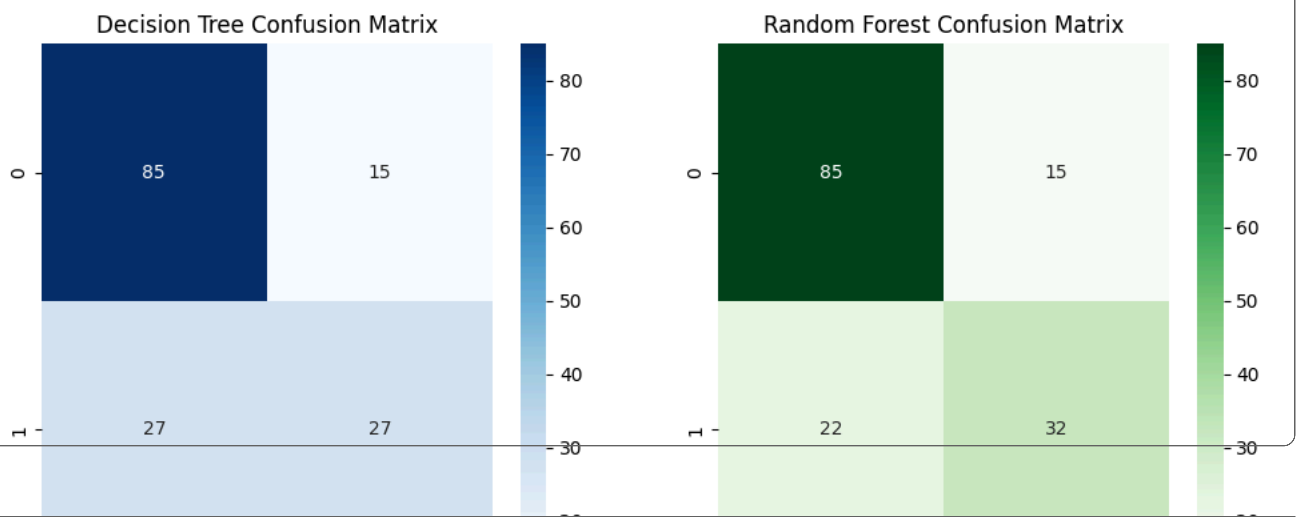
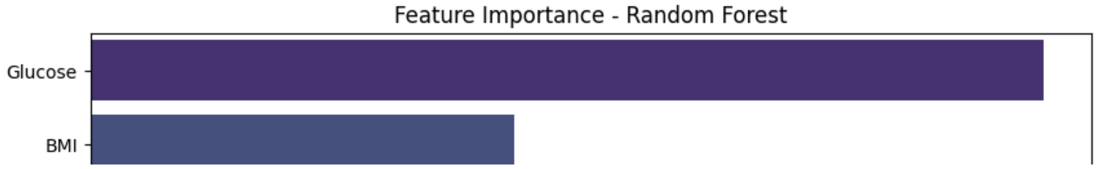
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split, GridSearchCV, cross_val_score
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier

# -------------------------------
# Step 1: Load Dataset
# -------------------------------
# PIMA Indians Diabetes dataset
url = "https://raw.githubusercontent.com/plotly/datasets/master/diabetes.csv"
data = pd.read_csv(url)

print("Dataset shape:", data.shape)
print(data.head())

# -------------------------------
# Step 2: Split into Train/Test
# -------------------------------
X = data.drop("Outcome", axis=1)
y = data["Outcome"]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)

# -------------------------------
# Step 3: Train Decision Tree
# -------------------------------
dt_model = DecisionTreeClassifier(random_state=42)
dt_model.fit(X_train, y_train)

y_pred_dt = dt_model.predict(X_test)

dt_acc = accuracy_score(y_test, y_pred_dt)
print("Decision Tree Accuracy:", dt_acc)

# -------------------------------
# Step 4: Train Random Forest
# -------------------------------
rf_model = RandomForestClassifier(random_state=42)
rf_model.fit(X_train, y_train)

y_pred_rf = rf_model.predict(X_test)

rf_acc = accuracy_score(y_test, y_pred_rf)
print("Random Forest Accuracy:", rf_acc)

