```
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn import svm
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
import seaborn as sns
import matplotlib.pyplot as plt

diabetes = pd.read_csv(r"C:\Users\Varshini\Downloads\heart (1).csv")

diabetes.head()
```

```
   age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak
slope  \
0   52    1   0       125   212    0        1      168      0      1.0
2
1   53    1   0       140   203    1        0      155      1      3.1
0
2   70    1   0       145   174    0        1      125      1      2.6
0
3   61    1   0       148   203    0        1      161      0      0.0
2
4   62    0   0       138   294    1        1      106      0      1.9
1

   ca  thal  target
0   2     3       0
1   0     3       0
2   0     3       0
3   1     3       0
4   3     2       0
```

```
diabetes.shape
```

```
(1025, 14)
```

```
diabetes.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1025 entries, 0 to 1024
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   age       1025 non-null   int64
 1   sex       1025 non-null   int64
 2   cp        1025 non-null   int64
 3   trestbps  1025 non-null   int64
 4   chol      1025 non-null   int64
 5   fbs       1025 non-null   int64
```

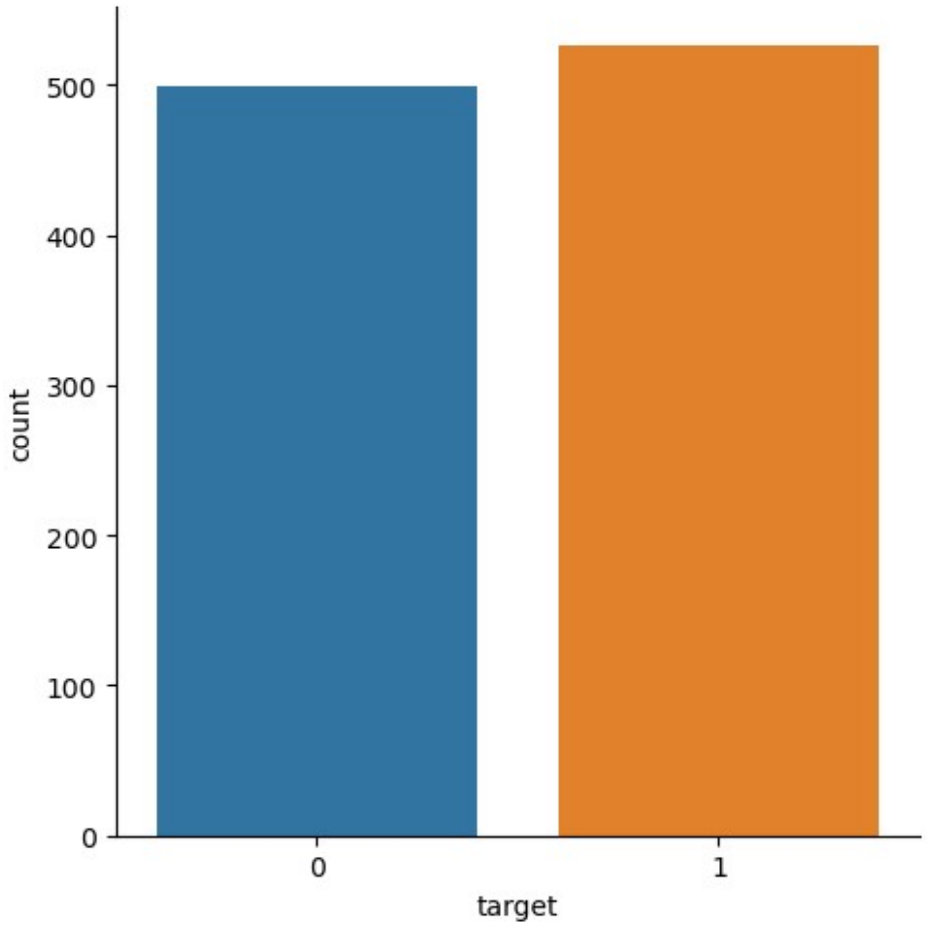
```
 6   restecg   1025 non-null   int64
 7   thalach   1025 non-null   int64
 8   exang     1025 non-null   int64
 9   oldpeak   1025 non-null   float64
 10  slope     1025 non-null   int64
 11  ca        1025 non-null   int64
 12  thal      1025 non-null   int64
 13  target    1025 non-null   int64
dtypes: float64(1), int64(13)
memory usage: 112.2 KB
```

```
diabetes.isnull().sum()
```

```
age         0
sex         0
cp          0
trestbps    0
chol        0
fbs         0
restecg     0
thalach     0
exang       0
oldpeak     0
slope       0
ca          0
thal        0
target      0
dtype: int64
```

```
sns.catplot(x='target',data=diabetes,kind='count')
```

```
<seaborn.axisgrid.FacetGrid at 0x25dc9354400>
```




```
correlation=diabetes.corr()
correlation
```

```
                age       sex        cp  trestbps      chol
fbs  \
age        1.000000 -0.103240 -0.071966  0.271121  0.219823  0.121243

sex       -0.103240  1.000000 -0.041119 -0.078974 -0.198258  0.027200

cp        -0.071966 -0.041119  1.000000  0.038177 -0.081641  0.079294

trestbps   0.271121 -0.078974  0.038177  1.000000  0.127977  0.181767

chol       0.219823 -0.198258 -0.081641  0.127977  1.000000  0.026917

fbs        0.121243  0.027200  0.079294  0.181767  0.026917  1.000000

restecg   -0.132696 -0.055117  0.043581 -0.123794 -0.147410 -0.104051

thalach   -0.390227 -0.049365  0.306839 -0.039264 -0.021772 -0.008866
```

```
exang     0.088163  0.139157 -0.401513  0.061197  0.067382  0.049261

oldpeak   0.208137  0.084687 -0.174733  0.187434  0.064880  0.010859

slope    -0.169105 -0.026666  0.131633 -0.120445 -0.014248 -0.061902

ca        0.271551  0.111729 -0.176206  0.104554  0.074259  0.137156

thal      0.072297  0.198424 -0.163341  0.059276  0.100244 -0.042177

target   -0.229324 -0.279501  0.434854 -0.138772 -0.099966 -0.041164


           restecg   thalach     exang   oldpeak     slope
ca  \
age      -0.132696 -0.390227  0.088163  0.208137 -0.169105  0.271551

sex      -0.055117 -0.049365  0.139157  0.084687 -0.026666  0.111729

