Continuous Assessment -2

NAME :- Ganesh Shankarrao Nayak

SUBJECT:- Machine Learning

ROLL.NO:- AI3019

- **Dataset Overview**

Dataset Name: StressLevelDataset.csv
Records: 1100
Features: 21 (20 input features + 1 target)
Target Variable: stress_level

- **Sample Columns:**

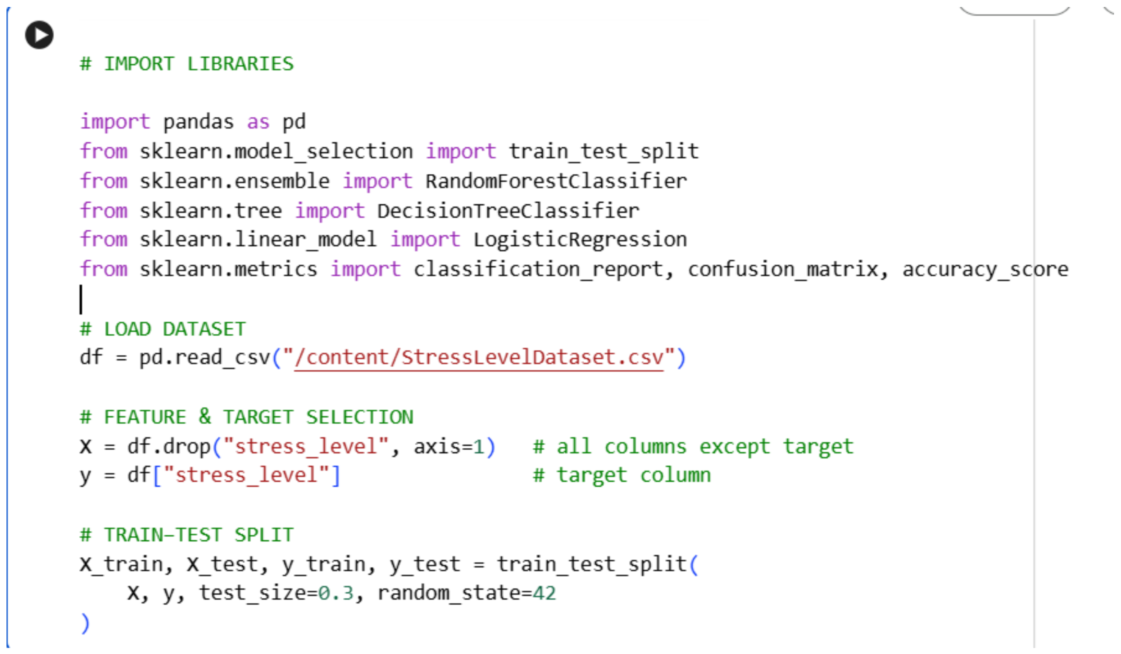
Type	Columns
Psychological	anxiety_level, depression, self_esteem
Health	blood_pressure, headache, breathing_problem
Environment	noise_level, living_conditions, safety
Academic	study_load, academic_performance
Social	peer_pressure, social_support, bullying
Target	stress_level

- **Modeling Approach**

  We train and evaluate multiple models:

