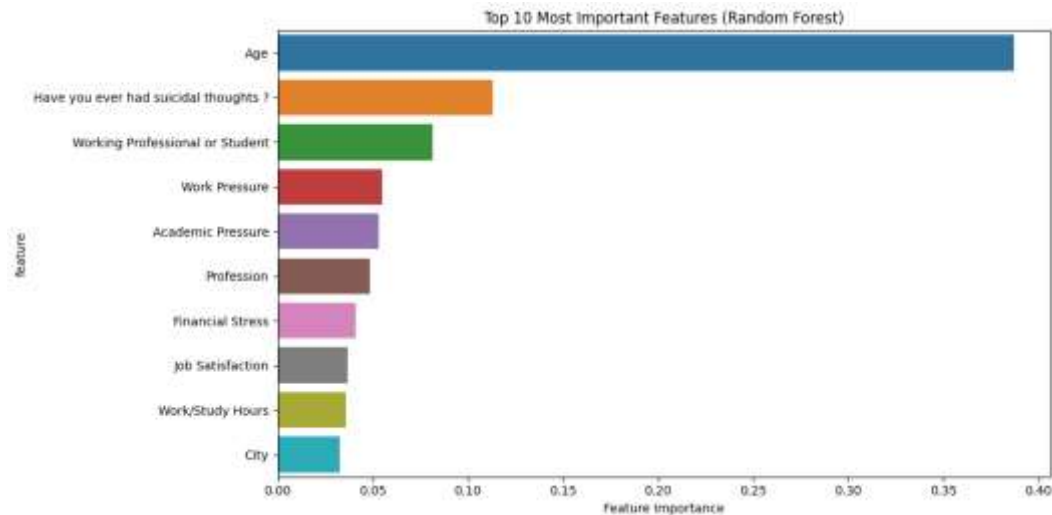
	precision	recall	f1-score	support
0	0.96	0.96	0.96	22986
1	0.84	0.81	0.82	5154
accuracy			0.94	28140
macro avg	0.90	0.89	0.89	28140
weighted avg	0.94	0.94	0.94	28140

Random Forest

Random Forest is a popular ensemble machine learning algorithm used for classification and regression. It works by combining the predictions of multiple decision trees to produce a more accurate and stable result.

➤ Feature Importance Plot

After Training Random Forest on our data we got the top important feature like following in the graph:



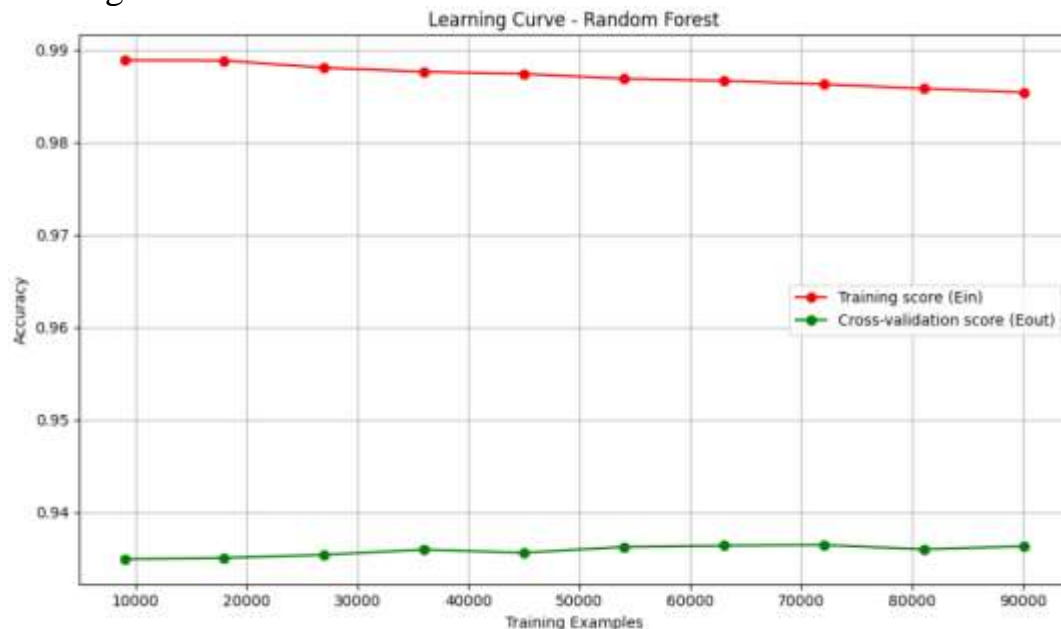
➤ Learning Curve Plot

What we got from the random forest when we calculated the accuracy on train dataset(which reflect E_{in}) and test dataset (which reflect E_{out}) we got the following:

Training Accuracy (E_{in}): 0.9847

Testing Accuracy (E_{out}): 0.9363

shows the training error (E_{in}) and testing error (E_{test}) as a function of the training set size:



➤ Hyperparameter Tuning

■ Grid Search:

We have used grid search to find our best parameters and here is the parameters that we have finetuned