# -------------------------------
# Step 5: Evaluation (Confusion Matrix & Classification Report)
# -------------------------------
print("\nDecision Tree Classification Report:\n", classification_report(y_test, y_pred_dt))
print("\nRandom Forest Classification Report:\n", classification_report(y_test, y_pred_rf))

cm_dt = confusion_matrix(y_test, y_pred_dt)
cm_rf = confusion_matrix(y_test, y_pred_rf)

fig, ax = plt.subplots(1, 2, figsize=(12, 5))
sns.heatmap(cm_dt, annot=True, fmt="d", cmap="Blues", ax=ax[0])
ax[0].set_title("Decision Tree Confusion Matrix")

sns.heatmap(cm_rf, annot=True, fmt="d", cmap="Greens", ax=ax[1])
ax[1].set_title("Random Forest Confusion Matrix")
plt.show()

# -------------------------------
# Step 6: Cross-validation
# -------------------------------
cv_dt = cross_val_score(dt_model, X, y, cv=5)
cv_rf = cross_val_score(rf_model, X, y, cv=5)
print("Decision Tree CV Mean Accuracy:", cv_dt.mean())
print("Random Forest CV Mean Accuracy:", cv_rf.mean())

# -------------------------------
# Step 7: Hyperparameter Tuning (GridSearchCV)
# -------------------------------
param_grid = {
    'n_estimators': [50, 100, 200],
    'max_depth': [None, 5, 10],
    'min_samples_split': [2, 5, 10],
    'min_samples_split': [2, 5, 10]
}

grid_rf = GridSearchCV(RandomForestClassifier(random_state=42), param_grid, cv=5, n_jobs=-1, scoring='accuracy')
grid_rf.fit(X_train, y_train)

print("Best Hyperparameters:", grid_rf.best_params_)
print("Best CV Score:", grid_rf.best_score_)

# Retrain with best params
best_rf = grid_rf.best_estimator_
y_pred_best = best_rf.predict(X_test)
print("Tuned Random Forest Accuracy:", accuracy_score(y_test, y_pred_best))

# --------------------------------
# Step 8: Feature Importance
# --------------------------------
importances = best_rf.feature_importances_
feat_names = X.columns

feat_imp = pd.Series(importances, index=feat_names).sort_values(ascending=False)

plt.figure(figsize=(10,6))
sns.barplot(x=feat_imp, y=feat_imp.index, palette="viridis")
plt.title("Feature Importance - Random Forest")
plt.show()
```

```
Dataset shape: (768, 9)
   Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0            6      148             72             35        0  33.6   
1            1       85             66             29        0  26.6   
2            8      183             64              0        0  23.3   
3            1       89             66             23       94  28.1   
4            0      137             40             35      168  43.1   

   DiabetesPedigreeFunction  Age  Outcome  
0                     0.627   50        1  
1                     0.351   31        0  
2                     0.672   32        1  
3                     0.167   21        0  
4                     2.288   33        1  
Decision Tree Accuracy: 0.7272727272727273
Random Forest Accuracy: 0.7597402597402597

Decision Tree Classification Report:
              precision    recall  f1-score   support

           0       0.76      0.85      0.80       100
           1       0.64      0.50      0.56        54

    accuracy                           0.73       154
   macro avg       0.70      0.68      0.68       154
weighted avg       0.72      0.73      0.72       154


Random Forest Classification Report:
              precision    recall  f1-score   support

           0       0.79      0.85      0.82       100
           1       0.68      0.59      0.63        54

    accuracy                           0.76       154
   macro avg       0.74      0.72      0.73       154
weighted avg       0.75      0.76      0.76       154
```



```
Decision Tree CV Mean Accuracy: 0.7163059163059163
Random Forest CV Mean Accuracy: 0.7669977081741788
Best Hyperparameters: {'max_depth': 5, 'min_samples_split': 10, 'n_estimators': 100}
Best CV Score: 0.7785685725709716
Tuned Random Forest Accuracy: 0.7207792207792207
/tmp/ipython-input-1328181292.py:105: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `

  sns.barplot(x=feat_imp, y=feat_imp.index, palette="viridis")
```


Feature Importance - Random Forest
Glucose
BMI