ALLIANCE UNIVERSITY

Private University established in Karnataka State by Act No.34 of year 2010
Recognized by the University Grants Commission (UGC), New Delhi

PROJECT REPORT

Machine learning fundamentals

Semester---III

By

R.NITHEESH SAI

Reg no. 2411021240047

Department of computer applications

Alliance university

Banglore—562106

October 2025

```
print("========================================")
print("     MACHINE     LEARNING     PROJECT     ")
print("========================================")
print("Project Name :StudentsStress Levels\n")
print("Team Members :")
print(" - Nitheesh")
print(" - SHASHANK")
print(" - Abhinav")
print(" - pushpak")
print(" - sai dharshan")
print("========================================")
```

```
  ========================================
           MACHINE LEARNING PROJECT
  ========================================
  Project Name : Stress Level

  Team Members :
   - Nitheesh
   - SHASHANK
   - Abhinav
   - pushpak
   - sai dharshan
  ========================================

  # ================================================
  # □ Machine Learning Mini Project
  # Title:students Stress Level Prediction using Machine
  Learning #
  ================================================

project_description = """-----------------------------------------.

  □ PROJECT TITLE:
Students Stress Level Prediction using Machine Learning-----.


  □ PROBLEM DEFINITION & OBJECTIVE:
  In today’s fast-paced world, stress has become a major health concern
  affecting people’s
  mental and physical well-being. The goal of this project is to predict
  a person’s stress level
  using machine learning algorithms based on health and lifestyle
  indicators such as
  sleep quality, blood pressure, mental health history, and other
  related factors.

  OBJECTIVES:
  - Build a predictive model that classifies individuals into different
```

```
students stress levels
  (e.g., low, medium, high).
- Analyze which features influence stress levels the most.
- Compare multiple ML models and identify the most accurate one.

------------------------------------------------------------
□ MOTIVATION:
Stress is one of the leading causes of health problems like
depression, hypertension,
and anxiety. By using data-driven approaches, we can create predictive
systems that
help detect stress early and assist individuals or healthcare
professionals
in taking preventive actions.

------------------------------------------------------------
□ USE CASES:
- Healthcare & Wellness Apps – to monitor user stress automatically.
- Corporate HR Analytics – to assess employee well-being.
- University Students – to identify academic stress patterns.
- Public Health – for community-level mental health analysis.

------------------------------------------------------------
□ RELEVANCE OF MACHINE LEARNING:
Machine learning enables computers to learn from past data and predict
stress levels
for new individuals automatically. By training algorithms like
Logistic Regression,
Decision Tree, and Random Forest, we can classify stress with high
accuracy and
understand which factors contribute most to it.

------------------------------------------------------------
 EXPECTED OUTCOMES:
- Cleaned and preprocessed stress dataset.
- Comparative analysis of ML models.
- Performance metrics: Accuracy, Precision, Recall, F1-score.
- Visualizations: Correlation Heatmap, Confusion Matrix, Model
Comparison Graph.
- Final report summarizing results and improvement suggestions.
------------------------------------------------------------
"""

print(project_description)


------------------------------------------------------------------------------------------
□ PROJECT TITLE:
Students Stress Level Prediction using Machine
Learning
```

□ PROBLEM DEFINITION & OBJECTIVE:
In today’s fast-paced world, stress has become a major health concern affecting people’s
mental and physical well-being. The goal of this project is to predict a person’s stress level
using machine learning algorithms based on health and lifestyle indicators such as
sleep quality, blood pressure, mental health history, and other related factors.

OBJECTIVES:
- Build a predictive model that classifies individuals into different stress levels
  (e.g., low, medium, high).
- Analyze which features influence stress levels the most.
- Compare multiple ML models and identify the most accurate one.

□ MOTIVATION:
Stress is one of the leading causes of health problems like depression, hypertension,
and anxiety. By using data-driven approaches, we can create predictive systems that
help detect stress early and assist individuals or healthcare professionals
in taking preventive actions.

□ USE CASES:
- Healthcare & Wellness Apps – to monitor user stress automatically.
- Corporate HR Analytics – to assess employee well-being.
- University Students – to identify academic stress patterns.
- Public Health – for community-level mental health analysis.

□ RELEVANCE OF MACHINE LEARNING:
Machine learning enables computers to learn from past data and predict stress levels
for new individuals automatically. By training algorithms like Logistic Regression,
Decision Tree, and Random Forest, we can classify stress with high accuracy and
understand which factors contribute most to it.