cp        0.043581  0.306839 -0.401513 -0.174733  0.131633 -0.176206

trestbps -0.123794 -0.039264  0.061197  0.187434 -0.120445  0.104554

chol     -0.147410 -0.021772  0.067382  0.064880 -0.014248  0.074259

fbs      -0.104051 -0.008866  0.049261  0.010859 -0.061902  0.137156

restecg   1.000000  0.048411 -0.065606 -0.050114  0.086086 -0.078072

thalach   0.048411  1.000000 -0.380281 -0.349796  0.395308 -0.207888

exang    -0.065606 -0.380281  1.000000  0.310844 -0.267335  0.107849

oldpeak  -0.050114 -0.349796  0.310844  1.000000 -0.575189  0.221816

slope     0.086086  0.395308 -0.267335 -0.575189  1.000000 -0.073440

ca       -0.078072 -0.207888  0.107849  0.221816 -0.073440  1.000000

thal     -0.020504 -0.098068  0.197201  0.202672 -0.094090  0.149014

target    0.134468  0.422895 -0.438029 -0.438441  0.345512 -0.382085


             thal    target
age       0.072297 -0.229324
sex       0.198424 -0.279501
cp       -0.163341  0.434854
```

```
trestbps  0.059276 -0.138772
chol      0.100244 -0.099966
fbs      -0.042177 -0.041164
restecg  -0.020504  0.134468
thalach  -0.098068  0.422895
exang     0.197201 -0.438029
oldpeak   0.202672 -0.438441
slope    -0.094090  0.345512
ca        0.149014 -0.382085
thal      1.000000 -0.337838
target   -0.337838  1.000000
```

```
plt.figure(figsize=(10,10))
sns.heatmap(correlation,cbar=True,square=True,annot=True,cmap='Blues')
```

```
<AxesSubplot:>
```

```
diabetes.describe()
```

```
                age          sex           cp     trestbps        chol
\
count  1025.000000  1025.000000  1025.000000  1025.000000  1025.00000

mean     54.434146     0.695610     0.942439   131.611707   246.00000

std       9.072290     0.460373     1.029641    17.516718    51.59251

min      29.000000     0.000000     0.000000    94.000000   126.00000

25%      48.000000     0.000000     0.000000   120.000000   211.00000

50%      56.000000     1.000000     1.000000   130.000000   240.00000

75%      61.000000     1.000000     2.000000   140.000000   275.00000

max      77.000000     1.000000     3.000000   200.000000   564.00000


                fbs      restecg      thalach        exang      oldpeak
\
count  1025.000000  1025.000000  1025.000000  1025.000000  1025.000000

mean      0.149268     0.529756   149.114146     0.336585     1.071512

std       0.356527     0.527878    23.005724     0.472772     1.175053

min       0.000000     0.000000    71.000000     0.000000     0.000000

25%       0.000000     0.000000   132.000000     0.000000     0.000000
```

```
50%       0.000000     1.000000   152.000000     0.000000     0.800000

75%       0.000000     1.000000   166.000000     1.000000     1.800000

max       1.000000     2.000000   202.000000     1.000000     6.200000
```

```
             slope           ca         thal       target
count  1025.000000  1025.000000  1025.000000  1025.000000
mean      1.385366     0.754146     2.323902     0.513171
std       0.617755     1.030798     0.620660     0.500070
min       0.000000     0.000000     0.000000     0.000000
25%       1.000000     0.000000     2.000000     0.000000
50%       1.000000     0.000000     2.000000     1.000000
75%       2.000000     1.000000     3.000000     1.000000
max       2.000000     4.000000     3.000000     1.000000
```

```
diabetes['target'].value_counts()
```

```
1    526
0    499
Name: target, dtype: int64
```

