CMSC 176 Project

Group 1: Harry Acosta, Ella Espique, Mark Olaguera

Relevant Tools

- IDE: Visual Studio Code
- Extensions: Python, Jupyter
- Github Repository Link: https://github.com/hwracosta/Group1-Capstone-Project/tree/main

Dataset

The dataset assigned to Group 1 is the Depression Dataset with the target variable "Depressed" where 1 stands for "Depressed" and 0 stands for "Not Depressed." 30 columns serve as features that will be used for classification. The following are the types of data available in the dataset:

1. **Categorical Data**: These represent non-numeric categories and have no natural order.
    - AGERNG (e.g., 16-20, 21-25)
    - GENDER (e.g., Male, Female)
    - EDU (Education level like Graduate, Post Graduate)
    - PROF (Profession like Student, Unemployed)
    - MARSTS (Marital Status: Married, Unmarried)
    - RESDPL (Residential Place: City, Town)
2. **Binary Numerical Data**: These are either "Yes" or "No" and represent binary choices often represented as 0 and 1.
    - ENVSAT (Environmental Satisfaction: Yes/No)
    - POSSAT (Positive Satisfaction: Yes/No)
    - DEBT, PHYSICAL EXERCISE (PHX), SMOKE, DRINK, ILLNESS
    - INSOM, ANXI, DEPRI, ABUSED, CHEAT, THREAT, SUICIDE
3. **Ordinal Categorical Data**: These variables represent categories with a logical order but not numerical differences.
    - AVGSLP (Average Sleep: 5 hours, 6 hours, etc.)
    - TSSN (Time Spent on Screen: 2-4 hours, 5-7 hours, etc.)
    - WRKPRE (Work Pressure: Mild, Moderate, Severe)

Learning Algorithms

For supervised learning, Naïve Bayes and Random Forest will be used on the dataset while k-means clustering will be applied for unsupervised learning.

Reasons for using Naïve Bayes:

1. Simplicity
    - It is a simple algorithm based on Bayes' Theorem, making it easy to explain and interpret. It also requires minimal tuning which reduces the complexity of the modeling process.
2. Fast and efficient
    - Since Naïve Bayes relies on calculating probabilities from the training data, it trains much faster compared to other classifiers.
    - The algorithm is computationally lightweight as it involves simple calculations of probabilities and counts. It can also handle large datasets with many features without a significant drop in performance.

Reasons for using Random Forest:

1. Effective handling of categorical features
    - The depression dataset contains categorical data with some being binary and ordinal. Random Forest works well with numerical transformations like binary encoding and ordinal encoding.
2. Non-linearity handling
    - Data on mental health often involves complex relationships between features that are not linearly separable.
    - Random Forest can capture nonlinear patterns effectively unlike linear models such as Logistic Regression.
3. Robustness to noise and outliers
    - Random Forest is less sensitive to noise and outliers due to its ensemble nature (combining multiple decision trees).
4. Overfitting control
    - The use of multiple decision trees reduces the risk of overfitting compared to a single decision tree.

Reasons for using K-Means Clustering:

1. Scalability
    - K-means clustering does not rely heavily on the number of vectors, thus it can be used on a dataset that does not have a lot of vectors.
2. Efficiency
    - It has a linear time complexity, meaning that it can handle large datasets with ease.
3. Flexibility
    - The algorithm can easily respond to changes, thus allowing for adaptability.

