
DEPRESSION DETECTION
Machine Learning Model
Presented By:
Ananya Singh(2205790)
Kreetarth Shubham(22051170)
Adnan Imam(22051225)
Krishna Agrawal(22051253)
Shrestha Srivastava(22051281)

Introduction & Objective

A

The Depression Detection Model aims to identify signs of depression through the analysis of social media data. By leveraging machine learning techniques, this model seeks to provide early warnings for mental health issues based on user-generated content.

B

The primary objective of this project is to design and implement a machine learning-based system capable of detecting depression in individuals. This system will assist mental health professionals by providing timely insights and support for early intervention.

C

The methodology for this project involves a structured approach to developing a machine learning model for depression detection. This includes defining the problem, selecting appropriate algorithms, and outlining the steps for implementation.

A

The Depression Detection Model aims to identify signs of depression through the analysis of social media data. By leveraging machine learning techniques, this model seeks to provide early warnings for mental health issues based on user-generated content.

B

The primary objective of this project is to design and implement a machine learning-based system capable of detecting depression in individuals. This system will assist mental health professionals by providing timely insights and support for early intervention.

C

The methodology for this project involves a structured approach to developing a machine learning model for depression detection. This includes defining the problem, selecting appropriate algorithms, and outlining the steps for implementation.

Project Setup

A

Project Dataflow

B

Packages and Libraries Installed

- `pandas` : Used for reading data from .csv file
- `numpy` : Used for fast numerical operations, arrays, matrices, and mathematical functions
- `scikit-learn` : A machine learning library with tons of tools. You mentioned two submodules:
  - `metrics` ->
    - `confusion_matrix` : A 2D matrix showing true vs predicted values.
    - `accuracy_score` : Calculates the percentage of correct predictions.
    - `ConfusionMatrixDisplay` : Visual tool to plot the confusion matrix.
    - `classification_report` : Gives precision, recall, f1-score, support for each class.
  - `model_selection` ->
    - `train_test_split` : Splits your data into training and testing sets.
  - `feature_extraction.text` ->
    - `TfidfVectorizer` : Converts text into TF-IDF vectors where,TF = term frequency, IDF = inverse document frequency. Helps highlight important words in a document.
- `better_profanity` : A Python library to detect and censor profane words in text.
  - `nltk tools` ->
    - `stem` : Reduces words to their root form

C

Connecting Google Drive

```
from google.colab import drive
```

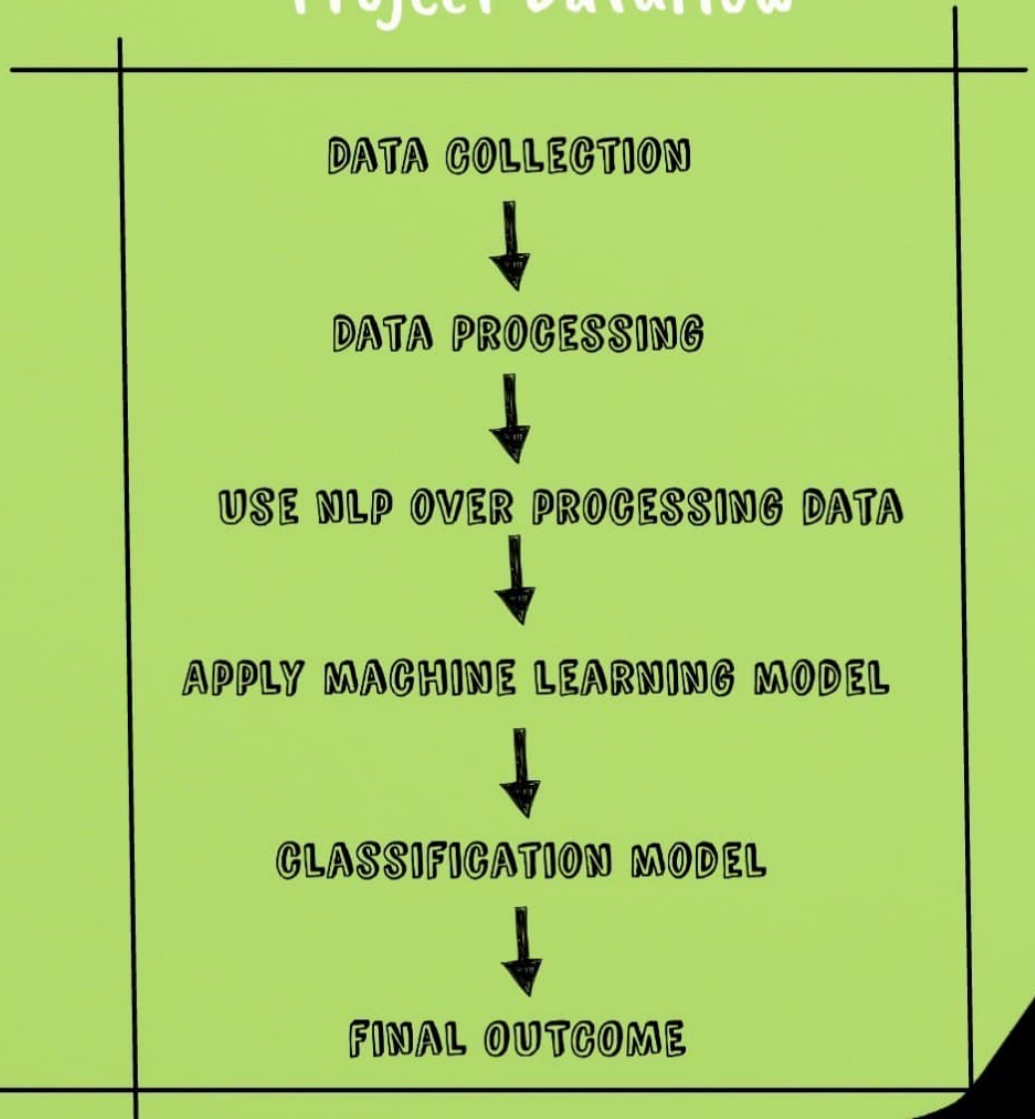
- This line imports the drive module from the google.colab package.
- It enables access to Google Drive functionality within the Colab environment.

```
drive.mount('/content/drive')
```