  - Decision Tree
  - Random Forest
  - Logistic Regression

- **Preprocessing Section**

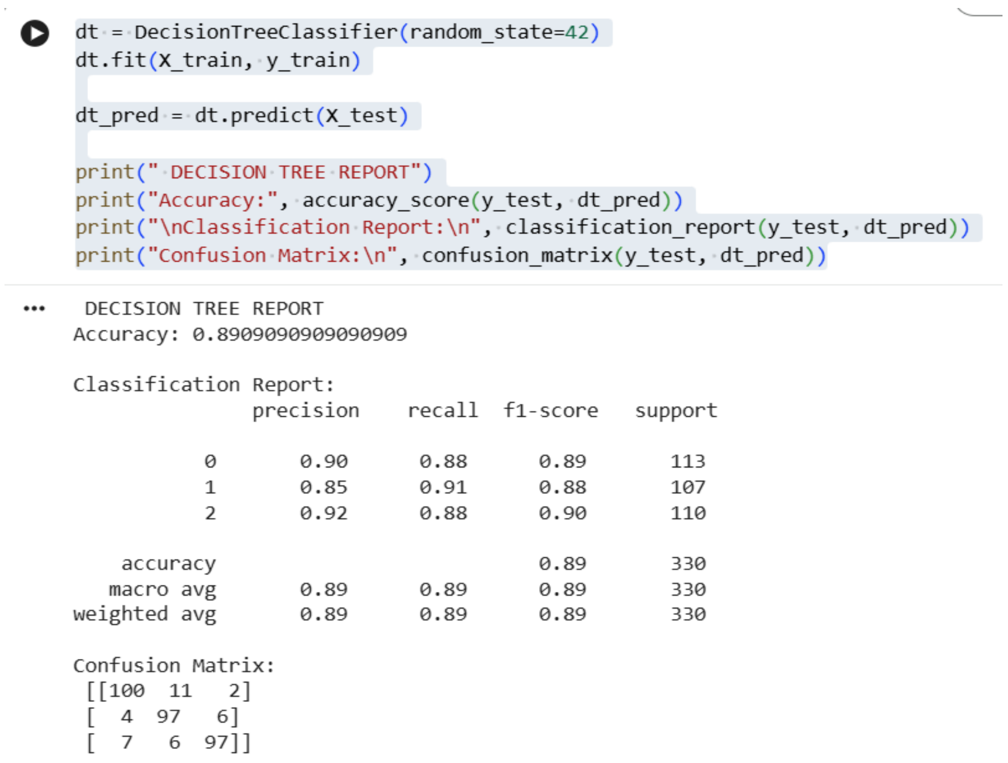


```
# IMPORT LIBRARIES

import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score

# LOAD DATASET
df = pd.read_csv("/content/StressLevelDataset.csv")

# FEATURE & TARGET SELECTION
X = df.drop("stress_level", axis=1)   # all columns except target
y = df["stress_level"]                # target column

# TRAIN-TEST SPLIT
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42
)
```

- **Summary**

  - You loaded the dataset
  - Separated features & target
  - Split data into training and testing sets
  - Prepared everything for model training

1. **Decision Tree (Short Code)**



```
dt = DecisionTreeClassifier(random_state=42)
dt.fit(X_train, y_train)

dt_pred = dt.predict(X_test)

print(" DECISION TREE REPORT")
print("Accuracy:", accuracy_score(y_test, dt_pred))
print("\nClassification Report:\n", classification_report(y_test, dt_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, dt_pred))
```

```
 DECISION TREE REPORT
Accuracy: 0.8909090909090909

Classification Report:
               precision    recall  f1-score   support

           0       0.90      0.88      0.89       113
           1       0.85      0.91      0.88       107
           2       0.92      0.88      0.90       110

    accuracy                           0.89       330
   macro avg       0.89      0.89      0.89       330
weighted avg       0.89      0.89      0.89       330

Confusion Matrix:
 [[100  11   2]
 [  4  97   6]
 [  7   6  97]]
```

