```
import pandas as pd
import matplotlib.pylab as plt
from sklearn.metrics import plot_confusion_matrix
from sklearn.metrics import confusion_matrix,accuracy_score
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
import os

for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
data = pd.read_csv('/Users/DELL/Desktop/AI project/heart.csv')
# Separate the features from the response
y = data["target"].copy()
X = data.drop("target", axis=1)
# Split the data into training and testing set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
# Scale features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
# Logistic Regression
l_reg = LogisticRegression(max_iter=1000, random_state=42)
l_reg.fit(X_train, y_train)
```

```
l_reg_predict_train = l_reg.predict(X_train)
l_reg_acc_score_train = accuracy_score(y_train, l_reg_predict_train)
print("Train Accuracy of Logistic Regression=", l_reg_acc_score_train, '\n')
l_reg_predict_test = l_reg.predict(X_test)
l_reg_acc_score_test = accuracy_score(y_test, l_reg_predict_test)
print("Test Accuracy of Logistic Regression=", l_reg_acc_score_test, '\n')
l_reg_conf_matrix = confusion_matrix(y_test, l_reg_predict_test)
print("confusion matrix")
print(l_reg_conf_matrix,'\n')
# Support Vector Machine(SVM)
svm_clf = SVC(kernel='poly',degree=3,coef0=2, C=8)
svm_clf.fit(X_train, y_train)
svm_predicted_train = svm_clf.predict(X_train)
svm_acc_score_train = accuracy_score(y_train, svm_predicted_train)
print("Train Accuracy of Support Vector Classifier=",svm_acc_score_train,'\n')
svm_predicted_test = svm_clf.predict(X_test)
svm_acc_score_test = accuracy_score(y_test, svm_predicted_test)
print("Test Accuracy of Support Vector Classifier=",svm_acc_score_test,'\n')
svm_conf_matrix = confusion_matrix(y_test, svm_predicted_test)
print("confusion matrix")
print(svm_conf_matrix,'\n')
# Random Forest
rf = RandomForestClassifier(n_estimators=39,max_depth=9,random_state=42)
rf.fit(X_train,y_train)
rf_predicted_train = rf.predict(X_train)
rf_acc_score_train = accuracy_score(y_train, rf_predicted_train)
print("Train Accuracy of Random Forest=",rf_acc_score_train,'\n')
rf_predicted_test = rf.predict(X_test)
rf_acc_score_test = accuracy_score(y_test, rf_predicted_test)
```

```
print("Test Accuracy of Random Forest=",rf_acc_score_test,'\n')
rf_conf_matrix = confusion_matrix(y_test, rf_predicted_test)
print("confusion matrix")
print(rf_conf_matrix,'\n')
# K-Nearest Neighbors
knn = KNeighborsClassifier(n_neighbors=1)
knn.fit(X_train, y_train)
knn_predicted_train = knn.predict(X_train)
knn_acc_score_train = accuracy_score(y_train, knn_predicted_train)
print("Train Accuracy of K-NeighborsClassifier=",knn_acc_score_train,'\n')
knn_predicted_test = knn.predict(X_test)
knn_acc_score_test = accuracy_score(y_test, knn_predicted_test)
print("Test Accuracy of K-NeighborsClassifier=",knn_acc_score_test,'\n')
knn_conf_matrix = confusion_matrix(y_test, knn_predicted_test)
print("confusion matrix")
print(knn_conf_matrix,'\n')
# Decision Tree
dt = DecisionTreeClassifier(max_depth=10,min_samples_leaf=1,min_samples_split=2,random_state=42)
dt.fit(X_train, y_train)
dt_predicted_train = dt.predict(X_train)
dt_acc_score_train = accuracy_score(y_train, dt_predicted_train)
print("Train Accuracy of DecisionTreeClassifier=",dt_acc_score_train,'\n')
dt_predicted_test = dt.predict(X_test)
dt_acc_score_test = accuracy_score(y_test, dt_predicted_test)
print("Test Accuracy of DecisionTreeClassifier=",dt_acc_score_test,'\n')
dt_conf_matrix = confusion_matrix(y_test, dt_predicted_test)
print("confusion matrix")
print(dt_conf_matrix,'\n')
# Plot Logistic Regression Confusion Matrix
```

```
plot_confusion_matrix(l_reg, X_test, y_test, cmap='Blues')
plt.title('Logistic Regression Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.xticks([0,1],['Heart disease','No heart disease'])
plt.yticks([0,1],['Heart disease','No heart disease'],rotation=90)
plt.show()
# Plot Support Vector Machine(SVM) Confusion Matrix
plot_confusion_matrix(svm_clf, X_test, y_test, cmap='Blues')
plt.title('Support Vector Machine Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.xticks([0,1],['Heart disease','No heart disease'])
plt.yticks([0,1],['Heart disease','No heart disease'],rotation=90)
plt.show()
# Plot Random Forest Confusion Matrix
plot_confusion_matrix(rf, X_test, y_test, cmap='Blues')
plt.title('Random Forest Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.xticks([0,1],['Heart disease','No heart disease'])
plt.yticks([0,1],['Heart disease','No heart disease'],rotation=90)
plt.show()
# Plot K-Nearest Neighbors Confusion Matrix
plot_confusion_matrix(knn, X_test, y_test, cmap='Blues')
plt.title('K-Nearest Neighbors Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.xticks([0,1],['Heart disease','No heart disease'])
```

```
plt.yticks([0,1],['Heart disease','No heart disease'],rotation=90)
plt.show()
# Plot Decision Tree Confusion Matrix
plot_confusion_matrix(dt, X_test, y_test, cmap='Blues')
plt.title('Decision Tree Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.xticks([0,1],['Heart disease','No heart disease'])
plt.yticks([0,1],['Heart disease','No heart disease'],rotation=90)
plt.show()
# Bar Plot for Accuracy
fig = plt.figure(figsize=(8, 6))
Algorithm = ['LR', 'SVM', 'RF', 'KNN', 'DT']
accuracy = [79.5,98.5,100,98.5,98.5]
plt.bar(Algorithm,accuracy,color='b', width=0.5)
plt.ylabel('Accuracy (percent)')
plt.xlabel('Algorithm')
plt.title('Accuracy comparison')
plt.show()
```