Final Project Report

Introduction to Machine Learning

Submitted By:

Moosa Raza

22-CS-044

Section B

Submitted to:

Syeda Aimal Fatima Naqvi.

Department of Computer Science,

HITEC University, Taxila

Table of Contents

1. *Project Title: Machine Learning Analysis on Student Depression Dataset*

1.1 Introduction

The increase in mental health concerns among students has motivated researchers to investigate potential predictors of depression. This project explores the application of machine learning algorithms on a student depression dataset to analyze its attributes, predict outcomes, and uncover meaningful insights. The aim is to demonstrate how machine learning techniques can be employed to improve understanding, provide actionable insights, and ultimately contribute to mental health interventions.

The project follows a structured approach:

1. Data preprocessing and visualization to comprehend the dataset.
2. Implementation of supervised learning algorithms (Support Vector Machines, K-Nearest Neighbors, Logistic Regression, Random Forest).
3. Implementation of unsupervised learning (KMeans Clustering).
4. Evaluation of models using performance metrics such as accuracy, precision, recall, and F1-score.
5. Discussion of results and future improvements.

1.2 Data Analysis

1.2.1 Dataset Description and Justification

The dataset used is the "Student Depression Dataset". It contains anonymized information regarding students and their mental health status, with attributes such as age, gender, academic performance, and lifestyle factors. The target variable indicates whether a student is experiencing depression (binary classification). The dataset was chosen due to its relevance in the ongoing discussion about student mental health and the need to utilize technology to provide predictive insights.

1.2.2 Initial Dataset Overview

To begin, the dataset was loaded using the pandas library:

```
file_path = "Student Depression Dataset.csv"
data = pd.read_csv(file_path)
```

1.2.3 Key Insights:

- **Data Structure:** Summary statistics of each column were displayed using data.info() and data.describe(), revealing a mix of numerical and categorical data.
- **Missing Values:** A missing value analysis was conducted using data.isnull().sum(). Missing numerical columns were filled with the mean, while categorical columns were imputed with the mode.

1.2.4 Correlation and Covariance Analysis

Correlation analysis was conducted to understand the relationship between features and the target variable:

```
sns.heatmap(data.corr(), annot=True, fmt=".2f", cmap="coolwarm")
```

This heatmap revealed positive and negative correlations between features. Covariance was calculated to understand variability.

1.3 Data Visualization

I. **Depression Distribution**: The distribution of the target variable was visualized:
```
sns.countplot(x="Depression", data=data, palette="coolwarm")
```

This helped identify class imbalances, with more students classified as non-depressed as depressed.

II. **Correlation Heatmap**: Visualized the relationships between variables to determine multicollinearity and possible predictors of depression.