Naïve Bayes

Naïve Bayes is a type of classification algorithm wherein probability is used to predict an object, in this case, depression is the one being predicted. It is called as such as the features are independent of each other. In this case, the features in the dataset are not based on each other.

```
# Import necessary libraries
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import CategoricalNB
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report, precision_score, recall_score, f1_score
import matplotlib.pyplot as plt
import numpy as np

# Load the dataset
dataset_path = 'C:/Users/Cindy/Downloads/Depression Dataset.csv'
dataset = pd.read_csv(dataset_path)
```

The libraries needed for the machine learning algorithm were first imported. The dataset is read from a CSV file located at the specified path and then loaded into the program.

```python
# Check for missing values
if dataset.isnull().sum().sum() > 0:
    print("Missing values detected.")
else:
    print("No missing values detected.")

# Check for inconsistent entries
for column in dataset.columns:
    unique_values = dataset[column].unique()
    print(f"{column} - Unique Values: {unique_values}")
```

✓ 0.0s

```
No missing values detected.
AGERNG - Unique Values: ['26-30' '21-25' '16-20' '31-35' '46-50' '41-45' '56-60' '36-40' '61+'
 '51-55']
GENDER - Unique Values: ['Female' 'Male']
```

Data preprocessing is performed on the data. The data is first checked for missing values by identifying if any cells contain missing values. We then loop through each column to print unique values to identify inconsistent or noisy data. This is useful for spotting unexpected values such as negative numbers and checking if formats are consistent. In the case of the Depression dataset, there are no missing values nor inconsistent entries found.

```python
# Convert all columns to categorical data type
for column in dataset.columns:
    dataset[column] = dataset[column].astype('category')

# Separate features and target variable
X = dataset.drop(columns=['DEPRESSED'])  # Assuming 'DEPRESSED' is the target column
y = dataset['DEPRESSED']

# Convert categorical columns to numerical using category codes
X = X.apply(lambda col: col.cat.codes)
y = y.cat.codes
```

The data gets converted to categorical format. This is important since many datasets include non-numeric (qualitative) data such as gender and marital status so transforming them into the category data type enables easier manipulation of the dataset. Using .astype('category'), all columns are converted to categorical data types which makes it suitable for the CategoricalNB classifier, Naive Bayes classifier for categorical data, to be used.

```
# Separate features and target variable
X = dataset.drop(columns=['DEPRESSED'])  # Assuming 'DEPRESSED' is the target column
y = dataset['DEPRESSED']

# Convert categorical columns to numerical using category codes
X = X.apply(lambda col: col.cat.codes)
y = y.cat.codes

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

The columns for the features and the target variable are then separated. Since Naïve Bayes require numeric data for calculations, the categorical columns are converted into numeric data using category codes wherein integer values are assigned to each category:. The dataset is soon split into training and testing sets where 80% are allocated for training data and 20% for testing data.

```
# Initialize the Categorical Naive Bayes classifier
nb_classifier = CategoricalNB()

# Train the classifier
nb_classifier.fit(X_train, y_train)

# Make predictions on the test set
y_pred = nb_classifier.predict(X_test)
```

The Categorical Naive Bayes (CNB) classifier is initialized and the model is trained on the training data using the .fit() method. The model makes predictions on the test data.

```
# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
classification_rep = classification_report(y_test, y_pred)

# Display results
print(f"Accuracy: {accuracy * 100:.2f}%")
print(f"Precision: {precision:.2f}")
print(f"Recall: {recall:.2f}")
print(f"F1-Score: {f1:.2f}")
print("Confusion Matrix:")
print(conf_matrix)
print("Classification Report:")
print(classification_rep)
```

```
Accuracy: 88.43%
Precision: 0.97
Recall: 0.84
F1-Score: 0.90
Confusion Matrix:
[[42  2]
 [12 65]]
Classification Report:
              precision    recall  f1-score   support