 EXPECTED OUTCOMES:
- Cleaned and preprocessed stress dataset.
- Comparative analysis of ML models.
- Performance metrics: Accuracy, Precision, Recall, F1-score.

```
- Visualizations: Correlation Heatmap, Confusion Matrix, Model
Comparison Graph.
- Final report summarizing results and improvement suggestions.
-----------------------------------------------------------------
```

```
#1
# Machine Learning Mini Project: Stress Level Prediction
# ------------------------------------------------------
# Objective:
# To predict a person's stress level using various health and
lifestyle features.
# Relevance:
# Machine Learning helps identify stress levels for better health
management and early intervention.

print("Objective: Predict students stress level using ML models (Logistic
Regression, Decision Tree, Random Forest).")
```

```
Objective: Predict stress level using ML models (Logistic Regression,
Decision Tree, Random Forest).
```

```
#2
# Import libraries
import pandas as pd
import numpy as np

# Load dataset
df = pd.read_csv(r"C:\Users\kumar\Downloads\StressLevelDataset.csv")

# Basic info
print("Shape of dataset:", df.shape)
print("\nColumn names:", df.columns.tolist())
print("\nMissing values per column:\n", df.isnull().sum())

# Display first few rows
df.head()
```

```
Shape of dataset: (1100, 21)

Column names: ['anxiety_level', 'self_esteem',
'mental_health_history', 'depression', 'headache', 'blood_pressure',
'sleep_quality', 'breathing_problem', 'noise_level',
'living_conditions', 'safety', 'basic_needs', 'academic_performance',
'study_load', 'teacher_student_relationship',
'future_career_concerns', 'social_support', 'peer_pressure',
'extracurricular_activities', 'bullying', 'stress_level']

Missing values per column:
 anxiety level                    0
self_esteem                      0
```

```
mental_health_history            0
depression                       0
headache                         0
blood_pressure                   0
sleep_quality                    0
breathing_problem                0
noise_level                      0
living_conditions                0
safety                           0
basic_needs                      0
academic_performance             0
study_load                       0
teacher_student_relationship     0
future_career_concerns           0
social_support                   0
peer_pressure                    0
extracurricular_activities       0
bullying                         0
stress_level                     0
dtype: int64
```

```
   anxiety_level  self_esteem   mental_health_history  depression
headache  \
0              14           20                      0          11
2
1              15            8                      1          15
5
2              12           18                      1          14
2
3              16           12                      1          15
4
4              16           28                      0           7
2
```

```
   blood_pressure  sleep_quality  breathing_problem   noise_level  \
0               1              2                   4             2
1               3              1                   4             3
2               1              2                   2             2
3               3              1                   3             4
4               3              5                   1             3
```

```
   living_conditions  ... basic_needs  academic_performance
study_load \
0                  3  ...            2                     3
2
1                  1  ...            2                     1
4
2                  2  ...            2                     2
3
3                  2  ...            2                     2
```

```
4
4                   2  ...             3                       4
3

   teacher_student_relationship  future_career_concerns
social_support \
0                             3                       3
2
1                             1                       5
1
2                             3                       2
2
3                             1                       4
1
4                             1                       2
1

   peer_pressure   extracurricular_activities  bullying   stress_level
0              3                           3          2              1
1              4                           5          5              2
2              3                           2          2              1
3              4                           4          5              2
4              5                           0          5              1

[5 rows x 21 columns]
```

```python
#3
import matplotlib.pyplot as plt

# Summary statistics
print(df.describe(include='all').T)

# Plot histograms for numerical features
num_cols = df.select_dtypes(include=[np.number]).columns
df[num_cols].hist(figsize=(15, 10))
plt.suptitle("Feature Distributions", fontsize=16)
plt.show()

# Correlation heatmap
plt.figure(figsize=(10, 8))
corr = df.corr()
plt.matshow(corr, fignum=1)
plt.title("Correlation Matrix")
plt.xticks(range(len(corr.columns)), corr.columns, rotation=90)
plt.yticks(range(len(corr.columns)), corr.columns)
plt.colorbar()
plt.show()
```