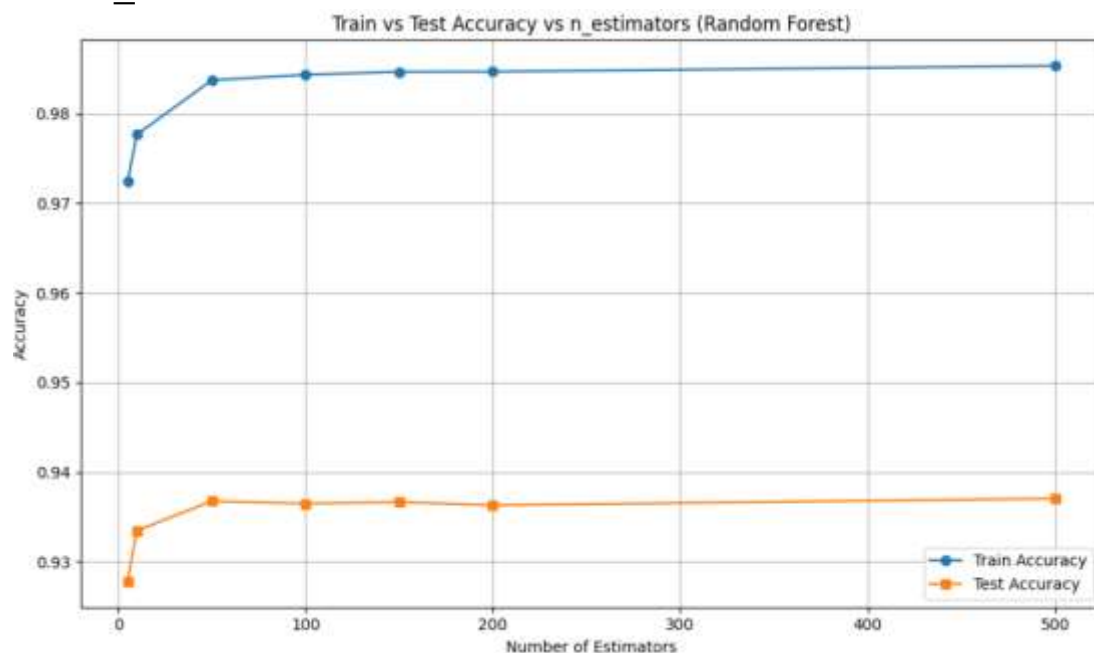
```
param_grid = {  
    'n_estimators': [100, 200, 300],  
    'max_depth': [10, 15, 20],  
    'min_samples_split': [2, 5, 10],  
    'min_samples_leaf': [1, 2, 4]  
}
```

And we got that the best parameters are

```
Best parameters: {'max_depth': 20, 'min_samples_leaf':  
1, 'min_samples_split': 5, 'n_estimators': 200}
```

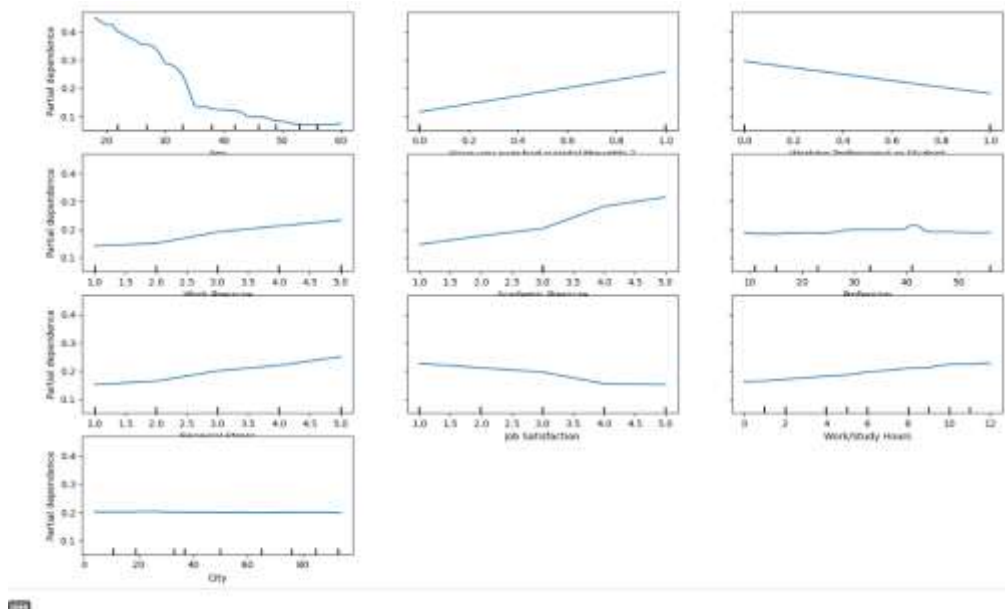
■ Visualization Graphs

we have plotted the n_estimators vs train accuracy to see the best n_estimator to use



