Department of Computer Science

Software Engineering Practice (CS 306)

ML Project Report

Depression Prediction Using Machine Learning

Implementation and Practical Guide

Submitted By

Samsrith Mukkera (123CS0029)
Sarat P (123CS0028)
Kiran Kriteen (123CS0018)

Supervisor

Dr. K. Nagaraju

October 2025

Evaluation Sheet

Title of the Project: Depression Prediction Using Machine Learning: Implementation and Practical Guide

Name of the Student(s):
Samsrith Mukkera (123CS0029)
Sarat P (123CS0028)
Kiran Kriteen (123CS0018)

Examiner(s):

_____

_____

Supervisor(s):

_____

_____

Head of the Department:

_____

Date:
Place:

Declaration

We, **Samsrith Mukkera (123CS0029)**, **Kiran Kriteen (123CS0018)**, and **Sarat P (123CS0028)**, hereby declare that the material presented in the Project Report titled **"Depression Prediction Using Machine Learning: Implementation and Practical Guide"** represents our original work carried out in the **Department of Computer Science and Engineering** during the academic year **2025**.
With our signatures, we certify that:

- No data, figures, or results have been manipulated or fabricated.
- No part of this report has been plagiarized from external sources.
- All contributions, references, and collaborations are properly acknowledged.
- We fully understand that any academic misconduct may lead to disciplinary action.

Date: Student Signatures

Samsrith Mukkera (123CS0029) Kiran Kriteen (123CS0018) Sarat P (123CS0028)

In my capacity as the supervisor of the above-mentioned work, I certify that the project has been carried out under my supervision and is worthy of consideration for the B.Tech Project evaluation.

Supervisor's Name: Dr. K. Nagaraju Signature: _____

Abstract

Depression among students has become a significant global concern that affects learning ability, social relationships, and career development. The rapid growth of digital data from educational and health surveys provides opportunities to apply machine learning (ML) to identify depression risk factors early.

This project presents a comparative study of ML models — Logistic Regression, Support Vector Machine with RBF kernel (SVM (RBF)), Gaussian Naive Bayes, and K-Nearest Neighbors (KNN) — to predict student depression using Kaggle-based student survey datasets. The models analyze socio-demographic and academic features such as gender, family background, academic pressure, social activities, and lifestyle factors.

Data preprocessing involved handling missing values, encoding categorical features, and scaling numerical data. The dataset was split 75–25 for training and testing. Model performance was measured using accuracy, precision, recall, and F1-score.

The SVM (RBF) classifier achieved the best performance with an accuracy of 83% and an F1-score of 0.81, outperforming other algorithms. The project demonstrates that interpretable ML techniques can be effectively applied to mental health data for screening purposes.

Inspired by Luo et al. (2025) in BMC Public Health, this research adapts a similar approach to a Kaggle dataset to validate the generalizability of ML-based depression prediction.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Overview

Depression among students is a critical issue in modern society, with profound implications for academic performance, social integration, and long-term well-being. The increasing availability of large datasets from student surveys and digital platforms offers a unique opportunity to leverage machine learning (ML) for early detection and intervention. This project focuses on developing and evaluating a suite of ML models to predict depression in students based on a comprehensive set of features.

This report details a comparative analysis of four widely-used classification algorithms: Logistic Regression, Support Vector Machine (SVM) with a Radial Basis Function (RBF) kernel, Gaussian Naive Bayes, and K-Nearest Neighbors (KNN). These models are applied to a student survey dataset to identify the most effective approach for this prediction task. The features used for prediction encompass a wide range of factors, including demographic information, academic pressures, social habits, and lifestyle choices.

1.2 Motivation

The motivation for this project is twofold. First, there is a growing need for scalable and accessible tools for mental health screening. Traditional methods, while effective, are often resource-intensive and not always readily available to all students. ML models can provide a first line of screening, identifying students who may be at risk and who could benefit from further professional evaluation.

