

```

# =====
# CB&DP Project – Depression Prediction
# Optimized & Visualized Results (v2.2)
# =====

# Phase 1 – Setup & Imports
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, cross_val_score
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix, ConfusionMatrixDisplay
import warnings
warnings.filterwarnings('ignore')
sns.set(style="whitegrid")

# Phase 2 – Load Dataset
file_path = "/content/drive/MyDrive/CB&DP-AI-Folder/Depression Student Dataset.csv"
data = pd.read_csv(file_path)
print("Dataset Loaded:", data.shape)

# Phase 3 – Data Cleaning
data.dropna(inplace=True)
print("Cleaned Data:", data.shape)

# Phase 4 – Auto-detect Target Column & Split Data
df = data.copy()
df.columns = [col.lower().strip() for col in df.columns]

target_col = [col for col in df.columns if "depression" in col]
if not target_col:
    raise KeyError("No column related to 'depression' found in dataset.")
target_col = target_col[0]
print(f"Target column detected: '{target_col}'")

X = df.drop(columns=[target_col])
y = df[target_col]

# Phase 5 – Encoding, Scaling & PCA
X = pd.get_dummies(X, drop_first=True)

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

pca = PCA(n_components=0.95)
X_pca = pca.fit_transform(X_scaled)
print("PCA Applied – Reduced dimensions:", X_pca.shape[1])

# Phase 6 – Train & Validate Model
X_train, X_test, y_train, y_test = train_test_split(
    X_pca, y, test_size=0.2, random_state=42
)

model = RandomForestClassifier(n_estimators=300, random_state=42)
cv_scores = cross_val_score(model, X_train, y_train, cv=5)
print(f"Cross-Validation Accuracy: {cv_scores.mean()*100:.2f}%")

model.fit(X_train, y_train)
y_pred = model.predict(X_test)

acc = accuracy_score(y_test, y_pred)
print(f"\n Final Model Accuracy: {acc*100:.2f}%")
print("\n Classification Report:\n", classification_report(y_test, y_pred))

# Phase 7 – Save Results
results = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})
results.to_csv("/content/drive/MyDrive/CB&DP-AI-Folder/Prediction_Results.csv", index=False)
print("Results saved successfully.")

# =====
# Phase 8 – Visualize Results
# =====

```

```

# Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot(cmap=plt.cm.Blues)
plt.title("Confusion Matrix")
plt.show()

# Accuracy Bar
plt.figure(figsize=(6,4))
plt.bar(['Random Forest'], [acc*100], color='skyblue')
plt.ylabel("Accuracy (%)")
plt.ylim(0, 100)
plt.title("Model Accuracy")
plt.show()

# Predicted vs Actual Comparison
plt.figure(figsize=(12,4))
plt.plot(y_test.values, label='Actual', marker='o')
plt.plot(y_pred, label='Predicted', marker='x')
plt.title("Predicted vs Actual Values")
plt.xlabel("Sample Index")
plt.ylabel("Depression Label")
plt.legend()
plt.show()
# =====
# Phase 9 – Save Results & Plots as PDF
# =====

# Save confusion matrix
disp.plot(cmap=plt.cm.Blues)
plt.title("Confusion Matrix")
plt.savefig("/content/drive/MyDrive/CB&DP-AI-Folder/Confusion_Matrix.pdf")
plt.close()

# Save predicted vs actual plot
plt.figure(figsize=(12,4))
plt.plot(y_test.values, label='Actual', marker='o')
plt.plot(y_pred, label='Predicted', marker='x')
plt.title("Predicted vs Actual Values")
plt.xlabel("Sample Index")
plt.ylabel("Depression Label")
plt.legend()
plt.savefig("/content/drive/MyDrive/CB&DP-AI-Folder/Predicted_vs_Actual.pdf")
plt.close()

# Save results dataframe as PDF via HTML (then convert to PDF)
results.to_html("/content/drive/MyDrive/CB&DP-AI-Folder/Prediction_Results.html")
print("✅ All plots and results saved as PDF/HTML in CB&DP-AI-Folder")

```


Predicted vs Actual Values