The graph shows that the accuracy of the model starts to saturate starting from n_estimators=200

- Partial Dependence Plots
we have plotted the partial dependence between each feature and the output class (Depression) to see the relation between them



- Bias Variance Analysis

When we calculate the bias and variance to see how well the models is we got the following results:

Bias: 0.0627

Variance: 0.0081

- Results

Classification Report:

	precision	recall	f1-score	support
0	0.96	0.96	0.96	22986
1	0.82	0.84	0.83	5154
accuracy			0.94	28140
macro avg	0.89	0.90	0.89	28140
weighted avg	0.94	0.94	0.94	28140

Logistic Regression

Logistic regression is a fundamental classification algorithm in machine learning and statistics, widely used for predicting binary or categorical outcomes. logistic regression estimates the probability that an instance belongs to a particular class

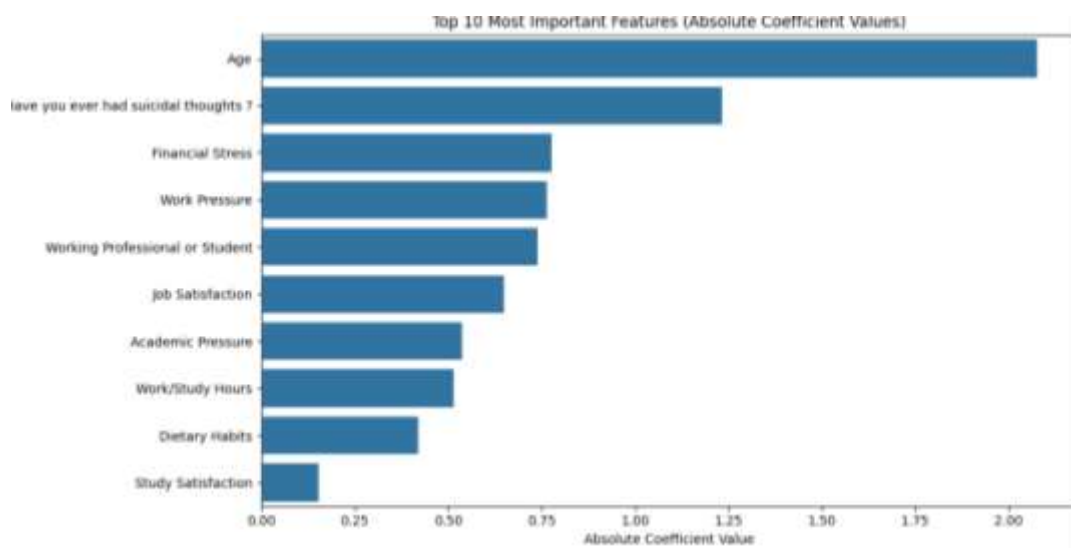
For our problem the two classes are (Yes/No)

Yes indicates that person has depression

No indicates that person doesn't have depression

➤ Feature Importance Plot

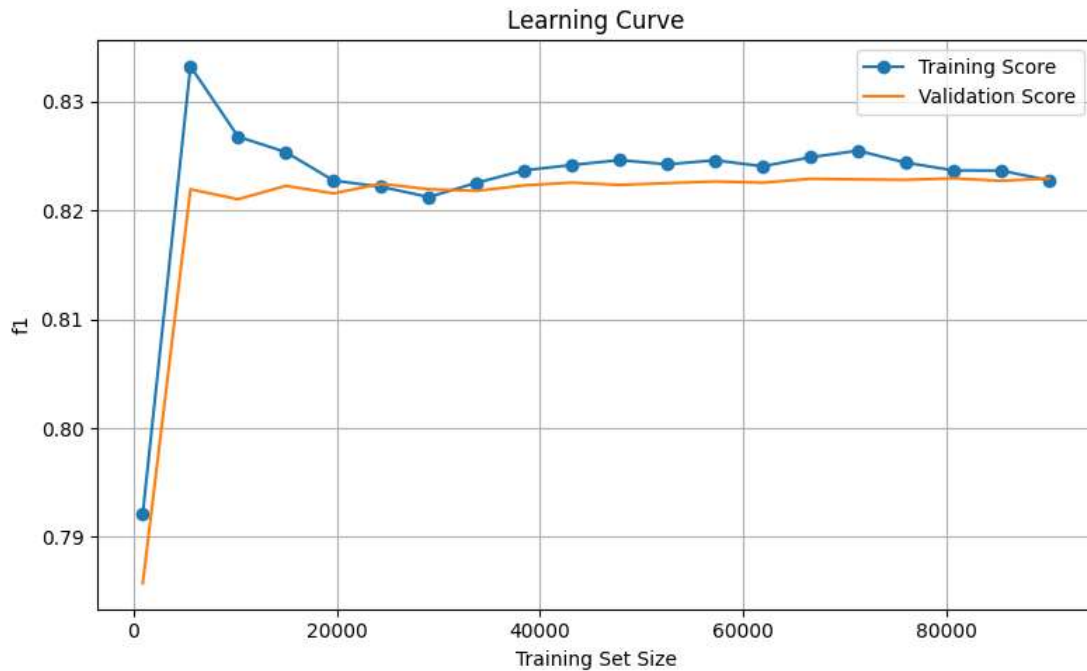
A feature importance plot shows the importance of each feature in the model. It can be used to identify the most important features and to understand the impact of each feature on the model's prediction.