           0       0.78      0.95      0.86        44
           1       0.97      0.84      0.90        77

    accuracy                           0.88       121
   macro avg       0.87      0.90      0.88       121
weighted avg       0.90      0.88      0.89       121
```

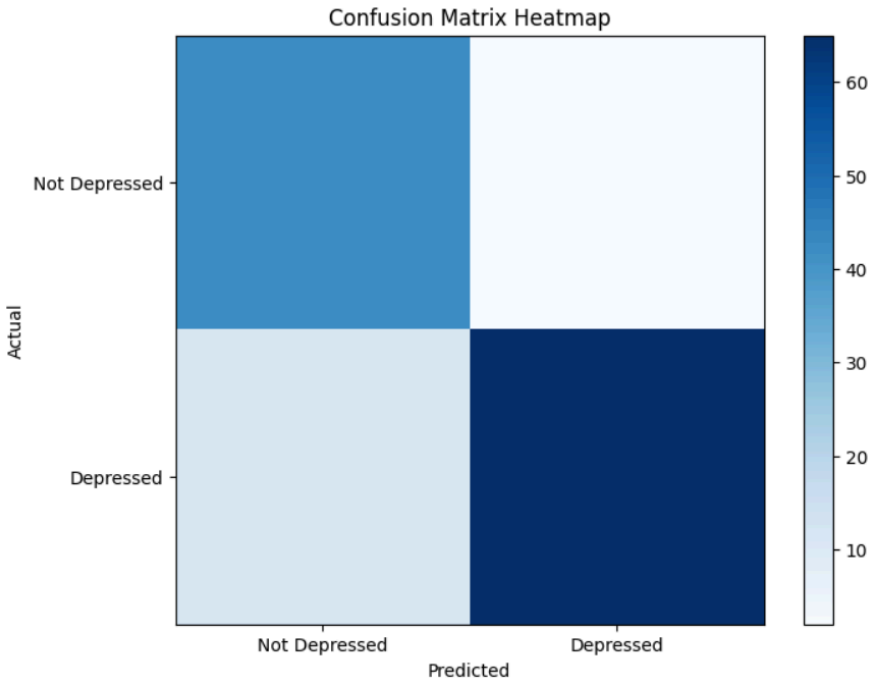
The model is evaluated and values for different measures were determined. Accuracy measures the percentage of correctly predicted labels. The result shows that the model correctly predicts cases around 88.43% of the time. Precision measures how many of the predicted positive cases were truly positive. Based on this, 97% of the cases who were predicted to be depressed are truly depressed. Recall measures how many actual positive cases were correctly predicted. 84% of all actual depressed cases were correctly identified by the model. Lastly, the F1-Score is the harmonic mean of precision and recall. Since the F1-score is 90, this suggests that there is a strong balance between precision and recall.

The confusion matrix is a matrix showing the counts of true positives, true negatives, false positives, and false negatives. 42 cases are true negatives, 65 are true positives, 2 are false positives, and 12 are false negatives.

The classification Report summarizes precision, recall, F1-score, and support. The support refers to the number of true samples belonging to a class in the dataset with 44 for not depressed and 77 for depressed.

```
# Visualize confusion matrix as a heatmap
plt.figure(figsize=(8,6))
plt.imshow(conf_matrix, interpolation='nearest', cmap='Blues')
plt.title('Confusion Matrix Heatmap')
plt.colorbar()
plt.xticks([0, 1], ['Not Depressed', 'Depressed'])
plt.yticks([0, 1], ['Not Depressed', 'Depressed'])
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```

The confusion matrix heatmap provides a graphical representation of model performance. The darker the color of a cell, the higher the frequency of cases in a cell.



The cell with the darkest color is the cell for true positives. This suggests that most of the cases in the dataset are actually depressed. The cell that is the next lesser lighter in the color is the cell for the true negatives while the cell that is the lightest in color or has the lowest frequency is the false negative.

Random Forest

Random Forest is a type of classification algorithm that uses multiple decision trees to predict an outcome. In this case, it is used to predict whether an individual is experiencing depression based on various features in the dataset. Here, multiple decision trees are created during training and their results are aggregated. This reduces the risk of overfitting compared to simply using a single decision tree.