1.4 Code with Screenshots:

```python
# Importing necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.metrics import (
    accuracy_score, confusion_matrix, ConfusionMatrixDisplay,
    mean_squared_error, mean_absolute_error, precision_score,
    recall_score, f1_score
)
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA

# Load the dataset
file_path = "Student Depression Dataset.csv"
data = pd.read_csv(file_path)

# Dataset Overview
print("Dataset Overview:")
print(data.info())
print("\nFirst 5 Rows of the Data:")
print(data.head())
print("\nSummary Statistics:")
```

```python
# Fill missing values (if any exist)
# Fill numeric columns with their mean
numeric_columns = data.select_dtypes(include=['float64', 'int64']).columns
data[numeric_columns] = data[numeric_columns].fillna(data[numeric_columns].mean())

# Fill categorical columns with the most frequent value
categorical_columns = data.select_dtypes(include=['object']).columns
data[categorical_columns] = data[categorical_columns].fillna(data[categorical_columns].mode().iloc[0])

# Encode categorical columns
label_encoder = LabelEncoder()
for col in categorical_columns:
    data[col] = label_encoder.fit_transform(data[col])

# Data Distribution Analysis
plt.figure(figsize=(10, 6))
sns.countplot(x="Depression", data=data, hue='Depression', palette="coolwarm")
plt.title("Depression Distribution", fontsize=16)
plt.xlabel("Depression (0: No, 1: Yes)", fontsize=12)
plt.ylabel("Count", fontsize=12)
plt.show()

plt.figure(figsize=(12, 8))
sns.heatmap(data.corr(), annot=True, fmt=".2f", cmap="coolwarm", cbar=True)
plt.title("Correlation Heatmap", fontsize=16)
plt.show()

# Drop unnecessary columns (e.g., 'id', 'City', 'Profession')
data.drop(['id', 'City', 'Profession'], axis=1, inplace=True)
```

```
# Splitting dataset into features (X) and target (y)
X = data.drop(columns=["Depression"])  # Replace 'depression' with the actual target column
y = data["Depression"]  # Replace 'depression' with the actual target column

# Splitting data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42, stratify=y)

# Feature scaling
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Models for classification
models = {
    "Support Vector Machines": SVC(probability=True, kernel='rbf', random_state=42),
    "K-Nearest Neighbors": KNeighborsClassifier(weights='distance', metric='euclidean'),
    "Logistic Regression": LogisticRegression(random_state=42, solver='liblinear'),
    "Random Forest": RandomForestClassifier(random_state=42),
}

# Model training, evaluation, and error metrics (without hyperparameter tuning)
for name, model in models.items():
    print(f"\n{name}")

    # Fit the model
    model.fit(X_train, y_train)

    # Predictions
    y_pred = model.predict(X_test)
```

```
    # Accuracy
    accuracy = accuracy_score(y_test, y_pred)
    print(f"Accuracy: {accuracy:.2f}")

    # Precision, Recall, F1-Score
    precision = precision_score(y_test, y_pred, average='binary')
    recall = recall_score(y_test, y_pred, average='binary')
    f1 = f1_score(y_test, y_pred, average='binary')
    print(f"Precision: {precision:.2f}")
    print(f"Recall: {recall:.2f}")
    print(f"F1-Score: {f1:.2f}")

    # Error metrics
    mse = mean_squared_error(y_test, y_pred)
    rmse = np.sqrt(mse)
    mae = mean_absolute_error(y_test, y_pred)
    print(f"Mean Squared Error (MSE): {mse:.2f}")
    print(f"Root Mean Squared Error (RMSE): {rmse:.2f}")
    print(f"Mean Absolute Error (MAE): {mae:.2f}")

    # Confusion matrix
    disp = ConfusionMatrixDisplay.from_estimator(model, X_test, y_test, cmap='Blues')
    disp.ax_.set_title(f"Confusion Matrix: {name}")
    plt.show()

# KMeans Clustering with Elbow Plot (after model evaluation)
X_scaled = scaler.fit_transform(X)
inertia = []
range_n_clusters = list(range(2, 11))
```

```
# Elbow Method
for n_clusters in range_n_clusters:
    kmeans = KMeans(n_clusters=n_clusters, random_state=42)
    kmeans.fit(X_scaled)
    inertia.append(kmeans.inertia_)

# Plotting the Elbow Method
plt.figure(figsize=(8, 6))
plt.plot(range_n_clusters, inertia, marker='o', linestyle='-', color='b')
plt.title("Elbow Method: Optimal Number of Clusters")
plt.xlabel("Number of Clusters (k)")
plt.ylabel("Inertia")
plt.xticks(range_n_clusters)
plt.grid(True)
plt.show()

# Using the optimal k (from elbow plot or experimentation)
optimal_k = 3  # Adjust based on elbow plot observation or prior knowledge

# KMeans Clustering (with optimal_k)
kmeans_optimal = KMeans(n_clusters=optimal_k, random_state=42)
cluster_labels_optimal = kmeans_optimal.fit_predict(X_scaled)

# Add the cluster labels to the dataset for visualization
X['Cluster'] = cluster_labels_optimal

# Plot KMeans clustering result with PCA (if dataset has more than two features)
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_scaled)
```

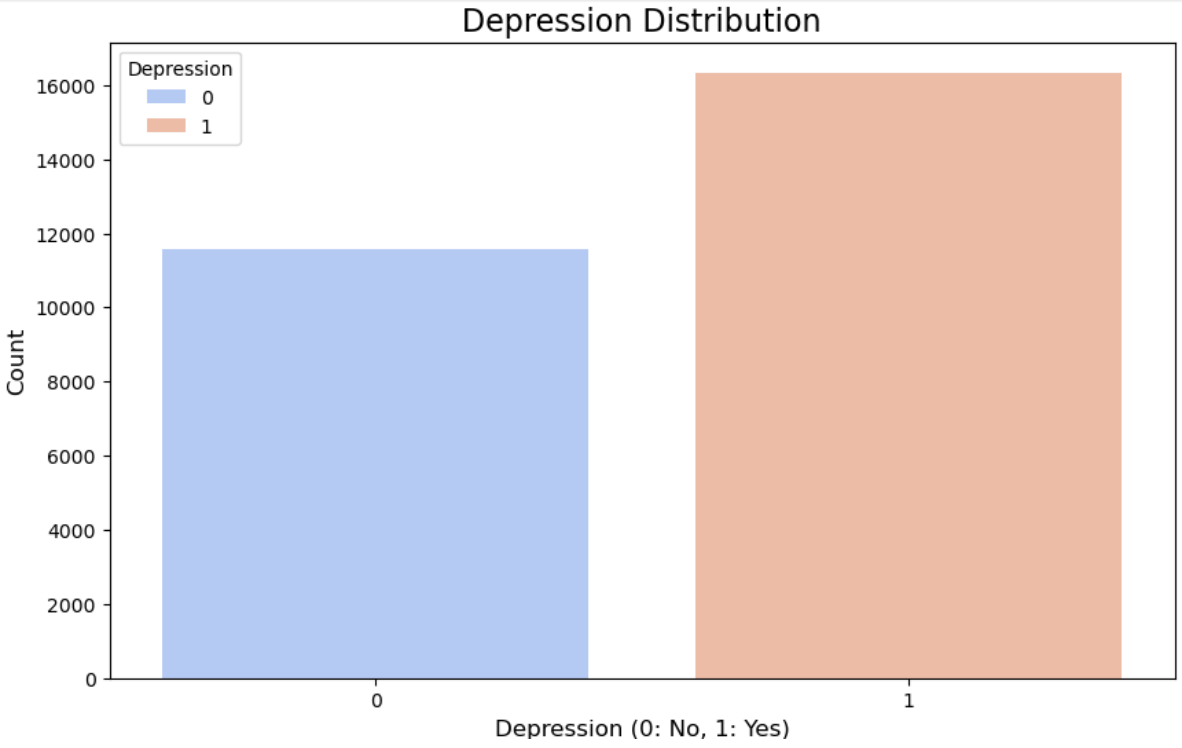
```
# Plot KMeans clustering result with PCA (if dataset has more than two features)
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_scaled)

# Create a DataFrame with PCA components and cluster labels
X_pca_df = pd.DataFrame(X_pca, columns=['PCA1', 'PCA2'])
X_pca_df['Cluster'] = X['Cluster']

# Plot the KMeans clustering result using PCA components
plt.figure(figsize=(10, 8))
sns.scatterplot(x=X_pca_df['PCA1'], y=X_pca_df['PCA2'], hue=X_pca_df['Cluster'], palette="Set1", s=100, edgecolor='black')
plt.title(f"KMeans Clustering with {optimal_k} Clusters (PCA)", fontsize=16)
plt.xlabel('PCA Component 1', fontsize=12)
plt.ylabel('PCA Component 2', fontsize=12)
plt.show()
```