- This command mounts your Google Drive to the Colab file system at the path /content/drive.
- Once mounted, all your Google Drive files and folders are accessible just like a local file system.

A

Project Dataflow



B

Packages and Libraries Installed

- pandas : Used for reading data from .csv file
- numpy : Used for fast numerical operations, arrays, matrices, and mathematical functions
- scikit-learn : A machine learning library with tons of tools. You mentioned two submodules:
  - metrics ->
    - confusion_matrix : A 2D matrix showing true vs predicted values.
    - accuracy_score : Calculates the percentage of correct predictions.
    - ConfusionMatrixDisplay : Visual tool to plot the confusion matrix.
    - classification_report : Gives precision, recall, f1-score, support for each class.
  - model_selection ->
    - train_test_split : Splits your data into training and testing sets.
  - feature_extraction.text ->
    - TfidfVectorizer : Converts text into TF-IDF vectors where,TF = term frequency, IDF = inverse document frequency. Helps highlight important words in a document.
- better_profanity : A Python library to detect and censor profane words in text.
  - nltk tools ->
    - stem : Reduces words to their root form

C

Connecting Google Drive

```
from google.colab import drive
```

- This line imports the drive module from the google.colab package.
- It enables access to Google Drive functionality within the Colab environment.

```
drive.mount('/content/drive')
```

- This command mounts your Google Drive to the Colab file system at the path /content/drive.
- Once mounted, all your Google Drive files and folders are accessible just like a local file system.

A Tools and Stemming

with Tools Used

- Stopwords : Stopwords are words like "is", "and", "the" that usually don't carry much meaning and are often removed from text during preprocessing.
- Stemming is the process of reducing a word to its base or root form, which may not always be a real word, but is good enough for comparison or grouping.
  - We have used SnowballStemmer() of the nltk.stem module
  - For example :
    - running -> run
    - playing -> play
    - crying -> cry

B Data Cleaning Functions

keep_alpha(wrig) – Removing URLs, Numbers, Special Characters

- re.sub(r"http\S+", "", s): Removes hyperlinks (URLs starting with http or https)
- re.sub(["a-zA-Z\s]', '', ...): Removes all characters that are not alphabet letters or spaces (removes numbers and special characters)
- re.sub('\n', '', ...): Removes newline characters
- This function ensures that only clean, readable text remains for further processing

nlp_preprocessing(wrig) – Text Normalization

- Lowercasing: Standardizes all words to lowercase for uniform comparison (e.g., "Sad" and "sad" are treated the same).
- Stopwords Removal: Removes common words (like "is", "and", "the") that don't add much meaning to the text.
- Stemming: Reduces words to their root form (e.g., "crying" -> "cry", "happily" -> "happi"), helping to generalize word forms.

