Source Code:

```
#importing libraries

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib import rcParams
from matplotlib.cm import rainbow
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes  import  GaussianNB
from sklearn.metrics import  accuracy_score, classification_report




#Reading data set file

data= pd.read_csv('/content/dataset.csv')
data.info()

data = pd.get_dummies(data, columns = ['sex', 'cp', 'fbs', 'restecg', 'exang', 'slope', 'ca', 'thal'])
standardScaler = StandardScaler()
columns_to_scale = ['age', 'trestbps', 'chol', 'thalach', 'oldpeak']
data[columns_to_scale] = standardScaler.fit_transform(data[columns_to_scale])


y = data['target']
X = data.drop(['target'], axis = 1)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.33, random_state = 0)
```

```
#KNN

knn_sc = []
for k in range(1,21):
    knn = KNeighborsClassifier(n_neighbors = k)
    knn.fit(X_train, y_train)
    knn_sc.append(knn.score(X_test, y_test))

plt.plot([k for k in range(1, 21)], knn_sc, color = 'red')
for i in range(1,21):
    plt.text(i, knn_sc[i-1], (i, knn_sc[i-1]))
plt.xticks([i for i in range(1, 21)])
plt.xlabel('Number of Neighbors (K)')
plt.ylabel('Scores')

#SVM

svm_sc = []
kernels = ['linear', 'poly', 'rbf', 'sigmoid']
for i in range(len(kernels)):
    svm= SVC(kernel = kernels[i])
    svm.fit(X_train, y_train)
    svm_sc.append(svm.score(X_test, y_test))
colors = rainbow(np.linspace(0, 1, len(kernels)))
plt.bar(kernels, svm_sc, color = colors)
for i in range(len(kernels)):
    plt.text(i, svm_sc[i], svm_sc[i])
plt.xlabel('Kernels')
plt.ylabel('Scores')

#Decision Tree

dt_sc = []
for i in range(1, len(X.columns) + 1):
    dt = DecisionTreeClassifier(max_features = i, random_state = 0)
    dt.fit(X_train, y_train)
    dt_sc.append(dt.score(X_test, y_test))

plt.plot([i for i in range(1, len(X.columns) + 1)], dt_sc, color = 'green')
for i in range(1, len(X.columns) + 1):
    plt.text(i, dt_sc[i-1], (i, dt_sc[i-1]))
plt.xticks([i for i in range(1, len(X.columns) + 1)])
plt.xlabel('Max features')
plt.ylabel('Scores')
```

```
#Naïve Bayes:

X = df.drop('target', axis=1)
y = df['target']

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

nb_model = GaussianNB()
nb_model.fit(X_train, y_train)

predictions = nb_model.predict(X_test)

accuracy = accuracy_score(y_test, predictions)
print(f'Accuracy: {accuracy:.2f}')

plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=['No Disease', 'Disease'],
yticklabels=['No Disease', 'Disease']
plt.xlabel('Predicted’)
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()

#Random Forest

rf_sc = []
estimators= [10, 100, 200, 500, 1000]
for i in estimators:
   rf = RandomForestClassifier(n_estimators = i, random_state = 0)
   rf.fit(X_train, y_train)
   rf_sc.append(rf.score(X_test, y_test))
colors = rainbow(np.linspace(0, 1, len(estimators)))
plt.bar([i for i in range(len(estimators))], rf_sc, color = colors, width = 0.8)
for i in range(len(estimators)):
   plt.text(i, rf_sc[i], rf_sc[i])
plt.xticks(ticks = [i for i in range(len(estimators))], labels = [str(estimator) for estimator in
estimators])
plt.xlabel('Number of estimators')
plt.ylabel('Scores')
```