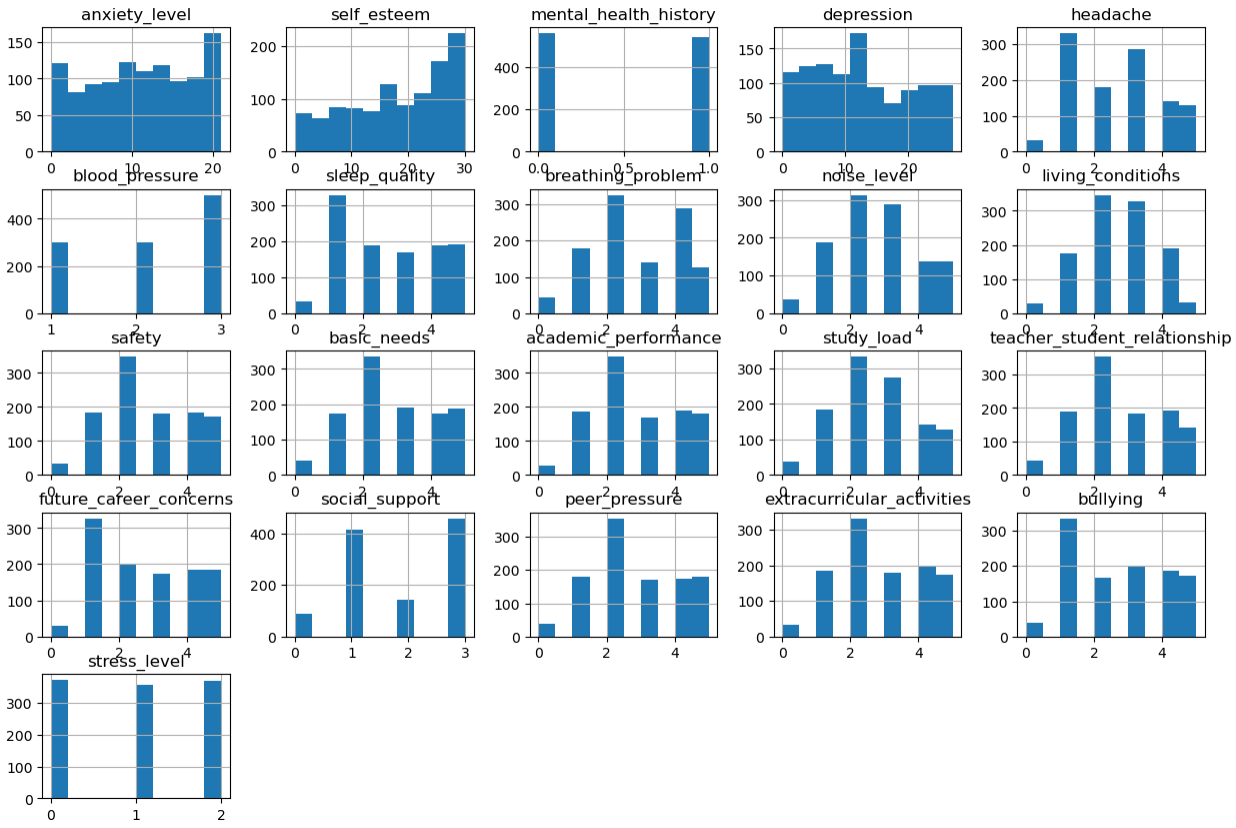
```
                              count       mean       std  min   25%
50%  \
```