Like in the Naïve Bayes algorithm, necessary libraries were imported, the dataset was loaded, and the columns were converted to categorical data type. However, in the Random Forest algorithm three sets of columns were defined: binary_columns, ordinal_columns, and categorical_columns.

```
# Define columns for encoding
binary_columns = ['ENVSAT', 'POSSAT', 'FINSTR', 'DEBT', 'SMOKE', 'DRINK', 'ILLNESS', 'PREMED',
                  'EATDIS', 'INSOM', 'ANXI', 'DEPRI', 'ABUSED', 'CHEAT', 'THREAT', 'SUICIDE',
                  'INFER', 'CONFLICT', 'LOST']
ordinal_columns = ['AVGSLP', 'TSSN', 'WRKPRE']
categorical_columns = ['AGERNG', 'GENDER', 'EDU', 'PROF', 'MARSTS', 'RESDPL', 'LIVWTH']
```

Binary Columns represent features with only two possible values (Yes or No). These include features like smoking, debt, and illness, where the presence or absence of a factor is represented. Ordinal Columns have a logical order but not numerical differences, such as average sleep duration and work pressure. Values like "Below 5 hours" and "More than 8 hours" have a clear order but cannot be treated as purely numeric without encoding. Categorical Columns represent categories without any specific order, such as age range, gender, and education level.

```
# Convert binary columns to numerical values
for col in binary_columns:
    dataset[col] = dataset[col].map({'Yes': 1, 'No': 0})

# Apply ordinal encoding for ordinal columns
ordinal_mappings = {
    'AVGSLP': {"Below 5 hours": 1, "5 hours": 2, "6 hours": 3, "7 hours": 4, "8 hours": 5, "More than 8 hours": 6},
    'TSSN': {"Less than 2 hours": 1, "2-4 hours a day": 2, "5-7 hours a day": 3, "8-10 hours a day": 4, "More than 10 hours": 5},
    'WRKPRE': {"No Pressure": 1, "Mild": 2, "Moderate": 3, "Severe": 4},
    'PHYEX': {"Never": 1, "Sometimes": 2, "Regularly": 3}
}

for col, mapping in ordinal_mappings.items():
    dataset[col] = dataset[col].map(mapping)

# Apply one-hot encoding for non-ordinal categorical columns
dataset = pd.get_dummies(dataset, columns=categorical_columns, drop_first=True)
```

Since machine learning algorithms require numeric input, the data will be converted into numeric type using encoding techniques. The for loop iterates first through each binary column and replaces the values "Yes" with 1 and "No" with 0. By converting the binary responses into numerical values, the model can process the data effectively.

Next, ordinal encoding is applied to columns where the categories have a meaningful order but not a numerical difference. The ordinal_mappings dictionary maps each category to a corresponding numerical value reflecting its rank or severity. Lastly, one-hot encoding was used for non-ordinal categorical columns. This treats each category as independent and prevents the algorithm from assuming ordinal relationships between categories.

```
# Initialize the Random Forest Classifier
rf_classifier = RandomForestClassifier(n_estimators=100, random_state=42)

# Train the classifier
rf_classifier.fit(X_train, y_train)

# Make predictions on the test set
y_pred = rf_classifier.predict(X_test)
```

The columns for the features and the target variable (“Depressed”) are separated once more followed by the splitting of data into training and testing sets using 80-20 split. A Random Forest classifier which is a type of ensemble learning model made up of multiple decision trees is then initialized. 100 decision trees will be made as specified while random_state=42 controls randomness and ensures that results are consistent across multiple runs. The Random Forest Classifier using the training dataset and the trained model is used to make predictions on the test dataset.

```
# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
classification_rep = classification_report(y_test, y_pred)

# Display results
print(f"Accuracy: {accuracy * 100:.2f}%")
print(f"Precision: {precision:.2f}")
print(f"Recall: {recall:.2f}")
print(f"F1-Score: {f1:.2f}")
print("Confusion Matrix:")
print(conf_matrix)
print("Classification Report:")
print(classification_rep)
```