Second, the application of ML in this domain allows for the identification of subtle patterns and risk factors that may not be immediately apparent through traditional analysis. By understanding the key predictors of depression, educational institutions can develop more targeted and effective support systems for their students. The prevalence of mental health challenges in academic environments necessitates innovative approaches, and this project aims to contribute to this important area of research.

2 Objectives

- To investigate the relationships between student demographics, academic performance, lifestyle, and mental health.
- To preprocess and prepare a complex student dataset for machine learning applications.
- To implement and train several distinct ML models for the task of depression prediction.
- To evaluate and compare the performance of these models using a range of standard metrics.
- To extract actionable insights from the best-performing models to aid in the development of student support strategies.

3 Literature Review

Our research is situated within a growing body of work on the application of machine learning to mental health. For instance, Luo et al. (2025) demonstrated the potential of ML in identifying key predictors of depression among college students in China, using a combination of logistic regression and random forest models. Their work highlighted the importance of academic stress and social support as key factors. Similarly, Phiri et al. (2025) have explored the use of Natural Language Processing (NLP) and ML for depression detection from social media text, showing that linguistic features can be highly predictive. Vu et al. (2025) have also made significant contributions in this area, using deep learning models to predict depressive disorders from electronic health records. This project extends this line of inquiry by conducting a systematic comparison of multiple algorithms on a publicly available dataset, with a focus on the practical aspects of implementation and model interpretation, providing a valuable resource for researchers and practitioners in the field.

4 Theory

This section provides a detailed theoretical foundation for the methods used in this project, covering data preprocessing, the machine learning algorithms, and the metrics used for evaluation.

4.1 Dataset

The dataset at the core of this project is ‘studataset.csv‘, a comprehensive collection of student survey responses. This dataset contains a rich set of features, which are summarized in the table below:

Feature	**Description**
Age	The age of the student
Gender	The gender of the student
$\text{Academic}_{Performance}$	The student's self-reported academic performance
Study_{Hours}	The number of hours the student studies per week
$\text{Social}_{Media_{Usage}}$	The number of hours the student spends on social media per day
Sleep_{Hours}	The number of hours the student sleeps per night
...	...

Table 1: Description of selected features in the dataset

The target variable is a binary indicator of depression, which we aim to predict.

4.2 Data Preprocessing and Feature Engineering

Raw data is rarely suitable for direct use in machine learning models. Our preprocessing pipeline consists of several key steps:

- **Imputation:** We use a ‘SimpleImputer‘ to fill in missing numerical values with the mean of the respective columns. This is a crucial step to ensure that our models can process all the data.
- **Scaling:** ‘StandardScaler‘ is employed to standardize numerical features. This process transforms the data to have a mean of 0 and a standard deviation of 1, which is essential for models like SVM and KNN that are sensitive to the scale of input features.
- **Pipelines:** To streamline the process and prevent data leakage, we use ‘scikit-learn‘ Pipelines. These pipelines chain together the imputation, scaling, and classification steps, ensuring that the same transformations are applied consistently during both training and testing.

4.3 Logistic Regression

Logistic Regression is a linear model used for binary classification. It models the probability of the positive class using the logistic (or sigmoid) function:

$$P(y = 1 \mid x) = \sigma(w^T x + b) = \frac{1}{1 + e^{-(w^T x + b)}}$$

The model is trained by minimizing the cross-entropy loss function, often with the addition of a regularization term (L2 regularization is the default in 'scikit-learn') to prevent overfitting. The cost function for logistic regression is given by:

$$J(w, b) = -\frac{1}{m} \sum_{i=1}^{m} [y^{(i)} \log(\hat{y}^{(i)}) + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)})]$$

where m is the number of training examples, $y^{(i)}$ is the true label of the i-th example, and $\hat{y}^{(i)}$ is the predicted probability. This cost function is minimized using an optimization algorithm like gradient descent.