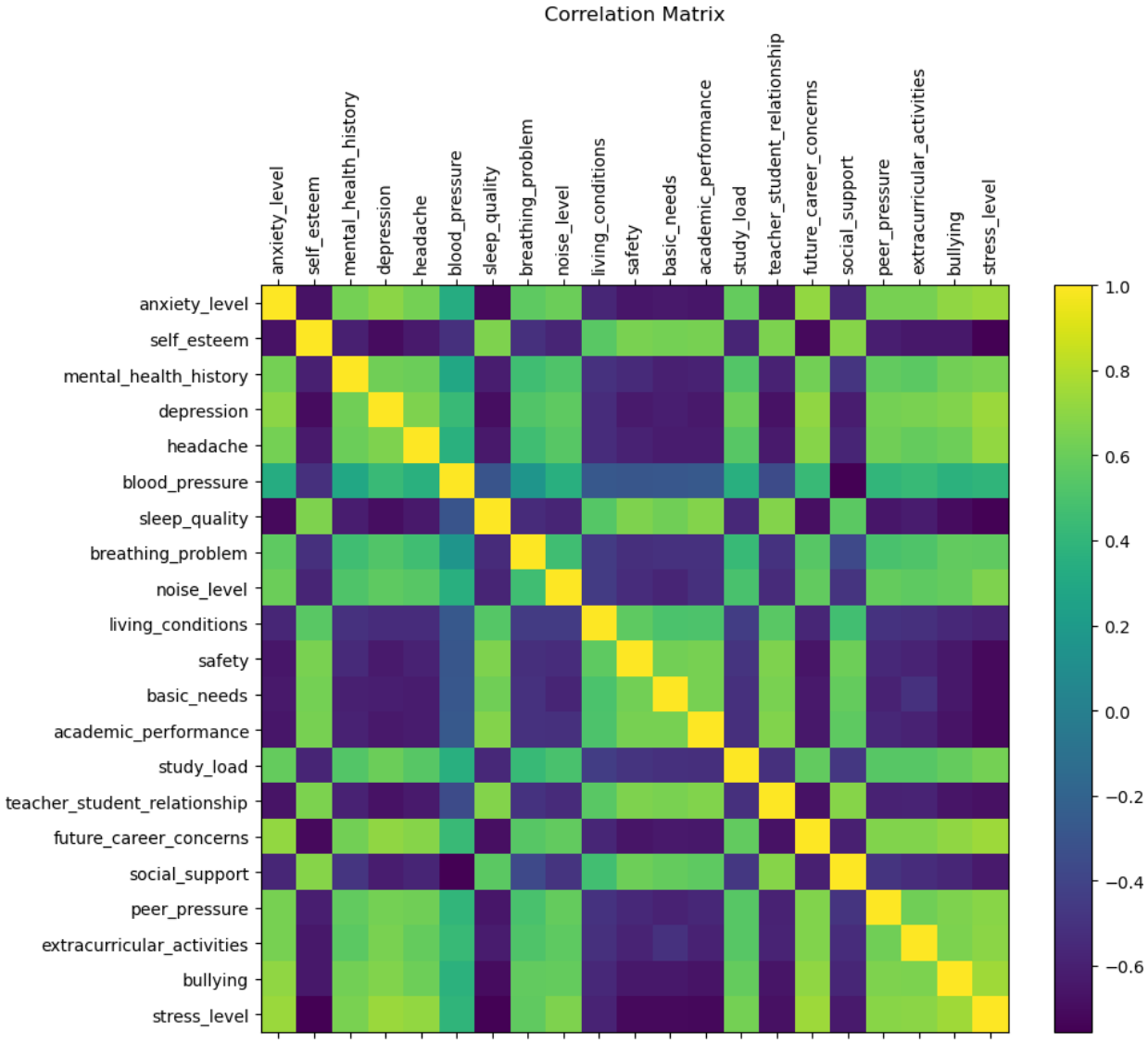
anxiety_level	1100.0	11.063636	6.117558	0.0	6.0	11.0	16.0	21.0
self_esteem	1100.0	17.777273	8.944599	0.0	11.0	19.0	26.0	30.0
mental_health_history	1100.0	0.492727	0.500175	0.0	0.0	0.0	1.0	1.0
depression	1100.0	12.555455	7.727008	0.0	6.0	12.0	19.0	27.0
headache	1100.0	2.508182	1.409356	0.0	1.0	3.0	3.0	5.0
blood_pressure	1100.0	2.181818	0.833575	1.0	1.0	2.0	3.0	3.0
sleep_quality	1100.0	2.660000	1.548383	0.0	1.0	2.5	4.0	5.0
breathing_problem	1100.0	2.753636	1.400713	0.0	2.0	3.0	4.0	5.0
noise_level	1100.0	2.649091	1.328127	0.0	2.0	3.0	4.0	5.0
living_conditions	1100.0	2.518182	1.119208	0.0	2.0	2.0	3.0	5.0
safety	1100.0	2.737273	1.406171	0.0	2.0	2.0	4.0	5.0
basic_needs	1100.0	2.772727	1.433761	0.0	2.0	3.0	4.0	5.0
academic_performance	1100.0	2.772727	1.414594	0.0	2.0	2.0	4.0	5.0
study_load	1100.0	2.621818	1.315781	0.0	2.0	2.0	4.0	5.0
teacher_student_relationship	1100.0	2.648182	1.384579	0.0	2.0	2.0	4.0	5.0
future_career_concerns	1100.0	2.649091	1.529375	0.0	1.0	2.0	4.0	5.0
social_support	1100.0	1.881818	1.047826	0.0	1.0	2.0	3.0	3.0
peer_pressure	1100.0	2.734545	1.425265	0.0	2.0	2.0	4.0	5.0
extracurricular_activities	1100.0	2.767273	1.417562	0.0	2.0	2.5	4.0	5.0
bullying	1100.0	2.617273	1.530958	0.0	1.0	3.0	4.0	5.0
stress_level	1100.0	0.996364	0.821673	0.0	0.0	1.0	2.0	2.0