The model was evaluated and the values for accuracy, precision, recall, and f1 score were determined. The confusion matrix and the classification report were also generated.

```
Accuracy: 86.78%
Precision: 0.87
Recall: 0.94
F1-Score: 0.90
Confusion Matrix:
[[33 11]
 [ 5 72]]
```

```
Classification Report:
              precision    recall  f1-score   support

           0       0.87      0.75      0.80        44
           1       0.87      0.94      0.90        77

    accuracy                           0.87       121
   macro avg       0.87      0.84      0.85       121
weighted avg       0.87      0.87      0.87       121
```

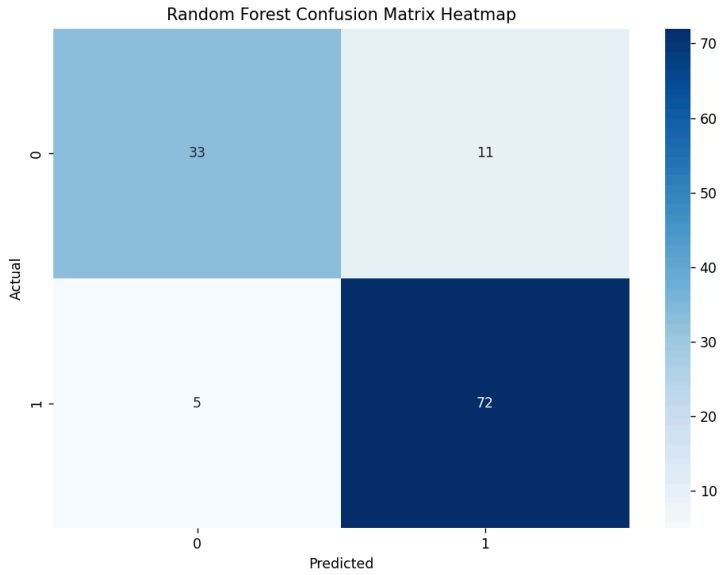
From the confusion matrix, 33 cases were correctly predicted to be not depressed while 72 cases were correctly predicted to be depressed. 11 cases were predicted depressed but are actually not depressed while 5 cases were predicted not depressed although they are actually depressed.

The accuracy of the classification table derived is approximately 86.78%. Given this, the model correctly predicts records around 86.78% of the time. The precision tells us how many of the predicted depressed cases were actually correct. The result shows that 87% of the cases predicted as depressed were indeed depressed. The recall measures the ability to capture all actual positive cases. The value of the recall is 0.94, indicating that the model correctly identified 94% of all actual depressed cases. Finally, the F1-score is the harmonic mean of Precision and Recall, balancing both metrics. An F1-Score of 0.90 signifies that there is a strong balance between precision and recall.

In the classification report, the support indicates the number of true samples belonging to this class in the dataset. 77 represents the total number of actual depressed cases in the dataset. We can also see that there is a significant difference between the recall values of the classes. The model has a higher recall for the depressed class (0.94), indicating that it captures most of the actual positive cases but might be slightly over-predicting positives (as precision is the same for both classes). On the other hand, the recall for the "Not Depressed" class (0.75) suggests the model misses some true negatives potentially due to focus on capturing positive cases.

```
# Visualize confusion matrix as a heatmap
plt.figure(figsize=(8,6))
sns.heatmap(conf_matrix, annot=True, cmap='Blues', fmt='d')
plt.title('Random Forest Confusion Matrix Heatmap')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.tight_layout()
plt.show()
```