From the plot we can conclude that the first feature “Age” is the most important one and the second is a little close to it and the following three feature are almost equal

➤ Learning Curves Plot

A learning curve is a plot that shows how a model's performance changes with varying amounts of training data. It helps diagnose underfitting or overfitting and understand whether more data would help.

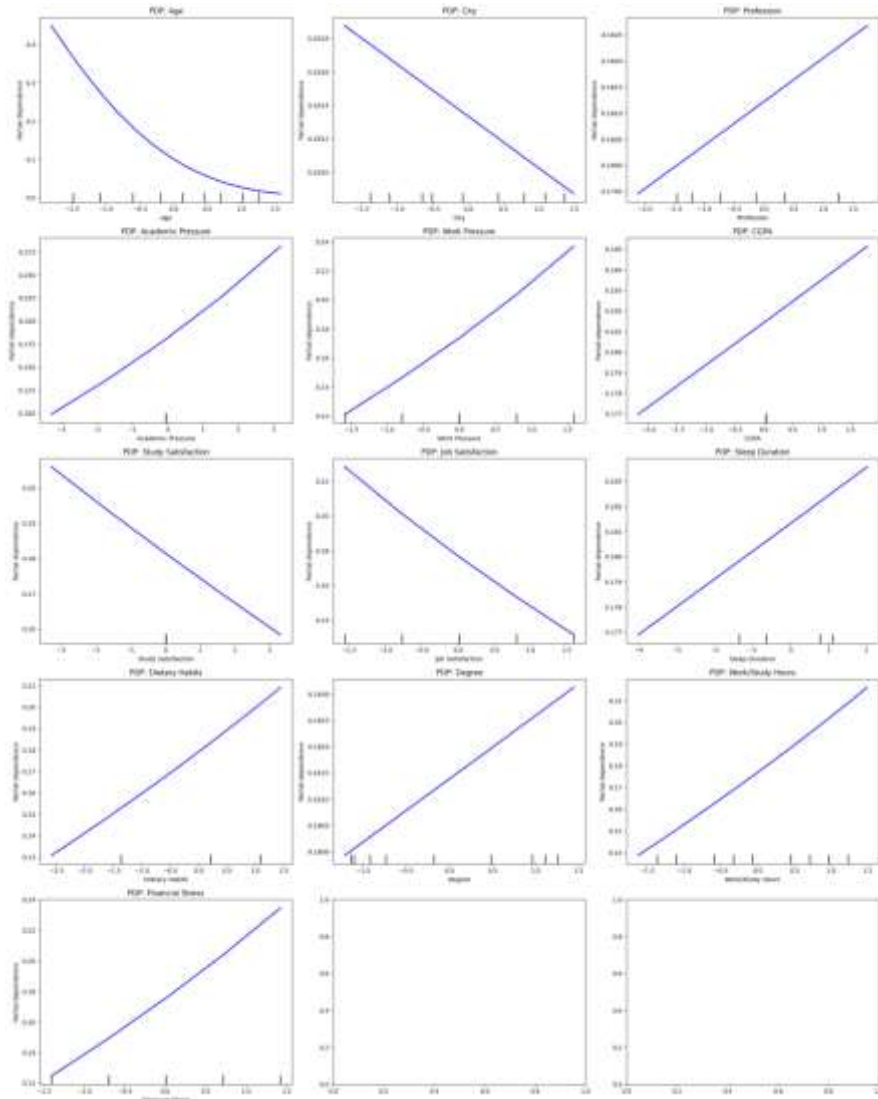


Observation Form The Plot:

- We used F1-score (Harmonic mean of precision & recall) as the dataset is highly imbalanced
- The training curve for small datasets is high due to overfitting in the beginning (because of the small data so it is easy for the model to overfit it)
- The validation curve is very low because of bad generalization in the beginning.
- With increasing dataset, the validation score improves and stabilizes.
- The gap between the validation and training indicates reduced overfitting.
- At the end of the graph it appears that almost the two curves have stabilized, So adding more data doesn't significantly improve validation performance

➤ Partial Dependence Plot

Using partial dependence plot We can see that the relationship between the model and each feature



So as we can see most of the features have linear relation between itself and the learned model

➤ Hyperparameter Tuning

We have applied a grid search to the logistic regression model using `GridSearchCV` function from `skit learn` library.