Data Cleaning

the process of identifying and correcting or removing incorrect, incomplete, duplicate, or irrelevant data within a dataset to improve its quality and usability for analysis or decision-making.

Tools and Stemming

nltk Tools Used

- stopwords : Stopwords are words like "is", "and", "the" that usually don't carry much meaning and are often removed from text during preprocessing.
- Stemming is the process of reducing a word to its base or root form, which may not always be a real word, but is good enough for comparison or grouping.
    - We have used SnowballStemmer() of the nltk.stem module
    - For example :
        - running -> run
        - playing -> play
        - crying -> cry

B Data Cleaning Functions

keep_alpha(msg) – Removing URLs, Numbers, Special Characters

- re.sub(r"http\S+", "", s): Removes hyperlinks (URLs starting with http or https)
- re.sub('[^a-zA-Z\s]', '', ...): Removes all characters that are not alphabet letters or spaces (removes numbers and special characters)
- re.sub('\n', '', ...): Removes newline characters
- This function ensures that only clean, readable text remains for further processing

nlp_preprocessing(msg) – Text Normalization

- Lowercasing: Standardizes all words to lowercase for uniform comparison (e.g., "Sad" and "sad" are treated the same).
- Stopwords Removal: Removes common words (like "is", "and", "the") that don't add much meaning to the text.
- Stemming: Reduces words to their root form (e.g., "crying" -> "cry", "happily" -> "happi"), helping to generalize word forms.


Importing Dataset
pd.read_csv(DATASET_FILE)
• This line loads a dataset from a CSV (Comma-Separated Values) file into a pandas DataFrame.
dropna()
• This line removes any rows in the DataFrame that contain missing (NaN) values.
• This is important for ensuring data quality which reduces future errors in tasks like text analysis.
sample(frac=1).reset_index()
• This line shuffles the rows of the DataFrame randomly. Shuffling is often done to avoid bias, especially when training models, ensuring that the data is randomly distributed.
A
Data Preprocessing
NLP Preprocessing
astype(str)
• This converts every value in the Tweet column to a string type.
str.strip()
• Removes any leading or trailing whitespace characters (like spaces, tabs, or newlines) from each Tweet.
apply(beep_alpha)
• This applies the beep_alpha function to each Tweet, which:
• Removes URLs
• Removes all non-alphabetic characters (numbers, punctuation, symbols)
• Removes newline characters
apply(nlp_preprocessing)
• Applies the nlp_preprocessing function to each Tweet, which:
• Converts text to lowercase
• Removes stopwords (common words like "the", "and", "is")
• Applies stemming (reduces words to their root form, e.g., "running" -> "run")
B
Vectorization
TfidfVectorizer()
• This creates an instance of the TfidfVectorizer from sklearn.feature_extraction.text.
• It will be used to transform the cleaned text into numerical vectors (features).
fit_transform(df['text'].values)
• fit_transform() learns the vocabulary from the text data (X) and then converts it into a TF-IDF(Term Frequency Inverse Document Frequency) weighted matrix (transform).
• X is a sparse matrix where each row represents a Tweet, and each column is a unique word from the entire dataset.
y = df['depressed'].values
• Extracts the labels (target variable), which indicate whether a Tweet shows signs of depression (1) or not (0) and is stored as a array.
C

Importing Dataset

pd.read_csv(DATASET_FILE)

- This line loads a dataset from a CSV (Comma-Separated Values) file into a pandas DataFrame.

dropna()

- This line removes any rows in the DataFrame that contain missing (NaN) values.
- This is important for ensuring data quality which reduces future errors in tasks like text analysis.

sample(frac=1).reset_index()

- This line shuffles the rows of the DataFrame randomly. Shuffling is often done to avoid bias, especially when training models, ensuring that the data is randomly distributed.