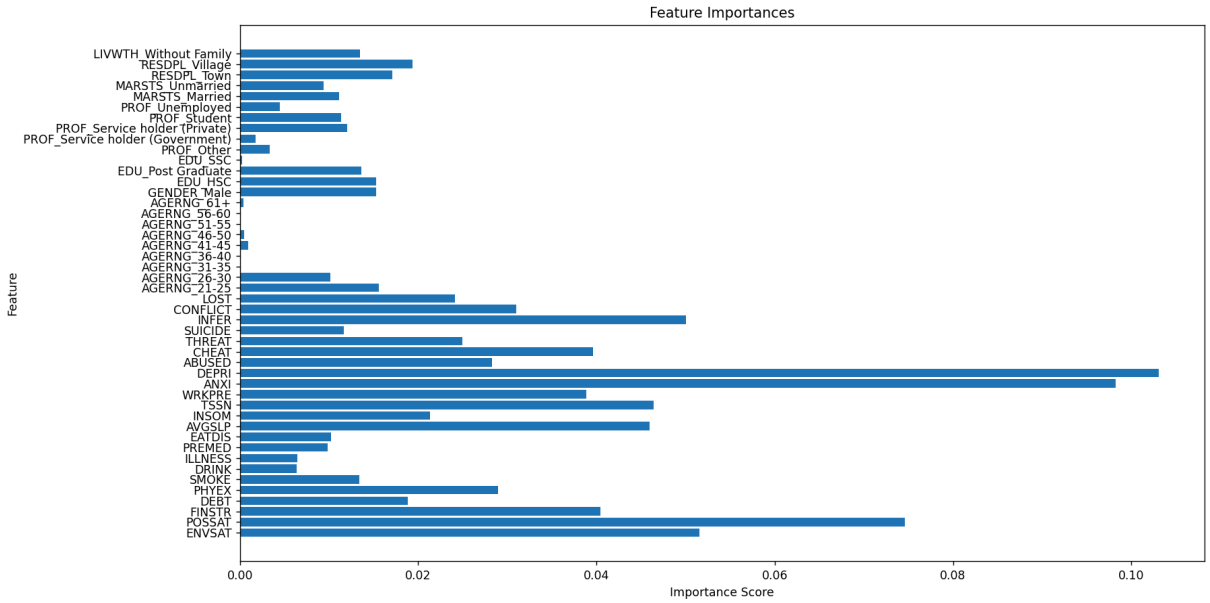
The heatmap provides a visual representation of the confusion matrix which summarizes the performance of a classification model (in this case, a Random Forest classifier). The matrix compares the predicted classes to the actual classes for both "Not Depressed" (0) and "Depressed" (1) categories.



The X-axis represents the Predicted label while the Y-axis represents the Actual labels. The color intensity of the matrix can be used to interpret the frequency of predictions in each cell. Darker shades represent higher counts of instances in that cell. The darkest cell among them would be the true positives or the 72 cases of people who are actually depressed. It also has the highest number of cases. The lightest cell would be the false negatives or the 5 cases of people who were predicted to be not depressed but are actually depressed.

```
# Feature Importance Plot
feature_importances = rf_classifier.feature_importances_
features = X.columns
plt.figure(figsize=(12,6))
plt.barh(features, feature_importances)
plt.title('Feature Importances')
plt.xlabel('Importance Score')
plt.ylabel('Feature')
plt.tight_layout()
plt.show()
```

Another graph used is a bar plot for feature importance. This illustrates the contribution of each feature to the decision-making process of the Random Forest Classifier. It helps us understand which variables had the most significant influence on predicting the target outcome.