This is the parameter grid used :

```
# Define the parameter grid
param_grid = {
    'C': [0.01, 0.1, 1, 10, 100],      # Regularization strength (inverse)
    'penalty': ['l1', 'l2'],           # Type of regularization
    'solver': ['liblinear', 'saga'],    # solvers supporting l1 and l2
    'class_weight': [None, 'balanced'] # Try both
}
```

```
Data loaded successfully!
Fitting 5 folds for each of 40 candidates, totalling 200 fits

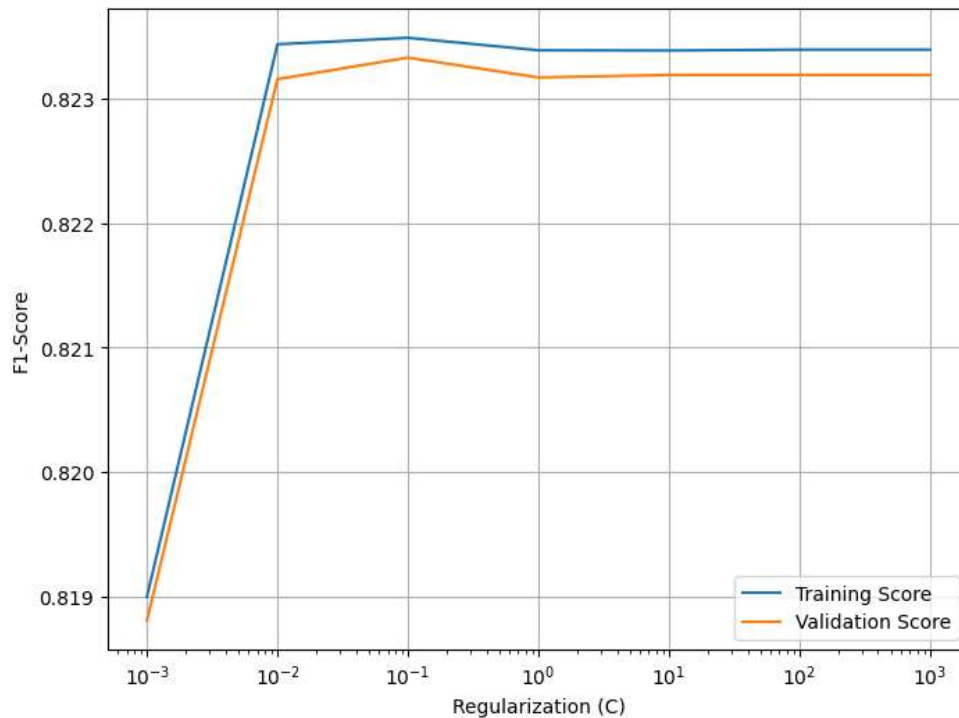
Best Parameters Found:
{'C': 0.1, 'class_weight': None, 'penalty': 'l1', 'solver': 'saga'}
```

From the result it is appeared that L1 regularization has outperformed the L2 regularization which means that the model has better generalization in the case of Lasso Regularization (implicit feature selection).

Despite class imbalance , weighting classes did not improve performance, implying the imbalance is manageable or the metric used (ROC_AUC) is less sensitive to it.

➤ Train-Validation Curve

Here is a Train-Validation Curve that we further used for the hyperparameter tuning process



From this curve it's obvious that at Regularization Strength (C) = 0.1 is the optimal regularization strength where below it the model is underfitting and above it the model is overfitting so this regularization strength is the most appropriate for this data

➤ Bias-variance Analysis

- MSE: 0.23049104
- Bias²: 0.22816216
- Variance: 0.00232888
- Noise: 1.51089558
- Bias² + Variance: 0.23049104
- Estimated Eout (MSE): 0.23049104

➤ Results

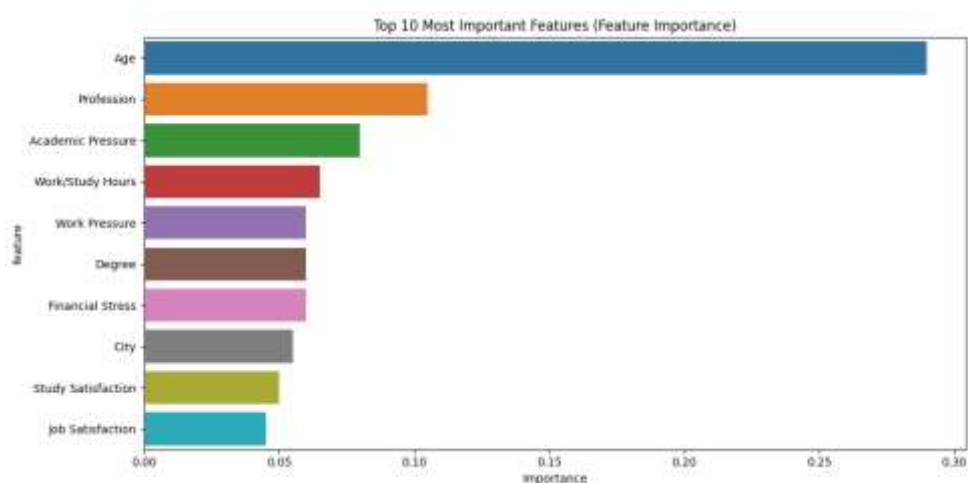
Logistic Regression Model Performance:				
Classification Report:				
	precision	recall	f1-score	support
0	0.96	0.97	0.96	22986
1	0.84	0.80	0.82	5154
accuracy			0.94	28140
macro avg	0.90	0.89	0.89	28140
weighted avg	0.94	0.94	0.94	28140

Adaboost

AdaBoost (Adaptive Boosting) is a powerful ensemble learning technique that combines multiple weak learners (typically decision trees with one split, called "decision stumps") into a strong classifier. It works by iteratively training models, giving more weight to misclassified samples in each round, and finally aggregating predictions through weighted voting.