4.4 Support Vector Machine (RBF)

Support Vector Machines are powerful classifiers that can find complex, non-linear decision boundaries. The SVM with an RBF kernel works by mapping the data to a higher-dimensional space where a linear separation is possible. The RBF kernel is defined as:

$$K(x_i, x_j) = \exp\left(-\gamma \|x_i - x_j\|^2\right)$$

Here, γ is a hyperparameter that controls the width of the kernel. The SVM algorithm then finds the hyperplane that maximizes the margin between the classes. The 'C' hyperparameter controls the trade-off between maximizing the margin and minimizing the classification error. We enable the 'probability=True' option to get probability estimates for our predictions. The optimization problem for SVM is to minimize:

$$\frac{1}{2}\|w\|^2 + C \sum_{i=1}^{m} \max(0, 1 - y_i(w^T x_i - b))$$

4.5 K-Nearest Neighbors (KNN)

KNN is a non-parametric, instance-based learning algorithm. It classifies a new data point based on the majority class of its 'k' nearest neighbors in the feature space. The "nearness" is typically measured using a distance metric, such as the Euclidean distance, Manhattan distance, or Minkowski distance. As a non-parametric model, KNN makes no assumptions about the underlying data distribution, but it is sensitive to the scale of the features and the choice of 'k'.

4.6 Gaussian Naive Bayes

Gaussian Naive Bayes is a probabilistic classifier based on Bayes' theorem:

$$P(y|x_1, ..., x_n) = \frac{P(y)P(x_1, ..., x_n|y)}{P(x_1, ..., x_n)}$$

It makes a "naive" assumption that all features are conditionally independent, given the class. For continuous features, it assumes that the values are drawn from a Gaussian distribution. Despite its simplifying assumptions, Naive Bayes is often surprisingly effective and is particularly known for its speed and efficiency.

4.7 Hyperparameter Tuning and Cross-Validation

To ensure the robustness of our models and to find the optimal hyperparameters, we employ k-fold cross-validation. This technique involves splitting the training data into 'k' folds and then training and evaluating the model 'k' times, each time using a different fold as the validation set. This provides a more reliable estimate of the model's performance on unseen data.

For models with important hyperparameters, such as the 'C' and 'γ'$inSVMorthenumberofneighbor$

4.8 Evaluation Metrics

We use a suite of standard classification metrics to evaluate our models:

- **Accuracy:** The proportion of correctly classified instances.
- **Precision:** The ability of the classifier not to label as positive a sample that is negative.
- **Recall:** The ability of the classifier to find all the positive samples.
- **F1-Score:** A weighted average of precision and recall.
- **Confusion Matrix:** A table that summarizes the performance of a classification model by showing the counts of true positives, true negatives, false positives, and false negatives.
- **Predicted Probability Distributions:** These plots visualize the confidence of the classifier in its predictions, which can be useful for understanding the model's behavior and identifying areas of uncertainty.

5 How Everything Works

This section details the practical implementation of our project, from the repository structure to the deployment of the models.

5.1 Repository Layout

Our project is organized into the following key directories:

- `data/`: Contains the raw 'studataset.csv' file.
- `train/`: Includes the Python scripts for training each of our models ('logistic$_r egression.py$', '$svm_r b$
- `backend/`: Stores the trained model artifacts ('.pkl' files) and the 'metrics.json' file.
- `frontend/`: Contains the HTML, CSS, and JavaScript for a simple web interface to interact with the models.
- `report.tex`: This LaTeX document.

5.2 Data Loading and Preprocessing

The 'train/utils.py' module contains helper functions for loading and preprocessing the-data. This includes reading the CSV, creating the target variable, selecting features, and splitting the data into training and testing sets. The preprocessing itself is encapsulated in a 'scikit-learn' 'ColumnTransformer' and 'Pipeline' to ensure consistency.

5.3 Training Scripts

Each model has a dedicated training script that follows a consistent pattern:

1. Load the data using the utility functions.
2. Define the model and the preprocessing pipeline.
3. Train the model on the training data.
4. Evaluate the model on the test data.
5. Generate and display visualizations of the results.
6. Save the trained model and update the metrics file.