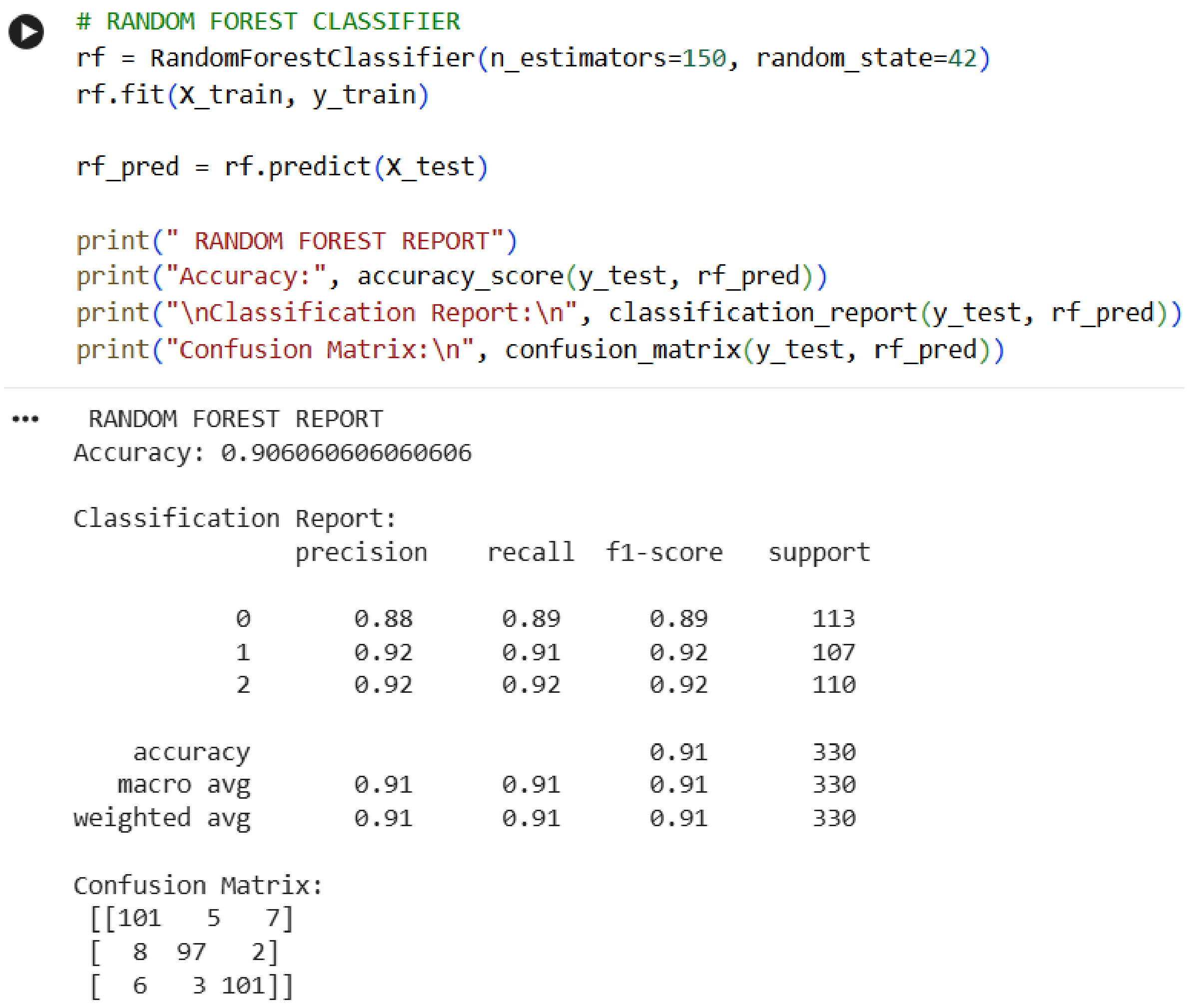
- **Summary**

The Decision Tree was trained using *dt.fit(X_train, y_train)*.

The Decision Tree model performs well with an **accuracy of 89%.**
It correctly identifies most classes with balanced precision and recall.

However, it makes some mistakes between classes that have similar patterns, which is common because decision trees may overfit.