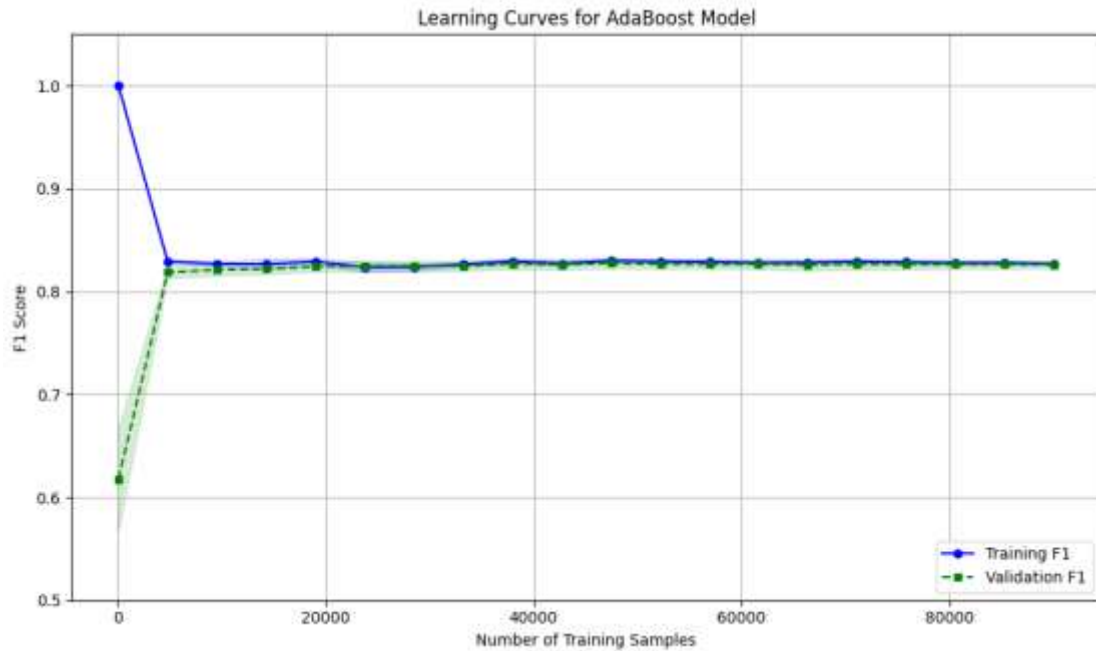
➤ Feature Importance Plot

A feature importance plot shows the importance of each feature in the model. It can be used to identify the most important features and to understand the impact of each feature on the model's predictions.