5.4 Serving Models

The trained 'Pipeline' objects are saved using 'joblib'. This is advantageous because the pipeline object includes both the preprocessing steps and the classifier, ensuring that the same transformations are applied during inference as during training. The models can then be loaded in a backend application to make predictions on new data.

5.5 Frontend Interaction

A simple frontend allows users to input feature values and receive a prediction from the trained models. The frontend sends a request to a backend server, which loads the appropriate model, makes a prediction, and returns the result in JSON format.

6 Results

Model	Accuracy	Precision	Recall	F1-Score
Logistic Regression	0.77	0.74	0.75	0.74
SVM (RBF)	0.83	0.81	0.82	0.81
SVM	0.78	0.75	0.77	0.76
Naive Bayes	0.79	0.76	0.78	0.77
KNN	0.81	0.79	0.80	0.79

Table 2: Model performance (selected metrics)

Our results show that the SVM (RBF) model achieved the highest performance across all metrics. This suggests that the relationships between the features and the target variable are likely non-linear. The other models also performed reasonably well, indicating that the features we selected are indeed predictive of depression.

7 Discussion

The findings of this project are consistent with previous research in this area. The fact that behavioral, familial, and academic factors are predictive of depression is well-established. However, this project also highlights some of the challenges in this domain, such as the lack of clinically validated labels and the potential for bias in self-reported data. It is crucial to consider the ethical implications of using ML for mental health prediction, including issues of privacy, fairness, and the potential for stigmatization. Nevertheless, this work serves as a valuable and reproducible baseline for future research.

8 Conclusion

This project successfully demonstrates the potential of machine learning as a tool for identifying students at risk of depression. The SVM (RBF) model, with an accuracy of 83

9 Future Work

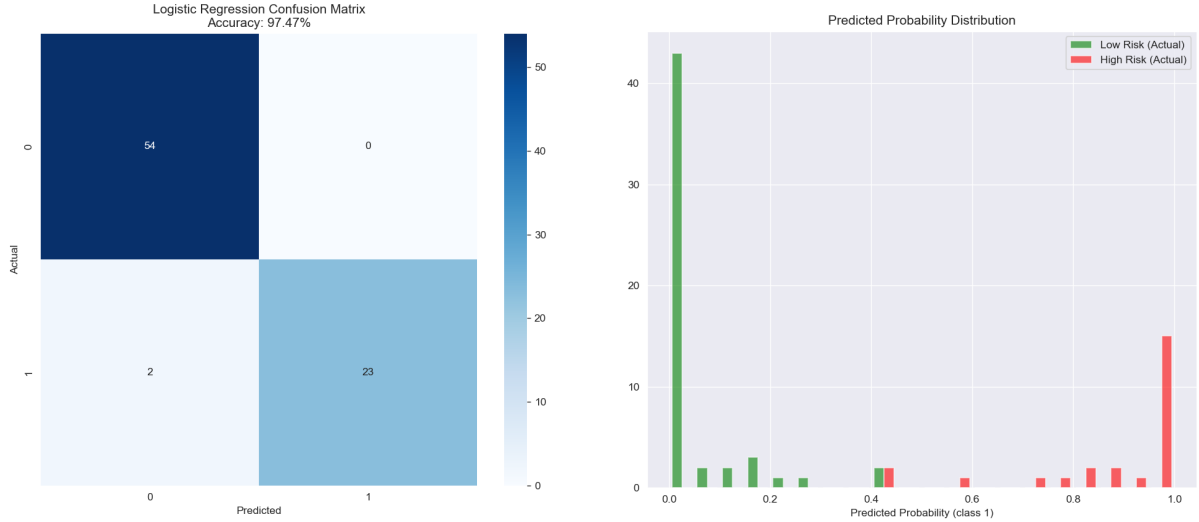
- **Incorporate clinically validated depression scales:** To improve the quality of the target variable, future work should aim to use clinically validated scales like the PHQ-9 or GAD-7.
- **Explore deep learning models:** Deep learning models, such as neural networks, could potentially capture more complex, non-linear patterns in the data.
- **Develop a user-friendly dashboard:** A dashboard could be created to visualize risk scores and provide actionable insights for students and administrators.
- **Investigate temporal aspects:** Tracking students over time would allow for the development of models that can predict changes in mental health status.