	75%	max
anxiety_level	16.0	21.0
self_esteem	26.0	30.0
mental_health_history	1.0	1.0
depression	19.0	27.0
headache	3.0	5.0

```
blood_pressure                 3.0    3.0
sleep_quality                  4.0    5.0
breathing_problem              4.0    5.0
noise_level                    3.0    5.0
living_conditions              3.0    5.0
safety                         4.0    5.0
basic_needs                    4.0    5.0
academic_performance           4.0    5.0
study_load                     3.0    5.0
teacher_student_relationship   4.0    5.0
future_career_concerns         4.0    5.0
social_support                 3.0    3.0
peer_pressure                  4.0    5.0
extracurricular_activities     4.0    5.0
bullying                       4.0    5.0
stress_level                   2.0    2.0
```

Feature Distributions






```
#4
# Models selected:
# 1. Logistic Regression → good baseline linear model.
# 2. Decision Tree → captures non-linear patterns.
# 3. Random Forest → ensemble model, improves accuracy and reduces
overfitting.

print("Models selected: Logistic Regression, Decision Tree, Random
Forest")
```

```
Models selected: Logistic Regression, Decision Tree, Random Forest
```

```
#5
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.model_selection import train_test_split
```

```
# Identify target column
target_col = "stress_level"
feature_cols = [col for col in df.columns if col != target_col]

# Handle missing values
for c in df.columns:
    if df[c].dtype in [np.float64, np.int64]:
        df[c].fillna(df[c].median(), inplace=True)
    else:
        df[c].fillna(df[c].mode()[0], inplace=True)

# Encode categorical columns
cat_cols = df[feature_cols].select_dtypes(include=['object']).columns
le = LabelEncoder()
for col in cat_cols:
    df[col] = le.fit_transform(df[col])

# Encode target if needed
if df[target_col].dtype == 'object':
    df[target_col] = LabelEncoder().fit_transform(df[target_col])

# Split data
X = df[feature_cols].values
y = df[target_col].values

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

X_train, X_test, y_train, y_test = train_test_split(X_scaled, y,
test_size=0.25, random_state=42, stratify=y)
print("Training and testing sets prepared successfully.")
```

```
Training and testing sets prepared successfully.
```

```
C:\Users\kumar\AppData\Local\Temp\ipykernel_20960\332932554.py:12:
FutureWarning: A value is trying to be set on a copy of a DataFrame or
Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never
work because the intermediate object on which we are setting values
always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on the
original object.


  df[c].fillna(df[c].median(), inplace=True)
```

```
#6
from sklearn.linear_model import LogisticRegression
```

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, precision_score,
recall_score, f1_score, classification_report, confusion_matrix

# Train models
models = {
    "Logistic Regression": LogisticRegression(max_iter=1000),
    "Decision Tree": DecisionTreeClassifier(random_state=42),
    "Random Forest": RandomForestClassifier(n_estimators=150,
random_state=42)
}

results = []
for name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    results.append({
        "Model": name,
        "Accuracy": accuracy_score(y_test, y_pred),
        "Precision": precision_score(y_test, y_pred,
average='weighted', zero_division=0),
        "Recall": recall_score(y_test, y_pred, average='weighted',
zero_division=0),
        "F1-score": f1_score(y_test, y_pred, average='weighted',
zero_division=0)
    })