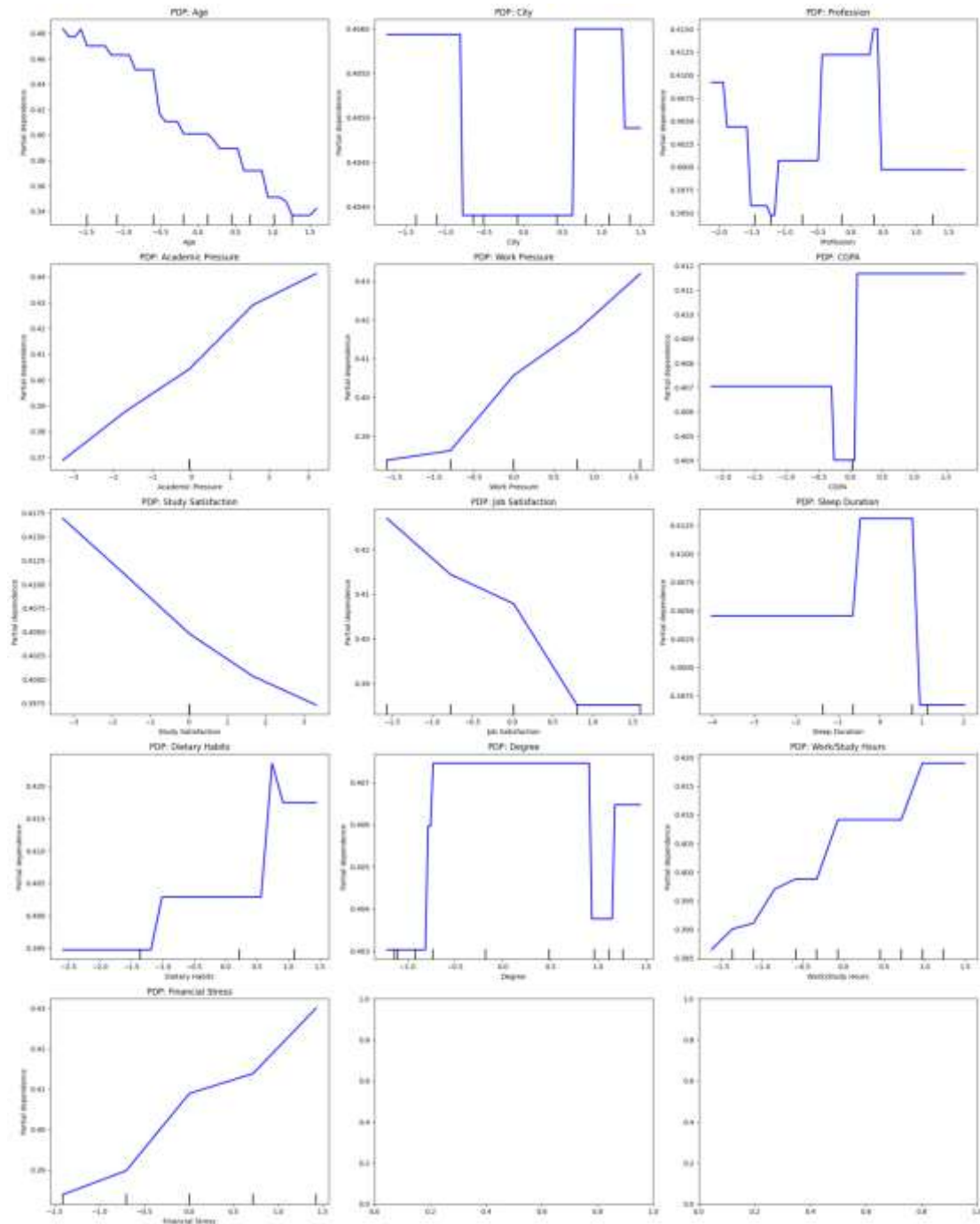
From the plot we can conclude that the first feature "Age" is the most important one and the other features are near to each other in importance.

➤ Learning Curves Plot



This plot shows that a dataset with at least 20000 number of training samples is enough for the model to learn. And training error (E_{in}) is very close to Validation Error (E_{out}) So this model generalizes well.

➤ Partial Dependence Plot



Some features have almost linear relation with the model like “Age”, “Academic Pressure”, “Work Pressure”, “Study Satisfaction”, “Work Satisfaction”, “Work/Study Hours” and “Financial Stress”, And another features have nonlinear relation like “City” and “Profession”.

➤ Hyper-parameter Tuning

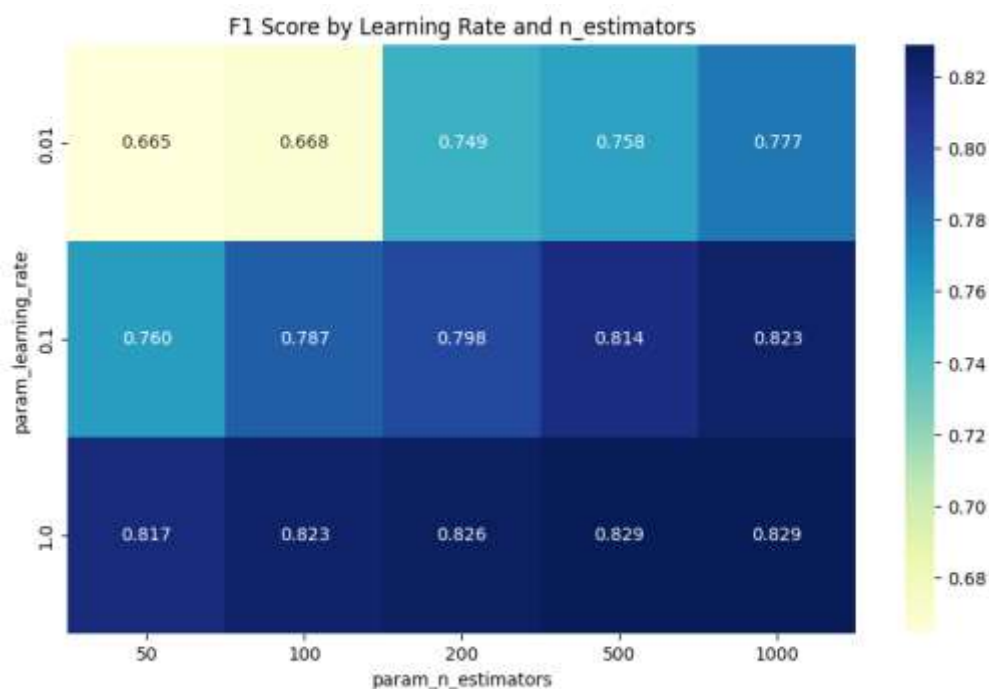
We have applied a grid search to the Adaboost model using `GridSearchCV` function from `skit learn` library.