NLP Preprocessing

astype(str)

- This converts every value in the tweet column to a string type.

str.strip()

- Removes any leading or trailing whitespace characters (like spaces, tabs, or newlines) from each tweet.

apply(keep_alpha)

- This applies the keep_alpha function to each tweet, which:
  - Removes URLs
  - Removes all non-alphabetic characters (numbers, punctuation, symbols)
  - Removes newline characters

apply(nlp_preprocessing)

- Applies the nlp_preprocessing function to each tweet, which:
- Converts text to lowercase
- Removes stopwords (common words like "the", "and", "is")
- Applies stemming (reduces words to their root form, e.g., "running" -> "run")

Vectorization

TfidfVectorizer()

- This creates an instance of the TfidfVectorizer from sklearn.feature_extraction.text.
- It will be used to transform the cleaned text into numerical vectors (features).

fit_transform(df['tweet'].values)

- fit_transform() learns the vocabulary from the text data (fit) and then converts it into a TF-IDF(Term Frequency Inverse Document Frequency) weighted matrix (transform).
- X is a sparse matrix where each row represents a tweet, and each column is a unique word from the entire dataset.

y = df['depressed'].values

- Extracts the labels (target variable), which indicates whether a tweet shows signs of depression (1) or not (0) and is stored as a array.

C

Model Training

A

Logistic Regression

- Creating instance of Logistic Regression Model imported from the linear_model submodule of sklearn:
  - LogisticRegression()
- Training the model on the Training Data set using:
  - fit()
- Predictions made on the Test Set using:
  - predict()

B

Decision Tree

- Creating instance of Decision Tree Model imported from the tree submodule of sklearn:
  - DecisionTreeClassifier()
- Training the model on the Training Data set using:
  - fit()
- Predictions made on the Test Set using:
  - predict()

C

Support Vector Machine

- Creating instance of Support Vector Machine Model imported from the svm submodule of sklearn:
  - SVC()
- Training the model on the Training Data set using:
  - fit()
- Predictions made on the Test Set using:
  - predict()

Logistic Regression

- Creating instance of Logistic Regression Model imported from the linear_model submodule of sklearn:
    - LogisticRegression()
- Training the model on the Training Data set using:
    - fit()
- Predictiions made on the Test Set using:
    - predict()

B

Decision Tree

- Creating instance of Decision Tree Model imported from the tree submodule of sklearn:
    - DecisionTreeClassifier()
- Training the model on the Training Data set using:
    - fit()
- Predictiions made on the Test Set using:
    - predict()

C

Support Vector Machine

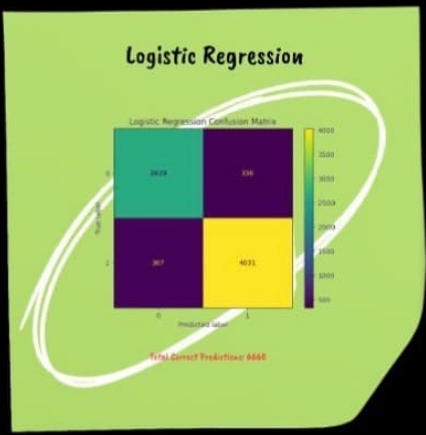
- Creating instance of Support Vector Machine Model imported from the svm submodule of sklearn:
    - SVC()
- Training the model on the Training Data set using:
    - fit()
- Predictiions made on the Test Set using:
    - predict()

