SOURCE CODE

```
# Importing essential libraries import numpy as np
import pandas as pd from sklearn.model_selection
import train_test_split
# Loading the dataset df=pd.read_csv('heart.csv')
# Returns number of rows and columns of the dataset df.shape
# Returns an object with all of the column headers df.columns
# Returns different datatypes for each columns (float, int, string, bool, etc.) df.dtypes
 # Returns the first x number of rows when head(x). Without a number it returns 5 df.head()
# Returns the last x number of rows when tail(x). Without a number it returns 5 df.tail()

# Returns true for a column having null values, else false df.isnull().any()
# Returns basic information on all columns df.info()
# Returns basic statistics on numeric columns df.describe().T
# Importing essential libraries import
matplotlib.pyplot as plt %matplotlib
inline import seaborn as sns
# Plotting histogram for the entire dataset
fig = plt.figure(figsize = (15,15)) ax = fig.gca() g = df.hist(ax=ax)
# Visualization to check if the dataset is balanced or not g =
 sns.countplot(x='target', data=df) plt.xlabel('Target') plt.ylabel('Count')
# Selecting correlated features using Heatmap
# Get correlation of all the features of the dataset corr_matrix = df.corr() top_corr_features =
corr_matrix.index

# Plotting the heatmap plt.figure(figsize=(20,20)) sns.heatmap(data=df[top_corr_features].corr(),
annot=True, cmap='RdYlGn') dataset = pd.get_dummies(df, columns=['sex', 'cp', 'fbs', 'restecg',
'exang', 'slope', 'ca', 'thal']) dataset.columns from sklearn.preprocessing import StandardScaler 58
standScaler = StandardScaler()
columns_to_scale = ['age', 'trestbps', 'chol', 'thalach', 'oldpeak']
dataset[columns_to_scale] = standScaler.fit_transform(dataset[columns_to_scale])
 dataset.head()
 # Splitting the dataset into dependent and independent features X =
dataset.drop('target', axis=1) y = dataset['target']
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.20,random_state=42)

# Importing essential libraries from sklearn.neighbors
import KNeighborsClassifier from sklearn.model_selection
import cross_val_score
KNN=KNeighborsClassifier(n_neighbors=5,weights='uniform',algorithm='auto',leaf_size=30
 ,p=2,metric='minkowski') KNN.fit(X_train,y_train) y_pred=KNN.predict(X_test) y_pred from
sklearn.metrics import accuracy_score,classification_report,
confusion_matrix ac=accuracy_score(y_test,y_pred)*100 print("KNN_Accuracy:",ac)
 print("Classification Report:")
print(classification_report(y_test, y_pred))
cm=confusion_matrix(y_test, y_pred) print("Confusion Matrix:")
plt.figure(figsize=(8, 6)) sns.heatmap(cm, annot=True, fmt="d", cmap="Blues")
plt.xlabel("Predicted Label") plt.ylabel("True Label")
plt.title("KNeighborsClassifier_Confusion Matrix")
plt.show()
# Importing essential libraries from sklearn.tree
import DecisionTreeClassifier dt=DecisionTreeClassifier()
dt.fit(X_train,y_train) y_pred1=dt.predict(X_test)
y_pred1 ac1=accuracy_score(y_test,y_pred1)*100
print("DecisionTreeClassifier Acuracy:",ac1)
print("Classification Report:")
print(classification_report(y_test, y_pred1))

cm=confusion_matrix(y_test, y_pred1) print("Confusion Matrix:")

plt.figure(figsize=(8, 6)) sns.heatmap(cm, annot=True, fmt="d", cmap="Blues")

plt.xlabel("Predicted Label") plt.ylabel("True Label")

 plt.title("DecisionTreeClassifier_Confusion Matrix") plt.show()

# Importing essential libraries from sklearn.ensemble

 import RandomForestClassifier rf=RandomForestClassifier()

rf.fit(X_train,y_train)

 y_pred2=dt.predict(X_test) y_pred2 from sklearn.metrics

import accuracy_score accuracy_score(y_test,y_pred2)

print("Classification Report:") print(classification_report(y_test, y_pred2))

cm=confusion_matrix(y_test, y_pred2)print("Confusion 59 Matrix:") plt.figure(figsize=(8, 6))

sns.heatmap(cm, annot=True, fmt="d", cmap="Blues") plt.xlabel("Predicted Label")

 plt.ylabel("True Label")

plt.title("RandomForestClassifier_Confusion Matrix")

 plt.show()
```