This is the parameter grid used :

```
# Define the hyperparameter grid
param_grid = {
    'n_estimators': [50, 100, 200, 500, 1000],      # Number of weak learners
    'learning_rate': [0.01, 0.1, 1.0],             # Shrinkage factor
}
```

where “`n_estimators`” is the maximum number of estimators at which boosting is terminated. In other words, the number of boosting rounds (or weak learners) to train. “`learning_rate`” is the shrinkage parameter that controls the contribution of each classifier.

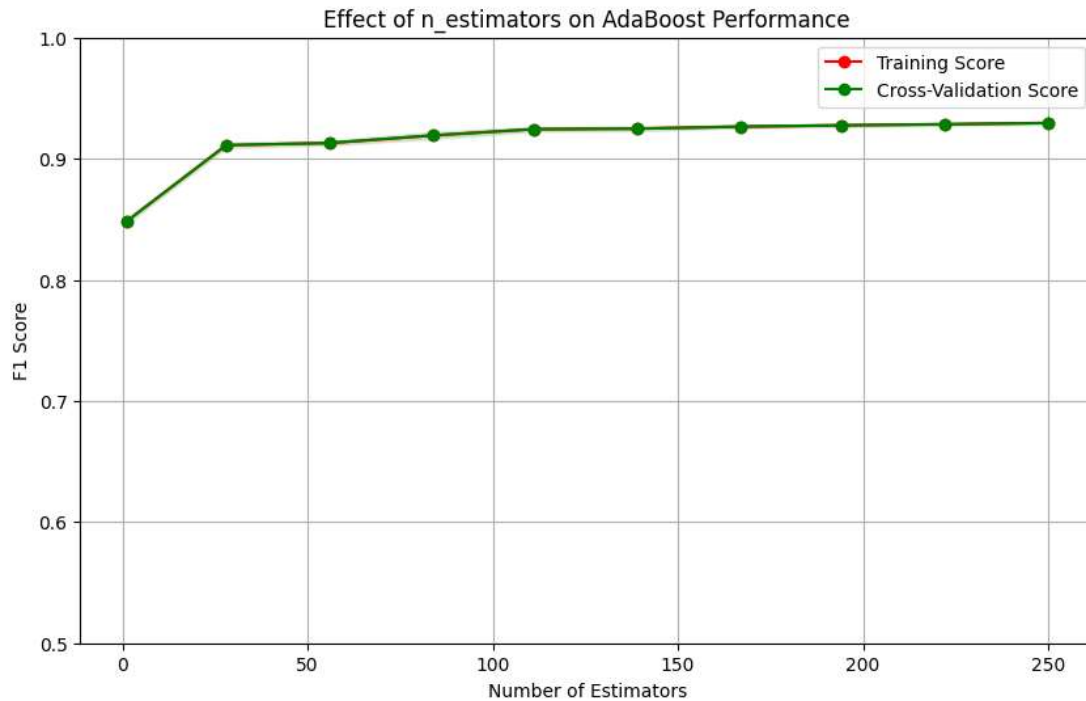
Here is heatmap visualizations of grid search result



we can see that some combinations give higher accuracy like `n_estimators = 50` and `learning rate = 1`

➤ Train-Validation Curve

Here is a plot showing how the model performance evolves with the increasing number of estimators.



We can see that for our problem using a lot of estimators isn't a good thing because that will increase the computations even, though there is no increase in the accuracy after 50 estimators.

➤ Results








Classification Report:					
	precision	recall	f1-score	support	
0	0.96	0.97	0.96	22986	
1	0.84	0.82	0.83	5154	
accuracy			0.94	28140	
macro avg	0.90	0.89	0.90	28140	
weighted avg	0.94	0.94	0.94	28140	

Conclusion

After we tried four models here is the brief summary of our models:

Model	Accuracy	F1 Score class 0	F1 Score class 1
ZeroR	0.7	0.9	0
SVM	0.94	0.96	0.82
Random Forest	0.94	0.96	0.83
Logistic Regression	0.94	0.96	0.82
Adaboost	0.94	0.96	0.83
XGboost	0.94	0.96	0.83

As the idea is from a competition on kaggle we looked at the leaderboard and found out that we are close to the maximum accuracy achieved

#	Team	Members	Score	Entries	Last	Solution
1	+ 273 Mahdi Ravaghi		0.94184	124	5mo	
2	+ 656 gougou1		0.94181	2	6mo	
3	+ 355 Martynov Andrey		0.94181	62	5mo	
4	+ 542 Jack Lee		0.94177	8	5mo	
5	+ 764 Curtis		0.94176	21	6mo	

Work Distribution

Team Member	Module
Rawan Mostafa Mahmoud	Feature Engineering, Adaboost
Fatma Ebrahim Sobhy	EDA, SVM
Menna Mohamed Abdelbaset	Baselines Models, Random Forest
Sara Bisheer Fekry	Logistic Regression, Adaboost