A

Support Vector Machine

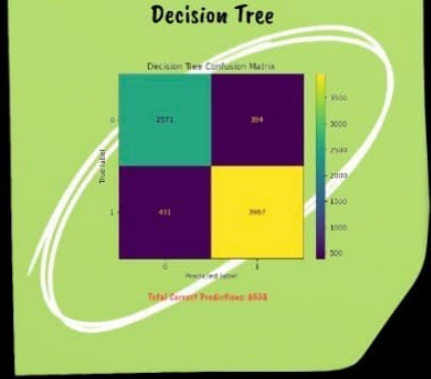
B

Logistic Regression



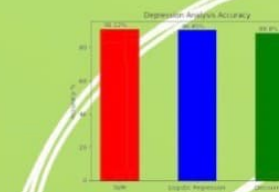
C

Decision Tree



D

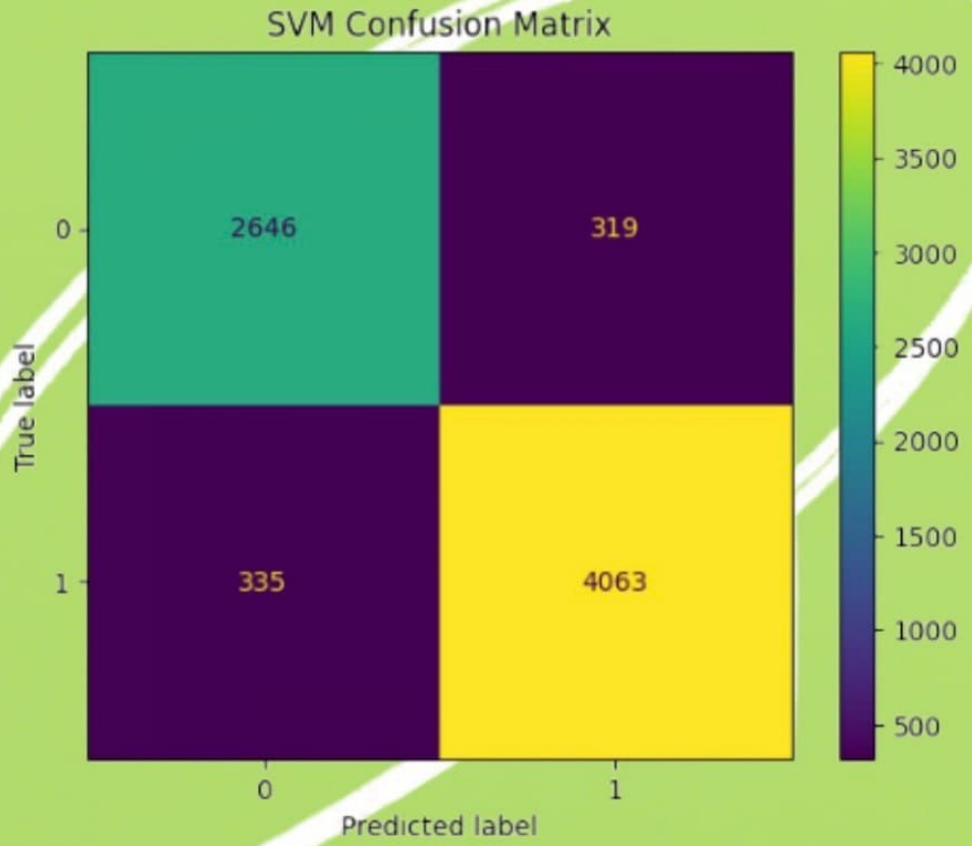
Accuracy Graph



Evaluation & Plotting

A

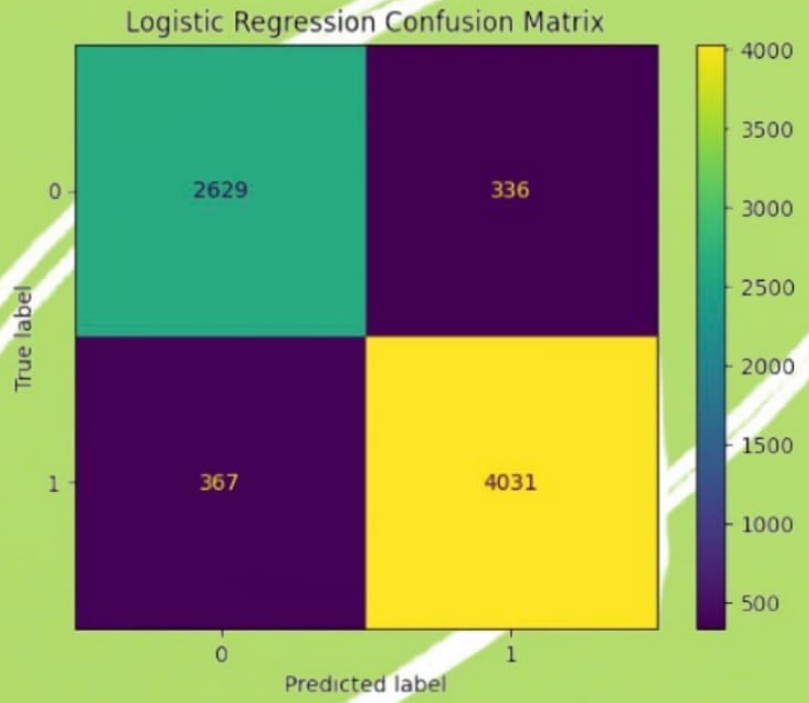
Support Vector Machine



Total Correct Predictions: 6709

B

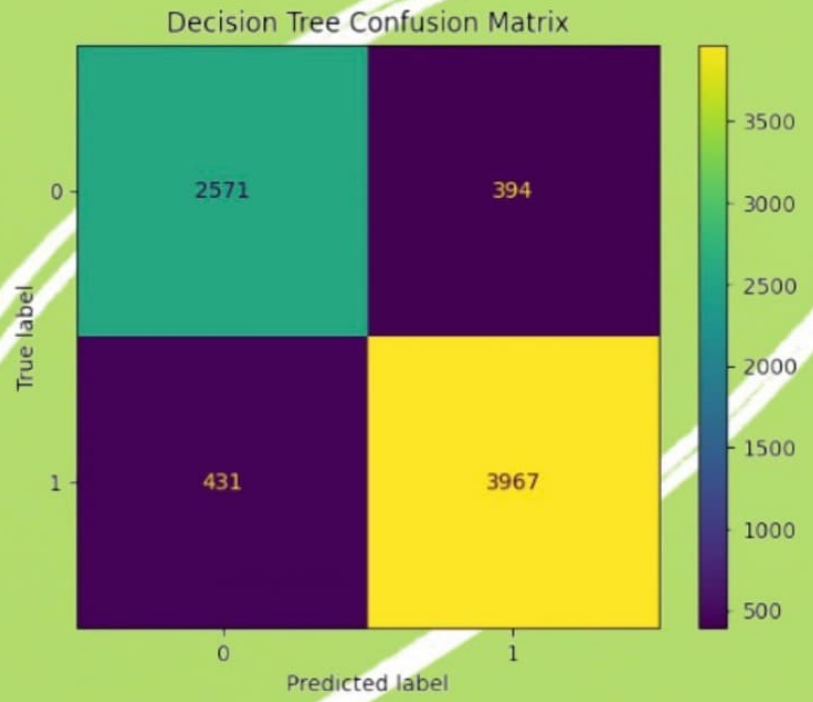
Logistic Regression



Total Correct Predictions: 6660

C

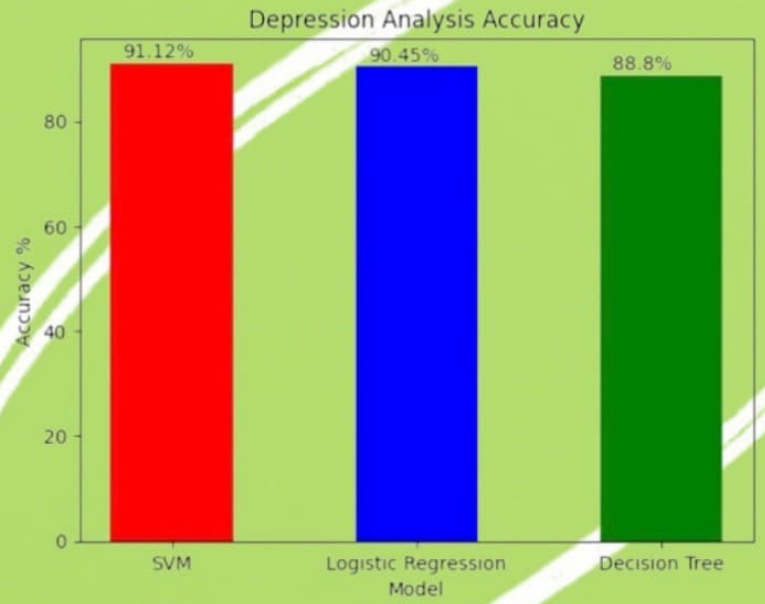
Decision Tree



Total Correct Predictions: 6538

D

Accuracy Graph



Conclusion

The objective of this project was to develop a machine learning-based model capable of detecting depression in individuals using structured data. The study involved understanding the dataset, preprocessing the data, applying various machine learning algorithms, and selecting the model that performed the best in terms of classification accuracy and interoperability.

In this project we got to learn about :

- Data Cleaning and Preprocessing
- Exploratory Data Analysis
- Model Implementation
- Model Evaluation

Best Model Selection: SVM emerged as the most reliable model with the highest accuracy and balanced metrics.

This project demonstrates that with appropriate data handling and the right choice of machine learning algorithm, it is feasible to create predictive models that can aid mental health professionals in the early detection of depression.


Thank
You!