0-->Healthy heart 1-->Defective heart

```
diabetes.groupby('target').mean()
```

```
              age       sex        cp    trestbps        chol
fbs  \
target

0       56.569138  0.827655  0.482966  134.106212  251.292585
0.164329
1       52.408745  0.570342  1.378327  129.245247  240.979087
0.134981

         restecg     thalach     exang   oldpeak     slope        ca
thal
target

0       0.456914  139.130261  0.549098  1.600200  1.166333  1.158317
2.539078
1       0.598859  158.585551  0.134981  0.569962  1.593156  0.370722
2.119772
```

```
X=diabetes.drop(columns='target',axis=1)
Y=diabetes['target']
```

```
X
```

```
      age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang
oldpeak  \
0      52    1   0       125   212    0        1      168      0
1.0
1      53    1   0       140   203    1        0      155      1
3.1
2      70    1   0       145   174    0        1      125      1
2.6
3      61    1   0       148   203    0        1      161      0
0.0
4      62    0   0       138   294    1        1      106      0
1.9
...   ...  ...  ..       ...   ...  ...      ...      ...    ...
...
1020   59    1   1       140   221    0        1      164      1
0.0
1021   60    1   0       125   258    0        0      141      1
2.8
1022   47    1   0       110   275    0        0      118      1
1.0
1023   50    0   0       110   254    0        0      159      0
0.0
1024   54    1   0       120   188    0        1      113      0
1.4

      slope  ca  thal
0         2   2     3
1         0   0     3
2         0   0     3
3         2   1     3
4         1   3     2
...     ...  ..   ...
1020      2   0     2
1021      1   1     3
1022      1   1     2
1023      2   0     2
1024      1   1     3

[1025 rows x 13 columns]

Y

0       0
1       0
2       0
3       0
4       0
       ..
1020    1
1021    0
```

```
1022    0
1023    1
1024    0
Name: target, Length: 1025, dtype: int64
```

```
scaler=StandardScaler()
```

```
standard_X=scaler.fit_transform(X)
```

```
standard_X
```

```
array([[-0.26843658,  0.66150409, -0.91575542, ...,  0.99543334,
         1.20922066,  1.08985168],
       [-0.15815703,  0.66150409, -0.91575542, ..., -2.24367514,
        -0.73197147,  1.08985168],
       [ 1.71659547,  0.66150409, -0.91575542, ..., -2.24367514,
        -0.73197147,  1.08985168],
       ...,
       [-0.81983438,  0.66150409, -0.91575542, ..., -0.6241209 ,
         0.23862459, -0.52212231],
       [-0.4889957 , -1.51170646, -0.91575542, ...,  0.99543334,
        -0.73197147, -0.52212231],
       [-0.04787747,  0.66150409, -0.91575542, ..., -0.6241209 ,
         0.23862459,  1.08985168]])
```

```
X=standard_X
```

```
X
```

```
array([[-0.26843658,  0.66150409, -0.91575542, ...,  0.99543334,
         1.20922066,  1.08985168],
       [-0.15815703,  0.66150409, -0.91575542, ..., -2.24367514,
        -0.73197147,  1.08985168],
       [ 1.71659547,  0.66150409, -0.91575542, ..., -2.24367514,
        -0.73197147,  1.08985168],
       ...,
       [-0.81983438,  0.66150409, -0.91575542, ..., -0.6241209 ,
         0.23862459, -0.52212231],
       [-0.4889957 , -1.51170646, -0.91575542, ...,  0.99543334,
        -0.73197147, -0.52212231],
       [-0.04787747,  0.66150409, -0.91575542, ..., -0.6241209 ,
         0.23862459,  1.08985168]])
```

```
Y
```

```
0       0
1       0
2       0
3       0
4       0
       ..