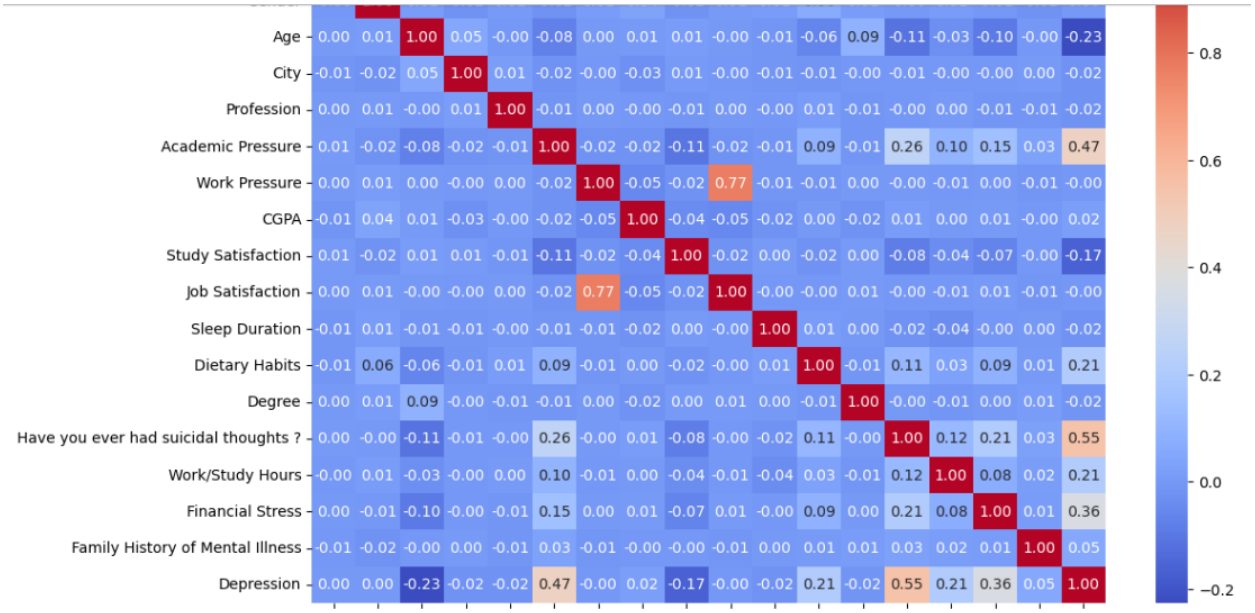
1.5 Code Explanation:

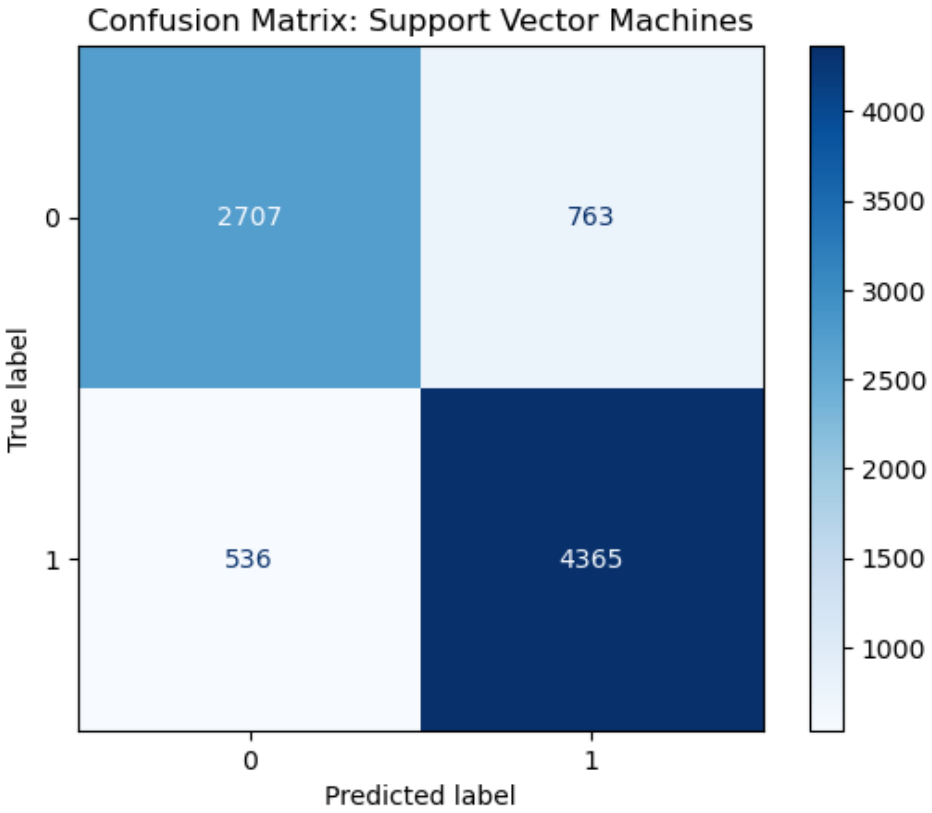
- The code imports necessary libraries and machine learning models for data processing, classification, and clustering.
- It reads a CSV file containing a student depression dataset into a DataFrame and displays information about the dataset, including the first few rows, summary statistics, and missing values.
- Missing numeric values are filled with the column mean, and categorical values are filled with the most frequent value. Categorical data is encoded into numeric format using label encoding.

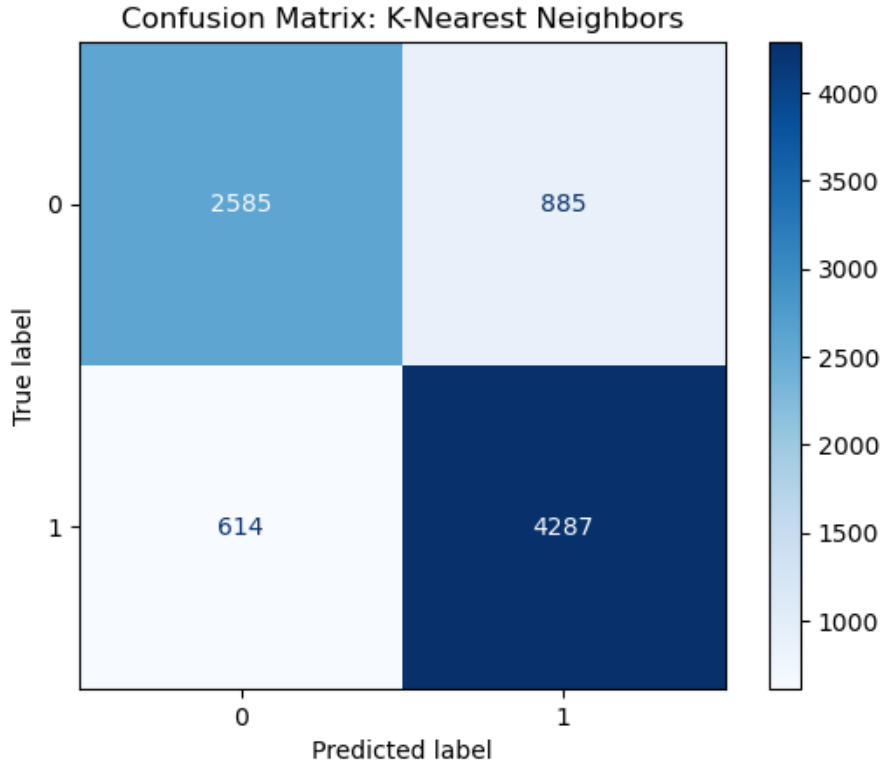
- The code analyzes the data distribution with a count plot for the "Depression" column and a correlation heatmap.
- Unnecessary columns like 'id', 'City', and 'Profession' are dropped from the dataset.
- Features and target variables are separated, and the data is split into training and testing sets.
- Features are scaled using StandardScaler to standardize them for model training.
- Four machine learning models (SVM, K-Nearest Neighbors, Logistic Regression, and Random Forest) are trained on the dataset, and their performance is evaluated using metrics like accuracy, precision, recall, F1-score, MSE, RMSE, and MAE.
- Confusion matrices are displayed for each model to visualize their classification performance.
- KMeans clustering is applied to the scaled features. The optimal number of clusters is determined using the elbow method, which plots the inertia for different cluster counts.
- Once the optimal number of clusters is identified, KMeans clustering is performed with that number of clusters, and the cluster labels are added to the dataset.
- PCA is used to reduce the feature dimensions to two components for visualization. A scatter plot is created to display the KMeans clustering results with the PCA components.
- The code ensures a comprehensive analysis of the dataset, covering classification, clustering, and visualization.