2. Random Forest.



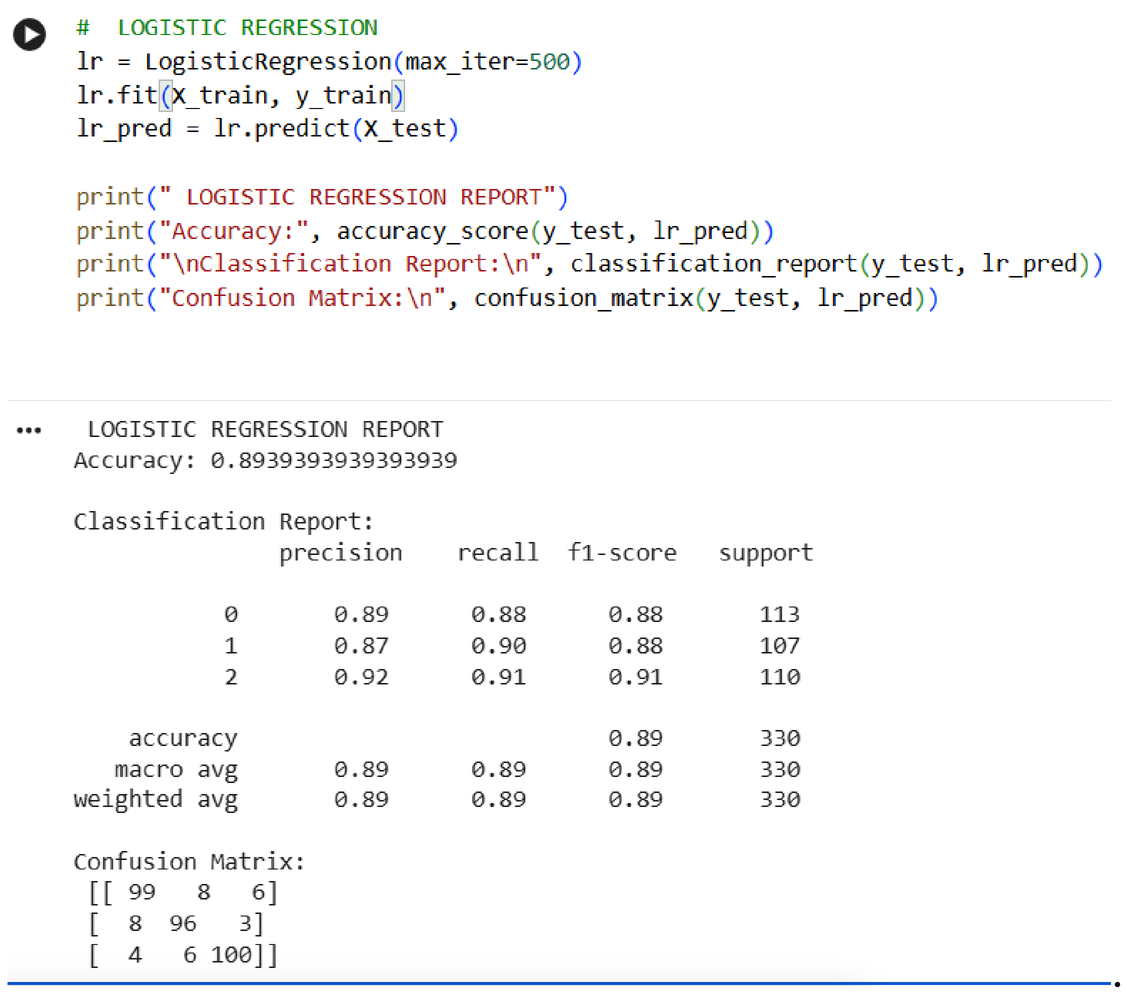
```
# RANDOM FOREST CLASSIFIER
rf = RandomForestClassifier(n_estimators=150, random_state=42)
rf.fit(X_train, y_train)

rf_pred = rf.predict(X_test)

print(" RANDOM FOREST REPORT")
print("Accuracy:", accuracy_score(y_test, rf_pred))
print("\nClassification Report:\n", classification_report(y_test, rf_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, rf_pred))
```

```
 RANDOM FOREST REPORT
Accuracy: 0.906060606060606

Classification Report:
               precision    recall  f1-score   support

           0       0.88      0.89      0.89       113
           1       0.92      0.91      0.92       107
           2       0.92      0.92      0.92       110

    accuracy                           0.91       330
   macro avg       0.91      0.91      0.91       330
weighted avg       0.91      0.91      0.91       330

Confusion Matrix:
 [[101   5   7]
 [  8  97   2]
 [  6   3 101]]
```

- **Summary**

  The Random Forest model was trained using *rf.fit(X_train, y_train)*.
  It achieved the highest **accuracy of 91%**, making it the most reliable model among the three.
  It balances precision and recall well and handles complex patterns effectively.
  Because it averages many decision trees, it avoids overfitting and provides a well-fitted, generalizable performance.