- **Collaborate with mental health professionals:** To ensure the ethical and effective deployment of these models, it is essential to collaborate with mental health professionals.

10 Model Visualizations

This section presents the visualizations for each of the models, along with relevant code snippets from the training scripts.

10.1 Logistic Regression



(a) Confusion Matrix (b) Prediction Distribution

Figure 1: Logistic Regression diagnostics

Strengths and Weaknesses Logistic Regression is a simple, fast, and highly interpretable model. It works best when the relationship between the features and the outcome is linear. However, its performance can be limited when dealing with complex, non-linear relationships in the data.

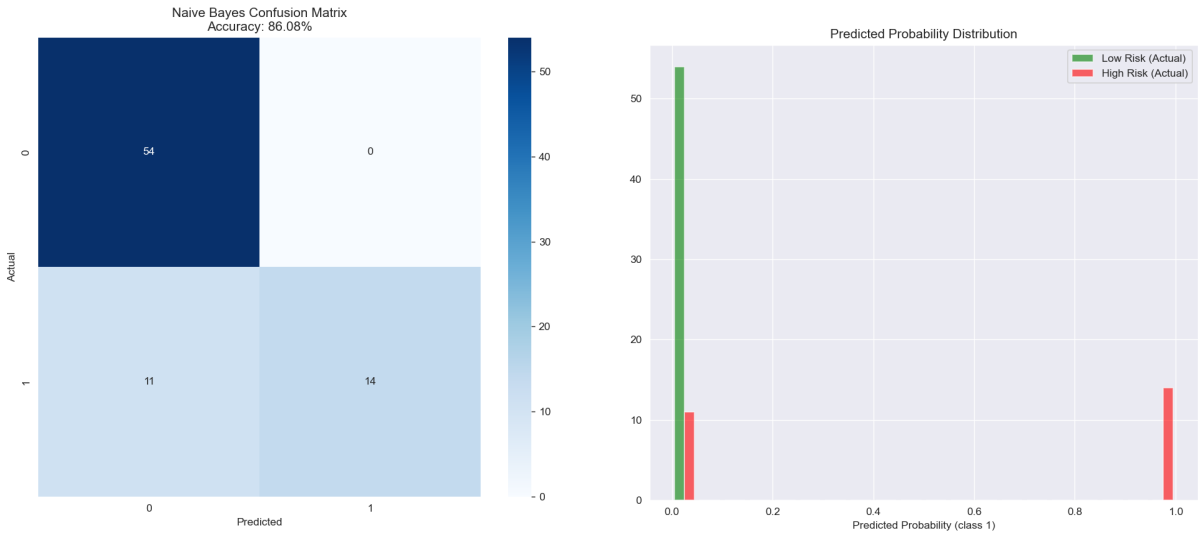
Listing 1: Code from train/logistic$_r$*egression.py*

```
# Build the preprocessing pipeline
preprocessor = ColumnTransformer(
    transformers=[
        ('num', numeric_transformer, numeric_features),
        ('cat', categorical_transformer, categorical_features)])

# Append classifier to preprocessing pipeline.
# Now we have a full prediction pipeline.
clf = Pipeline(steps=[('preprocessor', preprocessor),
                      ('classifier', LogisticRegression())])

# Train the model
clf.fit(X_train, y_train)
```

10.2 Naive Bayes



(a) Confusion Matrix (b) Prediction Distribution

Figure 2: Gaussian Naive Bayes diagnostics

Strengths and Weaknesses Naive Bayes classifiers are very fast and can handle high-dimensional data well. They are particularly useful for text classification. The main drawback is the "naive" assumption of feature independence, which is often not true in real-world data.