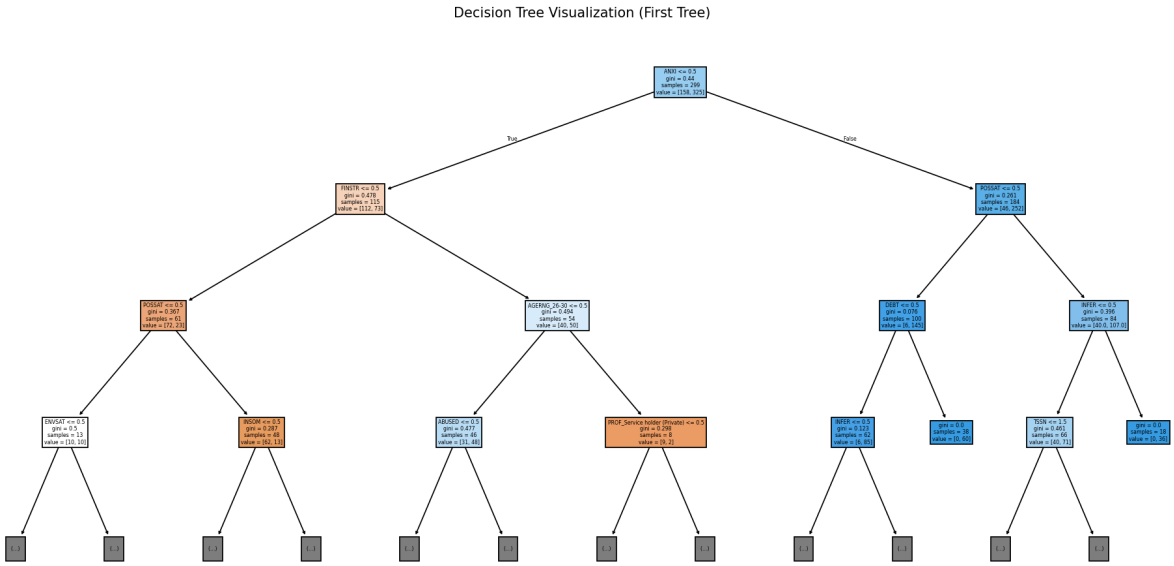


Here, a higher importance score means that the feature plays a more significant role in the prediction while a lower importance score indicates a lesser contribution to the prediction. Based on the plot, the most influential features are DEPRI (Depression Symptoms) and ANXI (Anxiety) since they have the longest bars or highest importance scores. These features contribute the most to predicting whether a person is depressed or not. Some features with moderate importance are ENVSTAT (Environmental Satisfaction) and POSSTAT (Positive Satisfaction) which suggests a link between well-being and mental health. CONFLICT and WRKPRE (Work Pressure) also have moderate scores. Some of the least influential features

are GENDER, DRINK, and LIVWTH (Living with Family). They have minimal impact on the prediction and may not be strongly correlated with the target variable in the dataset.

```
# Decision Tree Visualization (first tree in the forest)
plt.figure(figsize=(20,10))
plot_tree(rf_classifier.estimators_[0], filled=True, feature_names=X.columns, max_depth=3)
plt.title("Decision Tree Visualization (First Tree)")
plt.tight_layout()
plt.show()
```

This code is for visualizing the first tree in the random forest. A new figure with a size of 20x10 inches is initialized. rf_classifier.estimators_[0] refers to the first decision tree in the Random Forest. filled=True colors the nodes based on the class distribution for better visualization. feature_names=X.columns labels each node with the corresponding feature name while max_depth=3 limits the depth of the tree shown to 3 levels for simplicity.



This is the first tree. In each node, we can see the gini index, the number of samples in the node, and the distribution of the target variable classes. The root node is the ANXI or Anxiety class. Nodes split based on whether the feature value is below or above a certain threshold. The gray boxes at the bottom represent leaf nodes, where no further splits occur. The features near the top (like ANXI and POSSTAT) are the most important for early splits, indicating

high importance in the classification. The splits appear to be balanced since nodes with lower gini Index and skewed value distributions indicate successful splits.

Summary

The Random Forest classifier applied to the depression dataset demonstrated strong performance, as evidenced by an accuracy of 86.78%, a precision of 0.87, and a recall of 0.94. The confusion matrix highlights the model's ability to correctly identify both depressed and non-depressed individuals, with only a small proportion of false positives and false negatives. The high F1-score of 0.90 further indicates a balanced and reliable classification outcome, especially for the depressed class where recall was particularly strong.