1020    1
```

```
1021    0
1022    0
1023    1
1024    0
Name: target, Length: 1025, dtype: int64
```

```
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.2,strat
ify=Y,random_state=2)
```

```
X.shape,X_train.shape,X_test.shape
```

```
((1025, 13), (820, 13), (205, 13))
```

Support Vector Machine

```
model1=svm.SVC(kernel='linear')
```

```
model1.fit(X_train,Y_train)
```

```
SVC(kernel='linear')
```

```
X_training_prediction=model1.predict(X_train)
training_data_accuracy=accuracy_score(X_training_prediction,Y_train)
```

```
training_data_accuracy
```

```
0.8670731707317073
```

```
X_test_prediction=model1.predict(X_test)
test_data_accuracy=accuracy_score(X_test_prediction,Y_test)
```

```
test_data_accuracy
```

```
0.8195121951219512
```

Logistic Regression

```
model2=LogisticRegression()
```

```
model2.fit(X_train,Y_train)
```

```
LogisticRegression()
```

```
X_train_prediction=model2.predict(X_train)
training_data_accuracy=accuracy_score(X_train_prediction,Y_train)
```

```
training_data_accuracy
```

```
0.8585365853658536
```

```
X_test_prediction=model2.predict(X_test)
test_data_accuracy=accuracy_score(X_test_prediction,Y_test)
```

```
test_data_accuracy
```

```
0.8048780487804879
```

Random Forest Classifier

```
X=diabetes.drop(columns='target',axis=1)
```

```
Y=diabetes['target']
```

```
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.2,strat
ify=Y,random_state=2)
```

```
X.shape,X_train.shape,X_test.shape
```

```
((1025, 13), (820, 13), (205, 13))
```

```
X
```

```
      age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang
oldpeak  \
0      52    1   0       125   212    0        1      168      0
1.0
1      53    1   0       140   203    1        0      155      1
3.1
2      70    1   0       145   174    0        1      125      1
2.6
3      61    1   0       148   203    0        1      161      0
0.0
4      62    0   0       138   294    1        1      106      0
1.9
...   ...  ...  ..       ...   ...  ...      ...      ...    ...
...
1020   59    1   1       140   221    0        1      164      1
0.0
1021   60    1   0       125   258    0        0      141      1
2.8
1022   47    1   0       110   275    0        0      118      1
1.0
1023   50    0   0       110   254    0        0      159      0
0.0
1024   54    1   0       120   188    0        1      113      0
1.4

      slope  ca  thal
0         2   2     3
1         0   0     3
2         0   0     3
3         2   1     3
4         1   3     2
...     ...  ..   ...
1020      2   0     2
1021      1   1     3
```

```
1022      1   1     2
1023      2   0     2
1024      1   1     3

[1025 rows x 13 columns]
```

```
Y
```

```
0       0
1       0
2       0
3       0
4       0
       ..
1020    1
1021    0
1022    0
1023    1
1024    0
Name: target, Length: 1025, dtype: int64
```

```
model3=RandomForestClassifier()
```

```
model3.fit(X_train,Y_train)
```

```
RandomForestClassifier()
```

```
X_train_prediction=model3.predict(X_train)
training_data_accuracy=accuracy_score(X_train_prediction,Y_train)
```

```
training_data_accuracy
```

```
1.0
```

```
X_test_prediction=model3.predict(X_test)
test_data_accuracy=accuracy_score(X_test_prediction,Y_test)
```

```
test_data_accuracy
```

```
1.0
```

since Random Forest Classifier has more accuracy than SVM,Logistic Regression. Hence Rnadom Forest Classifier is the best algorithm to use for heart disease prediction dataset