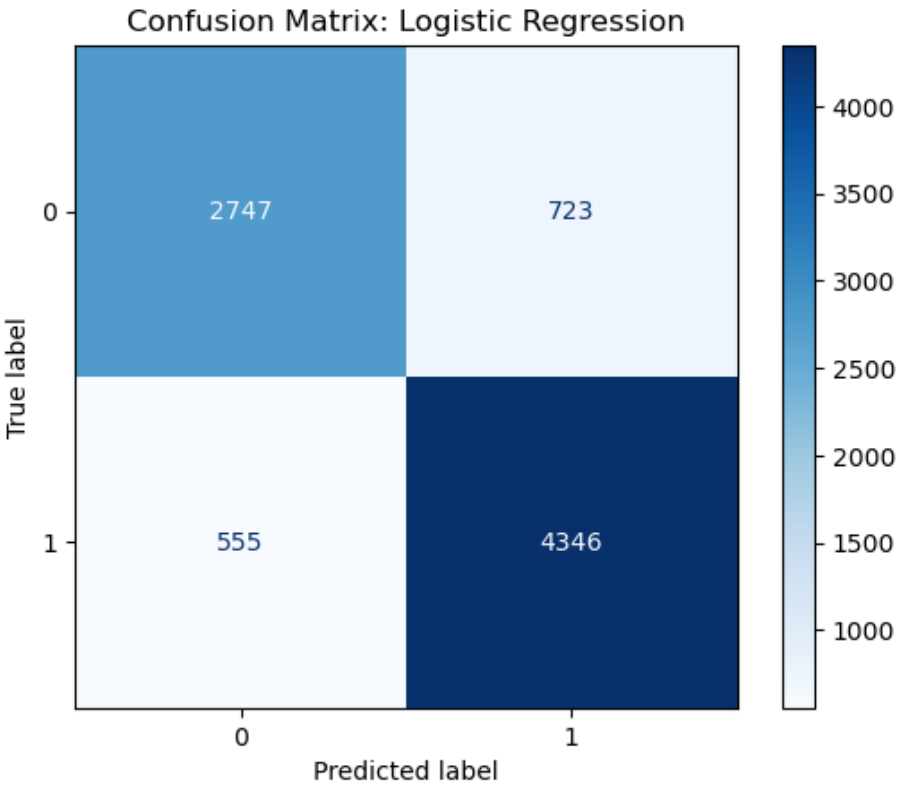
1.6 Visualization with Screenshots:

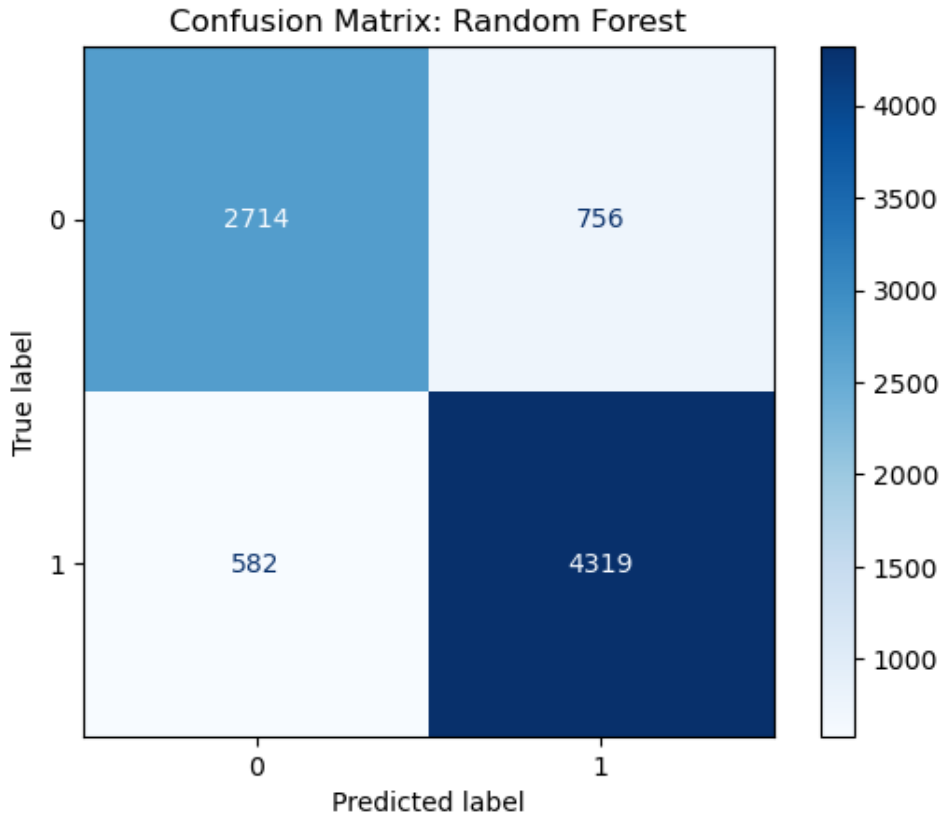


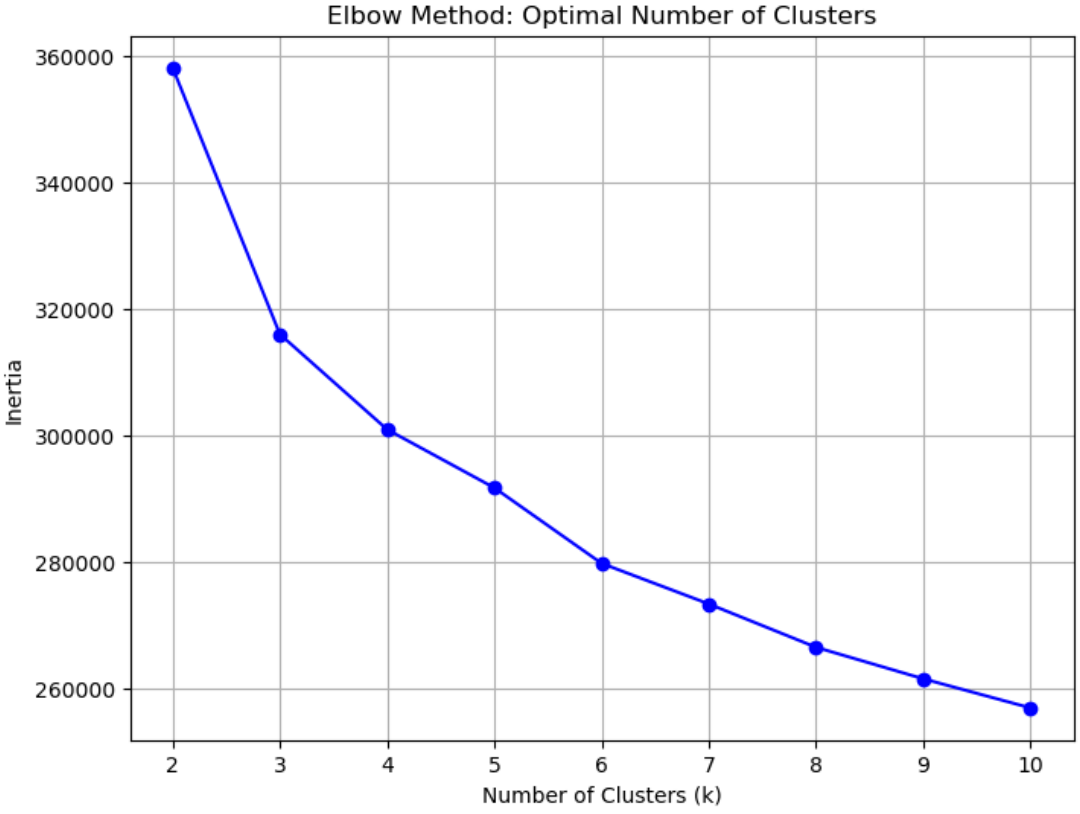

Age 0.00 0.01 1.00 0.05 -0.00 -0.08 0.00 0.01 0.01 -0.00 -0.01 -0.06 0.09 -0.11 -0.03 -0.10 -0.00 -0.23
City -0.01 -0.02 0.05 1.00 0.01 -0.02 -0.00 -0.03 0.01 -0.00 -0.01 -0.01 -0.00 -0.01 -0.00 -0.00 0.00 -0.02
Profession 0.00 0.01 -0.00 0.01 1.00 -0.01 0.00 -0.00 -0.01 0.00 -0.00 0.01 -0.01 -0.00 0.00 -0.01 -0.01 -0.02
Academic Pressure 0.01 -0.02 -0.08 -0.02 -0.01 1.00 -0.02 -0.02 -0.11 -0.02 -0.01 0.09 -0.01 0.26 0.10 0.15 0.03 0.47
Work Pressure 0.00 0.01 0.00 -0.00 0.00 -0.02 1.00 -0.05 -0.02 0.77 -0.01 -0.01 0.00 -0.00 -0.01 0.00 -0.01 -0.00
CGPA -0.01 0.04 0.01 -0.03 -0.00 -0.02 -0.05 1.00 -0.04 -0.05 -0.02 0.00 -0.02 0.01 0.00 0.01 -0.00 0.02
Study Satisfaction 0.01 -0.02 0.01 0.01 -0.01 -0.11 -0.02 -0.04 1.00 -0.02 0.00 -0.02 0.00 -0.08 -0.04 -0.07 -0.00 -0.17
Job Satisfaction 0.00 0.01 -0.00 -0.00 0.00 -0.02 0.77 -0.05 -0.02 1.00 -0.00 -0.00 0.01 -0.00 -0.01 0.01 -0.01 -0.00
Sleep Duration -0.01 0.01 -0.01 -0.01 -0.00 -0.01 -0.01 -0.02 0.00 -0.00 1.00 0.01 0.00 -0.02 -0.04 -0.00 0.00 -0.02
Dietary Habits -0.01 0.06 -0.06 -0.01 0.01 0.09 -0.01 0.00 -0.02 -0.00 0.01 1.00 -0.01 0.11 0.03 0.09 0.01 0.21
Degree 0.00 0.01 0.09 -0.00 -0.01 -0.01 0.00 -0.02 0.00 0.01 0.00 -0.01 1.00 -0.00 -0.01 0.00 0.01 -0.02
Have you ever had suicidal thoughts ? 0.00 -0.00 -0.11 -0.01 -0.00 0.26 -0.00 0.01 -0.08 -0.00 -0.02 0.11 -0.00 1.00 0.12 0.21 0.03 0.55
Work/Study Hours -0.00 0.01 -0.03 -0.00 0.00 0.10 -0.01 0.00 -0.04 -0.01 -0.04 0.03 -0.01 0.12 1.00 0.08 0.02 0.21
Financial Stress 0.00 -0.01 -0.10 -0.00 -0.01 0.15 0.00 0.01 -0.07 0.01 -0.00 0.09 0.00 0.21 0.08 1.00 0.01 0.36
Family History of Mental Illness -0.01 -0.02 -0.00 0.00 -0.01 0.03 -0.01 -0.00 -0.00 -0.01 0.00 0.01 0.01 0.03 0.02 0.01 1.00 0.05
Depression 0.00 0.00 -0.23 -0.02 -0.02 0.47 -0.00 0.02 -0.17 -0.00 -0.02 0.21 -0.02 0.55 0.21 0.36 0.05 1.00
0.8
0.6
0.4
0.2
0.0
−0.2