3. Logistic Regression Algorithm

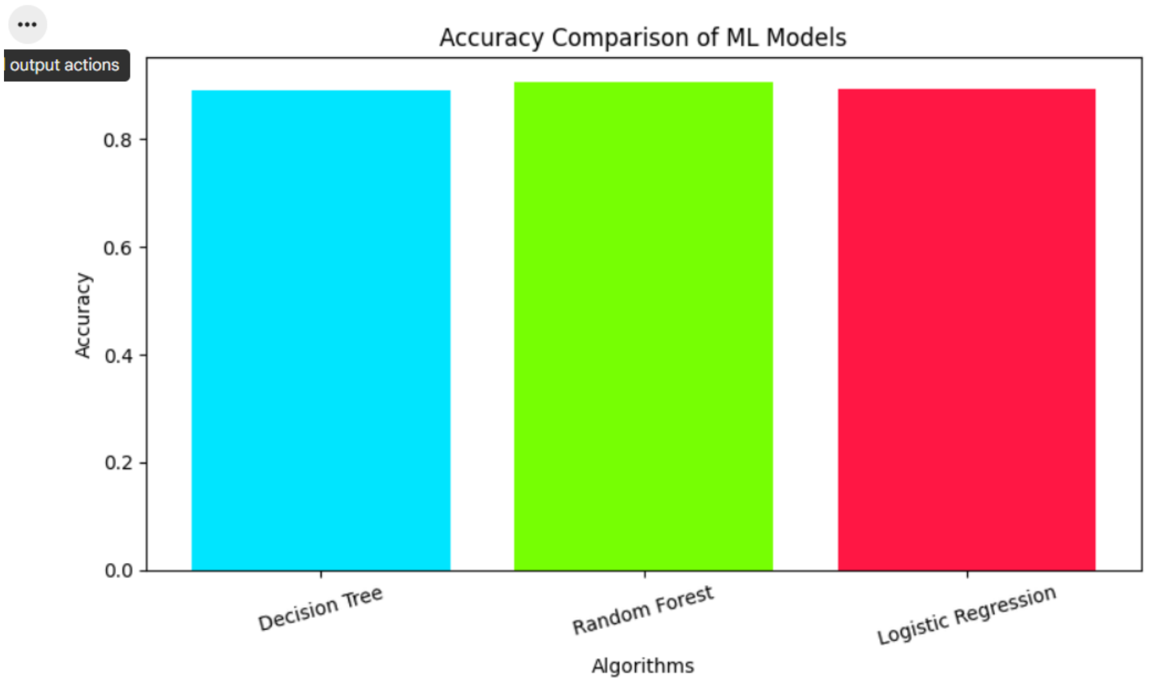


```
#  LOGISTIC REGRESSION
lr = LogisticRegression(max_iter=500)
lr.fit(X_train, y_train)
lr_pred = lr.predict(X_test)

print(" LOGISTIC REGRESSION REPORT")
print("Accuracy:", accuracy_score(y_test, lr_pred))
print("\nClassification Report:\n", classification_report(y_test, lr_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, lr_pred))
```

```
 LOGISTIC REGRESSION REPORT
Accuracy: 0.8939393939393939

Classification Report:
               precision    recall  f1-score   support

           0       0.89      0.88      0.88       113
           1       0.87      0.90      0.88       107
           2       0.92      0.91      0.91       110

    accuracy                           0.89       330
   macro avg       0.89      0.89      0.89       330
weighted avg       0.89      0.89      0.89       330

Confusion Matrix:
 [[ 99   8   6]
 [  8  96   3]
 [  4   6 100]]
```

- **Summary**

The Logistic Regression model was trained using lr.fit(X_train, y_train).
It achieved an **accuracy of 89%,** performing consistently across the dataset.
It gives stable predictions but may miss some complex relationships between features.
This happens because Logistic Regression is a simpler model, so it can underfit slightly.



- **Final Conclusion**

Based on the analysis of the stress-level dataset, all three models—Decision Tree, Logistic Regression, and Random Forest—performed well, but with different behaviors. The Decision Tree showed strong accuracy but displayed signs of overfitting, meaning it sometimes memorized patterns too closely. Logistic Regression gave stable results but slightly underfit due to its simple linear nature. The Random Forest model delivered the best overall performance, achieving the highest accuracy and providing a balanced fit without overfitting.

Overall, **Random Forest is the most reliable and well-generalized model for predicting stress levels in this dataset**.