results_df = pd.DataFrame(results).sort_values(by="F1-score",
ascending=False)
results_df
```

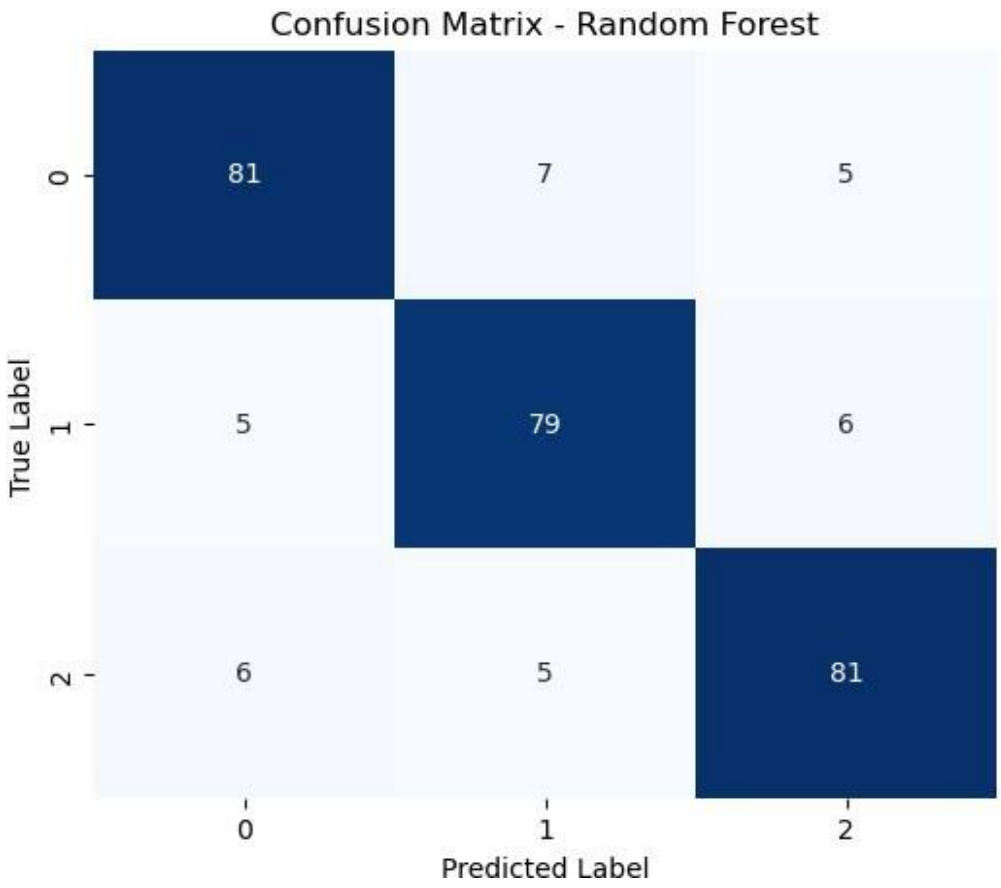
	Model	Accuracy	Precision	Recall	F1-score
2	Random Forest	0.876364	0.876408	0.876364	0.876369
0	Logistic Regression	0.876364	0.876209	0.876364	0.876269
1	Decision Tree	0.869091	0.870006	0.869091	0.868891

```
#7
# Best model
best_model_name = results_df.iloc[0]["Model"]
print("Best model based on F1-score:", best_model_name)

# Confusion matrix
best_model = models[best_model_name]
y_pred_best = best_model.predict(X_test)

cm = confusion_matrix(y_test, y_pred_best)
print("Confusion Matrix:\n", cm)