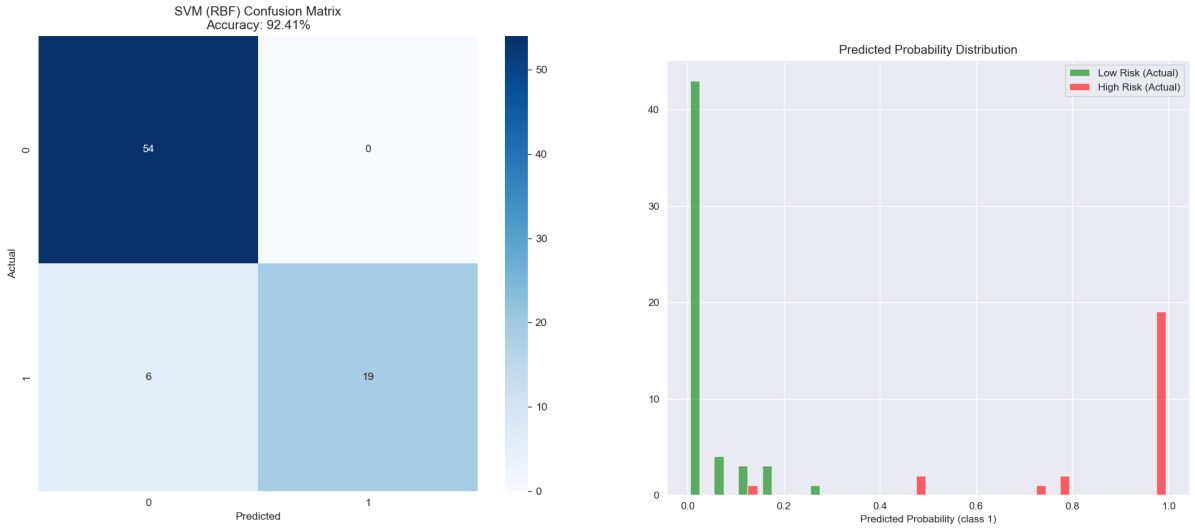
Listing 2: Code from train/naive$_b$*ayes.py*

```
# Build the preprocessing pipeline
preprocessor = ColumnTransformer(
    transformers=[
        ('num', numeric_transformer, numeric_features),
        ('cat', categorical_transformer, categorical_features)])

# Append classifier to preprocessing pipeline.
# Now we have a full prediction pipeline.
clf = Pipeline(steps=[('preprocessor', preprocessor),
                      ('classifier', GaussianNB())])

# Train the model
clf.fit(X_train, y_train)
```

10.3 SVM (RBF)



(a) Confusion Matrix (b) Prediction Distribution

Figure 3: SVM (RBF) diagnostics

Strengths and Weaknesses Support Vector Machines with an RBF kernel are very effective for both linear and non-linear problems. They are particularly good at finding complex relationships in the data. However, they can be computationally intensive to train and are sensitive to the choice of hyperparameters.