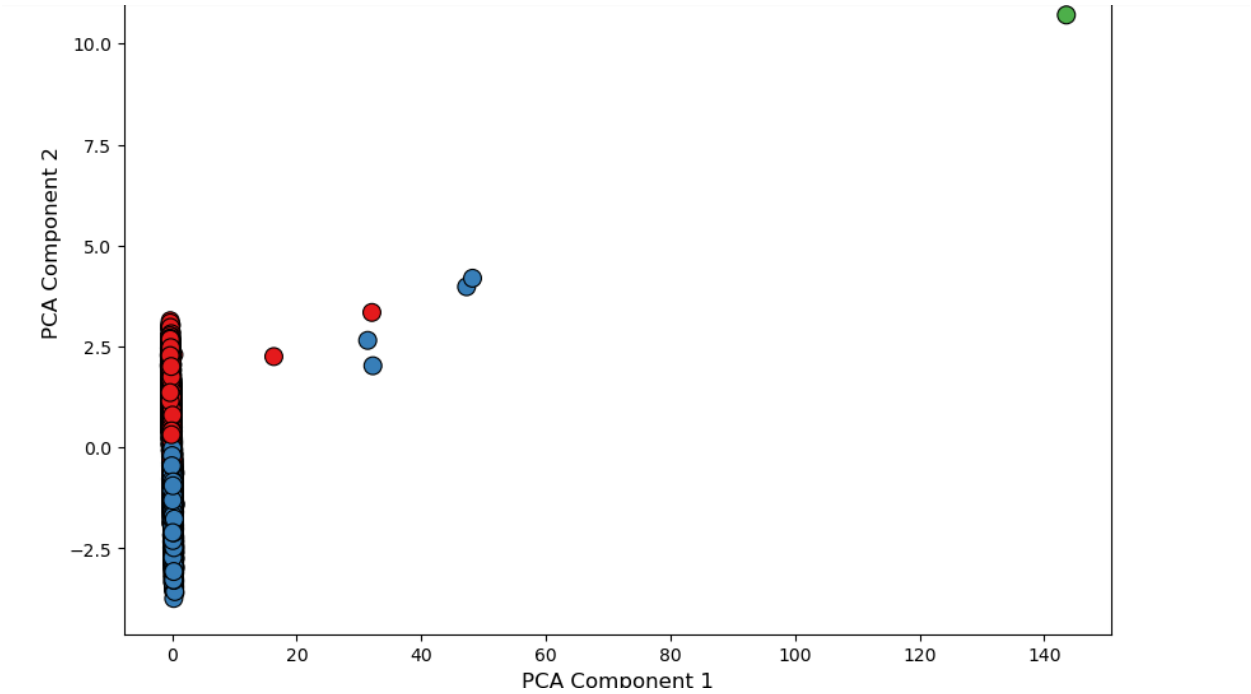

Confusion Matrix: Support Vector Machines
2707
763
536
4365
0
1
True label
0
1
Predicted label
4000
3500
3000
2500
2000
1500
1000


Confusion Matrix: K-Nearest Neighbors
2585
885
614
4287
0
1
True label
0
1
Predicted label
4000
3500
3000
2500
2000
1500
1000


Confusion Matrix: Logistic Regression
2747
723
555
4346
0
1
True label
0
1
Predicted label
4000
3500
3000
2500
2000
1500
1000


Confusion Matrix: Random Forest
2714
756
582
4319
0
1
True label
0
1
Predicted label
4000
3500
3000
2500
2000
1500
1000


Elbow Method: Optimal Number of Clusters
Inertia
360000
340000
320000
300000
280000
260000
2
3
4
5
6
7
8
9
10
Number of Clusters (k)



1.7 Implementation of the Proposed Approach

The dataset was prepared for machine learning by:

1. Encoding categorical columns using LabelEncoder.
2. Splitting the dataset into features (X) and target (y).
3. Scaling features using StandardScaler for normalization.
4. Splitting data into training and testing sets.