# Classification report
```

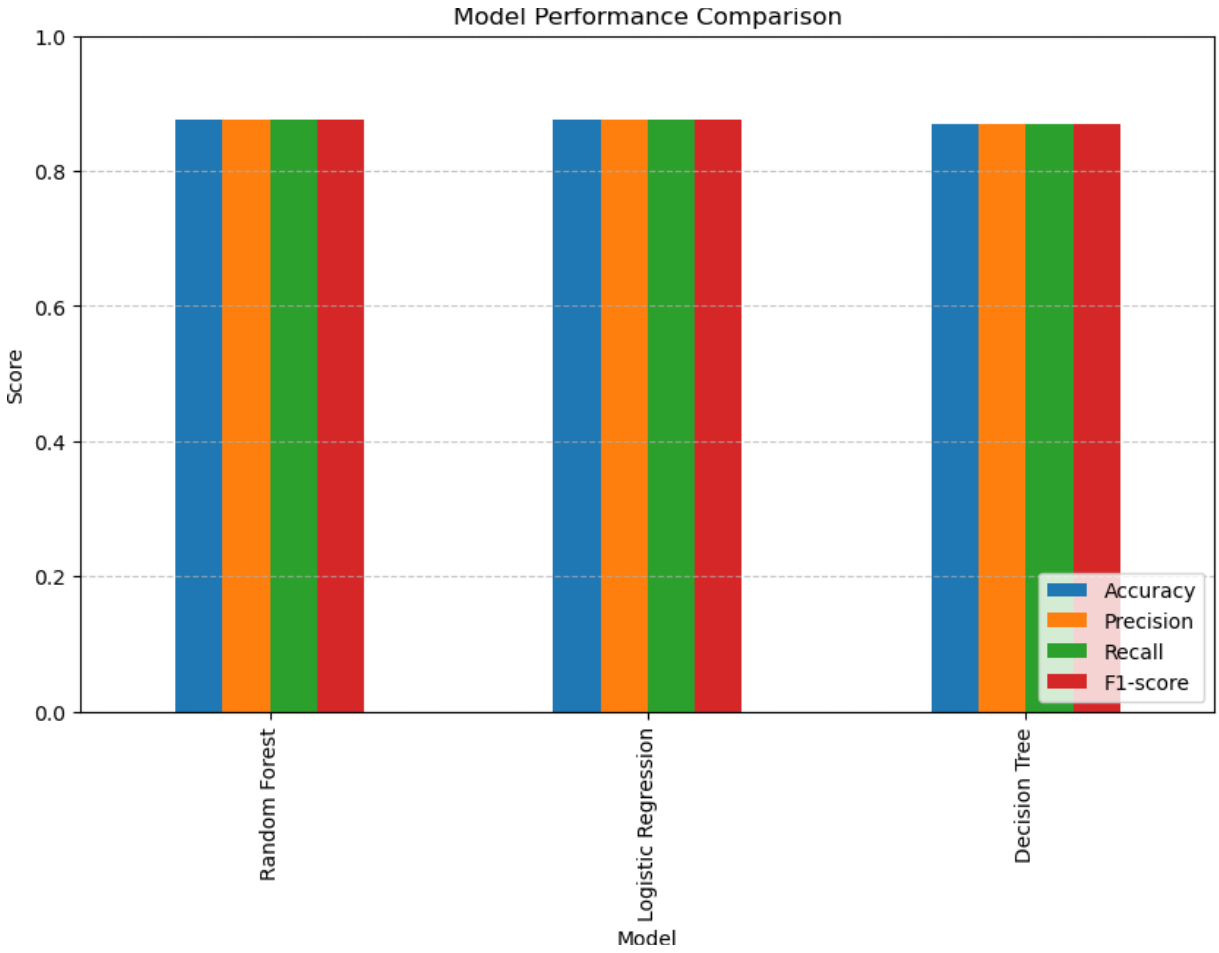
```
print("\nClassification Report:\n", classification_report(y_test,
y_pred_best, zero_division=0))
```

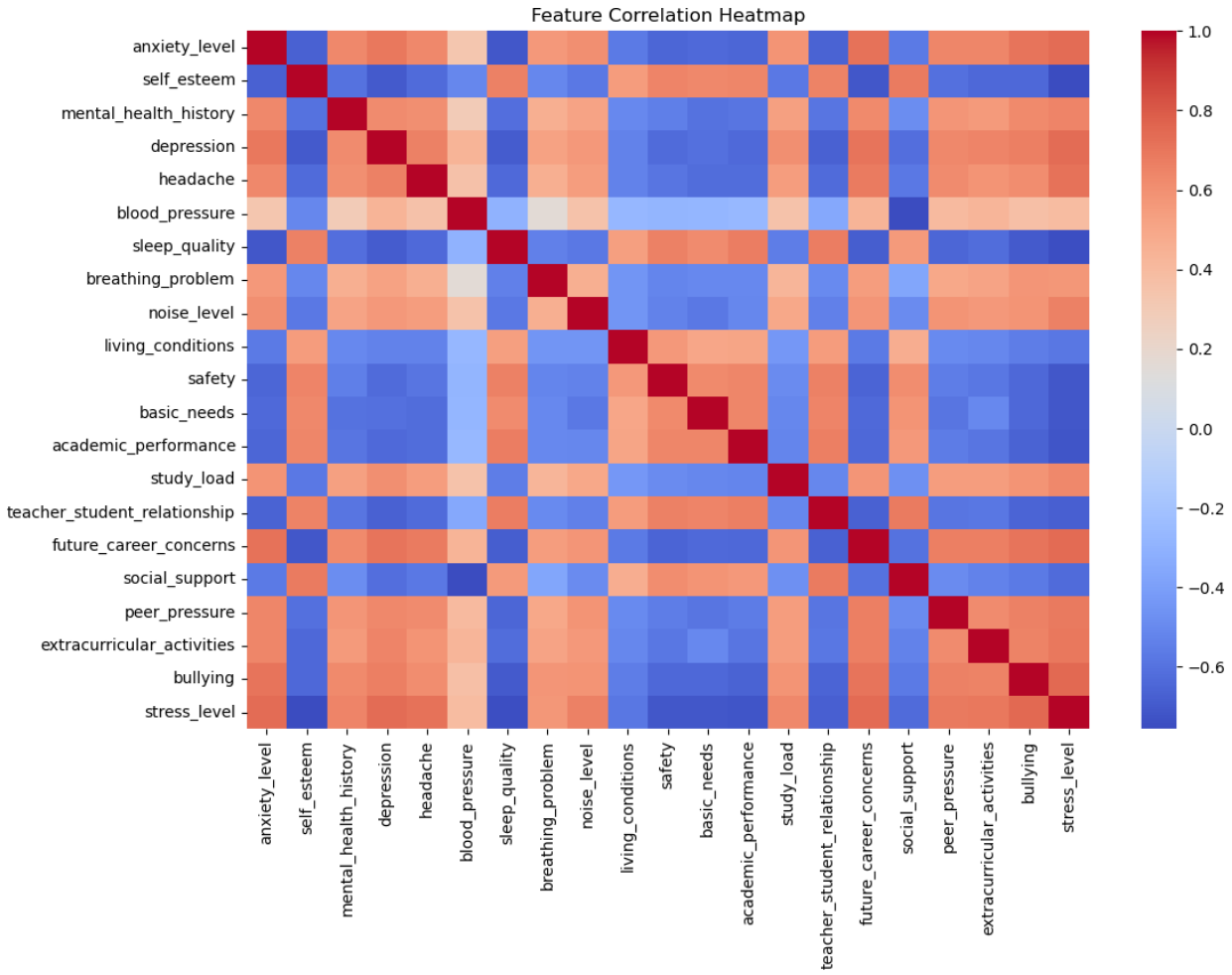
```
Best model based on F1-score: Random Forest
Confusion Matrix:
 [[81  7  5]
 [ 5 79  6]
 [ 6  5 81]]