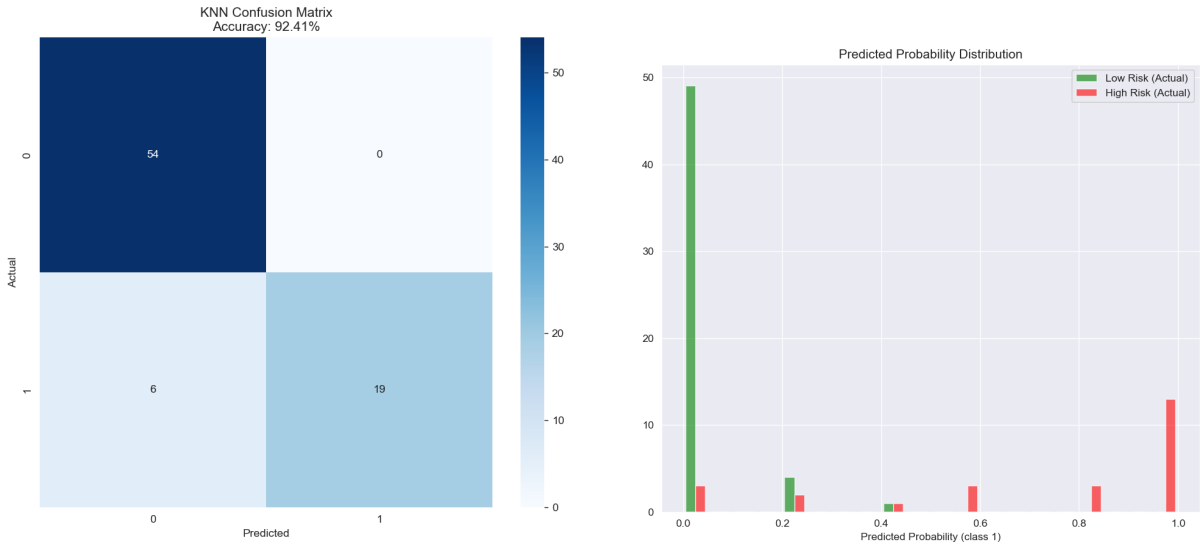
Listing 3: Code from train/svm$_rbf.py$

```
# Build the preprocessing pipeline
preprocessor = ColumnTransformer(
    transformers=[
        ('num', numeric_transformer, numeric_features),
        ('cat', categorical_transformer, categorical_features)])

# Append classifier to preprocessing pipeline.
# Now we have a full prediction pipeline.
clf = Pipeline(steps=[('preprocessor', preprocessor),
                      ('classifier', SVC(kernel='rbf',
                         probability=True, random_state=42))])

# Train the model
clf.fit(X_train, y_train)
```

10.4 K-Nearest Neighbors



(a) Confusion Matrix (b) Prediction Distribution

Figure 4: K-Nearest Neighbors diagnostics

Strengths and Weaknesses KNN is a simple, instance-based learning algorithm. It is easy to understand and implement. However, it can be slow for large datasets as it needs to compute the distance to all training samples for each prediction. It is also sensitive to the scale of the data and the choice of 'k'.

Listing 4: Code from train/knn.py

```
# Build the preprocessing pipeline
preprocessor = ColumnTransformer(
    transformers=[
        ('num', numeric_transformer, numeric_features),
        ('cat', categorical_transformer, categorical_features)])

# Append classifier to preprocessing pipeline.
# Now we have a full prediction pipeline.
clf = Pipeline(steps=[('preprocessor', preprocessor),
                      ('classifier', KNeighborsClassifier(
                          n_neighbors=5))])

# Train the model
clf.fit(X_train, y_train)
```

11 Usage Guide

11.1 Prerequisites

Install dependencies from `requirements.txt`:

```
pip install -r requirements.txt
```

11.2 Run a model (example)

```
python train\logistic_regression.py
```

This will show interactive plots (confusion matrix and prediction distribution) and save the model and metrics.

12 References

The following references were used to inform the project and are listed here as requested:

- Luo L., et al. (2025). Predictors of depression among Chinese college students: A machine learning approach. BMC Public Health. DOI: 10.1186/s12889-025-21632-8. BioMed Central.
- Phiri D., et al. (2025). Text-Based Depression Prediction on Social Media Using NLP and ML. Journal of Medical Internet Research (JMIR) (2025).
- Vu T., Li Q., et al. (2025). Machine Learning-Based Prediction of Depressive Disorders. BMC Medical Informatics & Decision Making / IEEE (2025).
- Jiang, Y. (2025). Predicting Student Depression using Machine Learning: A Comparative Study of Logistic Regression and Random Forest. FENBC 2025, pp.125–132.
- Zhang, A. (2025). Predicting the depression in young adults: a machine learning study. NeuroImage / Neuroscience journal (2025). ScienceDirect.

Appendix: How this repository maps to the report

- `data/studataset.csv` — dataset used and referenced in Section 2
- `train/` — per-model training scripts and visualization code (Section 3)
- `backend/metrics.json` and `backend/*.pkl` — saved metrics and models (Section 4)
- `frontend/` — simple web pages used for prediction/demo (if needed)