The feature importance plot revealed that psychological and emotional factors such as depression symptoms (DEPRI) and anxiety (ANXI) were the most influential predictors in the model's decision-making process. Lifestyle-related variables, including positive satisfaction (POSSTAT) and environmental satisfaction (ENVSTAT), also played a moderate role, whereas demographic features like age range and gender contributed minimally.

The decision tree visualization provided a clear insight into the hierarchical decision-making process of the Random Forest. Key features such as anxiety and financial strain (FINSTR) were frequently used as primary split points, reinforcing their predictive significance. The tree structure showed effective classification patterns with relatively low impurity (gini index), suggesting meaningful splits in the dataset.

In conclusion, the Random Forest classifier effectively identified patterns in the depression dataset, emphasizing psychological and emotional factors as key predictors. This model provides actionable insights for mental health interventions, particularly by identifying individuals with high psychological stress and emotional distress. Further optimization, such as hyperparameter tuning, could enhance its predictive accuracy while reducing potential overfitting.

K-Means Clustering

K-Means clustering is an unsupervised machine learning algorithm used to group data points into distinct clusters based on feature similarity. In this project, we applied K-Means to a dataset with demographic, lifestyle, and psychological features to explore patterns related to depression. Unlike supervised learning methods, K-Means doesn't rely on labeled data; instead,

it identifies inherent groupings within the dataset, providing valuable insights into hidden relationships.

Step-by-Step Explanation

1. **Data Preprocessing**: The dataset was preprocessed by encoding categorical variables (e.g., gender, education level) into numeric values using LabelEncoder. Numerical features were then standardized using StandardScaler to ensure all variables had equal importance during clustering.
2. **Determining Optimal Clusters**: The Elbow Method was used to identify the optimal number of clusters. This involves plotting the "inertia" (within-cluster sum of squares) for different values of kkk and finding the "elbow" point where the rate of decrease slows. The optimal kkk, determined using KneeLocator, was 8 clusters.
3. **Clustering**: K-Means was applied with k=8k=8k=8, assigning each data point to one of the clusters. These clusters group individuals based on similarities across features such as age range, anxiety levels, and living conditions.
4. **Cluster Evaluation**: The Silhouette Score was computed to assess the quality of clustering. With a score of 0.05, the clusters showed overlap, indicating that the dataset's features might not be strongly separable. However, meaningful patterns were still identified.
5. **Cluster Characteristics**: Each cluster's key features were analyzed to understand its unique traits. For instance, clusters with higher levels of anxiety (ANXI), abuse history (ABUSED), and sleep disorders (INSOM) were more strongly associated with depression, while those with higher environmental satisfaction (ENVSAT) and strong social support (LIVWTH) had fewer depressed individuals.
6. **Comparison with Actual Labels**: A contingency table and heatmap visualized the relationship between the identified clusters and the actual depression labels. This highlighted clusters with a significant proportion of depressed individuals, providing insights into overlapping and influential factors.
7. **Visualization**: Principal Component Analysis (PCA) was used to reduce the dataset's dimensions, allowing a 2D visualization of the clusters. While the clusters showed some overlap, the visualization provided a clearer picture of the distribution of individuals based on their features.

Conclusion

The K-Means clustering revealed significant patterns in the dataset, highlighting how combinations of psychological, demographic, and lifestyle factors relate to depression. Clusters with high anxiety, poor living conditions, and histories of abuse showed a stronger association with depression. On the other hand, individuals with higher environmental satisfaction and better support systems were less likely to belong to clusters with high depression rates.

Although the Silhouette Score suggests weak separation between clusters, the analysis provides actionable insights. For instance, mental health interventions could target clusters with high psychological stress and poor support systems. This study demonstrates the potential of unsupervised learning in uncovering patterns in mental health data and guiding targeted interventions.

By leveraging clustering, we can better understand the complex interplay of factors contributing to depression, paving the way for more data-driven, individualized mental health strategies.