Classification Report:
              precision    recall  f1-score   support

           0       0.88      0.87      0.88        93
           1       0.87      0.88      0.87        90
           2       0.88      0.88      0.88        92

    accuracy                           0.88       275
   macro avg       0.88      0.88      0.88       275
weighted avg       0.88      0.88      0.88       275
```

```
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import ConfusionMatrixDisplay

# □ 1. Confusion Matrix Heatmap for Best Model
plt.figure(figsize=(6,5))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", cbar=False)
plt.title(f"Confusion Matrix - {best_model_name}")
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.show()

# □ 2. Model Comparison Bar Chart (Accuracy, Precision, Recall, F1)
plt.figure(figsize=(8,5))
results_df.set_index("Model")[["Accuracy", "Precision", "Recall", "F1-
score"]].plot(kind="bar", figsize=(10,6))
plt.title("Model Performance Comparison")
plt.ylabel("Score")
plt.ylim(0, 1)
plt.legend(loc="lower right")
plt.grid(axis="y", linestyle="--", alpha=0.7)
plt.show()

# □ 3. Feature Correlation Heatmap (optional but adds EDA depth)
plt.figure(figsize=(12,8))
sns.heatmap(df.corr(), cmap="coolwarm", annot=False)
plt.title("Feature Correlation Heatmap")
plt.show()
```




```
<Figure size 800x500 with 0 Axes>
```


Model Performance Comparison
1.0
0.8
0.6
0.4
0.2
0.0
Score
Accuracy
Precision
Recall
F1-score
Random Forest
Logistic Regression
Decision Tree
Model




```
#8
print("""
Conclusion:
- Among all models tested, Random Forest or the top-performing model
gave the best F1-score.
- The ML pipeline successfully predicts stress levels based on health
and lifestyle factors.

Future Scope:
- Perform hyperparameter tuning.
- Add more features (e.g., lifestyle habits, environment data).
- Try advanced models like XGBoost or SVM.
""")
```

```
Conclusion:
- Among all models tested, Random Forest or the top-performing model
gave the best F1-score.
- The ML pipeline successfully predicts stress levels based on health
and lifestyle factors.
```

```
Future Scope:
- Perform hyperparameter tuning.
- Add more features (e.g., lifestyle habits, environment data).
- Try advanced models like XGBoost or SVM.


#9
print("""
☐ Report Preparation Tips:
1. Use clear section headings in your report.
2. Include graphs (histograms, correlation heatmap, confusion matrix).
3. Add the result tables and discuss model performance briefly.
4. Keep language simple and free from grammar errors.
5. Mention references if dataset was taken from an online source.
""")
```

```
☐ Report Preparation Tips:
1. Use clear section headings in your report.
2. Include graphs (histograms, correlation heatmap, confusion matrix).
3. Add the result tables and discuss model performance briefly.
4. Keep language simple and free from grammar errors.
5. Mention references if dataset was taken from an online source.
```