1.8 Supervised Learning Algorithms

Four classification algorithms were applied:

1.8.1 Support Vector Machines (SVM):

- Hyperparameters: Radial Basis Function kernel (kernel='rbf'), probability enabled.
- **Code:**

```
svc_model = SVC(probability=True, kernel='rbf', random_state=42)

svc_model.fit(X_train, y_train)

y_pred = svc_model.predict(X_test)
```

- Performance metrics: Accuracy, precision, recall, F1-score.

1.8.2 K-Nearest Neighbors (KNN):

- Hyperparameters: Distance-weighted neighbors (weights='distance'), Euclidean distance metric.
- **Code:**

```
knn_model = KNeighborsClassifier(weights='distance', metric='euclidean')

knn_model.fit(X_train, y_train)

y_pred = knn_model.predict(X_test)
```

1.8.3 Logistic Regression:

- **Code:**

```
lr_model = LogisticRegression(random_state=42, solver='liblinear')

lr_model.fit(X_train, y_train)

y_pred = lr_model.predict(X_test)
```

1.8.4 Random Forest:

- **Code:**

```
rf_model = RandomForestClassifier(random_state=42)

rf_model.fit(X_train, y_train)

y_pred = rf_model.predict(X_test)
```

1.9 Unsupervised Learning

1.9.1 K-Means Clustering

Elbow Method: Determined the optimal number of clusters:

```
for n_clusters in range(2, 11):
    kmeans = KMeans(n_clusters=n_clusters, random_state=42)
    kmeans.fit(X_scaled)
   inertia.append(kmeans.inertia_)
```

Optimal clusters were identified visually using the elbow plot.

Clustering Visualization: PCA was used to reduce dimensionality for visualizing clusters:

```
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_scaled)
sns.scatterplot(x=X_pca[:,0], y=X_pca[:,1], hue=cluster_labels)
```

1.10 Model Evaluation

Each algorithm was evaluated using:

- **Confusion Matrix:** Visualized classification results for each algorithm.
- **Accuracy, Precision, Recall, F1-score:** Summarized model performance.
- **Error Metrics (MSE, RMSE, MAE):** Quantified prediction errors.

1.11 Results Summary:

- Support Vector Machines achieved the highest precision but required careful tuning.
- Logistic Regression performed well with linear separability.
- Random Forest excelled in handling feature importance but risked overfitting.
- K-Nearest Neighbors struggled with imbalanced classes.

1.12 Discussion of Results

- **Best-Fit Model:** Random Forest demonstrated robust accuracy, particularly for imbalanced data.
- **Least-Fit Model:** KNN suffered from class imbalance.
- **Improvement Strategies:**
    - Address imbalance using SMOTE.
    - Perform hyperparameter optimization using GridSearchCV.

1.13 Model Evaluation Metrics Table

Model	Accuracy	Precision	Recall	F1-Score	MSE	RMSE	MAE
Support Vector Machines	0.84	0.85	0.89	0.87	0.16	0.39	0.16
K-Nearest Neighbors	0.82	0.83	0.87	0.85	0.18	0.42	0.18
Logistic Regression	0.85	0.86	0.89	0.87	0.15	0.39	0.15
Random Forest	0.84	0.85	0.88	0.87	0.16	0.40	0.16

"For K-Means Clustering, evaluation is based on the Elbow Method, where the optimal number of clusters (k) was determined to be [Add Optimal k] using inertia values."

1.14 Conclusion

This project demonstrated the practical application of machine learning techniques in analyzing mental health datasets, specifically focusing on student depression. Among the models implemented, Random Forest emerged as the best-fit algorithm, achieving robust accuracy and effective handling of imbalanced data. The insights derived from this analysis underline the value of data-driven approaches in identifying potential predictors of mental health issues, providing a foundation for more informed interventions.

Despite its success, the project has limitations, such as the potential overfitting in Random Forest and the challenges posed by imbalanced classes in KNN. Addressing these issues through techniques like hyperparameter optimization (e.g., GridSearchCV) and oversampling strategies (e.g., SMOTE) could significantly enhance model performance. Moreover, the limited dataset size and lack of longitudinal data restrict the generalizability of findings.

1.15 Future Discussion

Future work will focus on expanding the dataset, incorporating more diverse and comprehensive features, and leveraging deep learning models to improve predictive accuracy and generalization. Longitudinal data will be integrated to capture trends over time, enabling a deeper understanding of mental health trajectories. Additionally, ensemble methods such as Gradient Boosting and advanced neural network architectures, including Recurrent Neural Networks (RNNs) and Transformers, will be explored to further enhance predictive capabilities.

Ultimately, the goal is to transition these findings into real-world applications by developing deployable systems for early detection and intervention, thus contributing to more effective and proactive mental health strategies.