1. **Introduction**

The study examines the effectiveness and potential of machine learning algorithms by taking into account a number of practical issues. Machine learning algorithms are utilized to anticipate Heart disease. There are millions of people who are diagnosed with this disease, which is a fatal health issue. In such cases, it is very important to have a kind of tool which could be used for early detection of this disease. Machine learning algorithms are a promising technique for early detection since they can predict the likelihood of developing cardiac disease. These algorithms may scan a substantial amount of patient data and identify patterns and trends that can help in the early detection of cardiac illness. The main goal of this task is to build a model using machine learning algorithms that could be used by medical professionals to make wise decisions.

1. **Methods** *KNN Model*

An appreciated machine learning approach for classification and regression tasks is the K-Nearest Neighbors (KNN) algorithm. KNN can be used in the framework of heart disease prediction for estimating the risk of heart disease based on patient data. The KNN algorithm categorises unclassified data points based on their proximity and similarity to other accessible data points. The main premise of this technique is that related data points can be found nearby. Due to its simplicity of usage, suitability for solving classification and regression issues, and the ease with which the findings it produces may be interpreted, it is frequently used to resolve issues in a variety of industries. We must first preprocess and normalize the dataset including the patient data before we can construct the KNN model for heart disease prediction. The trained model can then be used in the testing set of new patients to forecast their likelihood of developing heart disease. We assess the model's accuracy using a number of metrics, including precision, recall, and F1-score.

SVM Model

Heart disease can be predicted using the potent “machine learning algorithm Support Vector Machine (SVM)”. A “supervised learning” method called SVM uses labeled data to divide fresh data into many categories. The SVM algorithm may evaluate patient data in the context of heart disease prediction and categorize patients as either having heart disease or not. Before establishing a Support Vector Machine (SVM) model for coronary artery bypass graft prediction, we need to gather information about patients on age, gender, blood pressure, cholesterol levels, and other crucial parameters required. SVM classifiers excel in high-dimensional space performance and have exceptional accuracy since they only employ a portion of the training data, which uses less memory. It works rather effectively when there is a big gap between the classes. High-dimensional spaces are ideal for SVM because they make effective use of memory and are useful when the number of dimensions exceeds the number of samples. The SVM algorithm will be trained using this data to differentiate between people who are most likely to develop cardiovascular conditions and those who are not. Using patient data and the SVM model that has been trained, it is possible to predict the risk that new patients will ultimately get heart disease. To evaluate the model's accuracy, a number of determinants can be employed such as precision, recall, and F1-score.

Overall, an SVM model can be a helpful tool for predicting cardiac disease, can assist health care providers in making knowledgeable decisions, and might improve patient outcomes.

Random Forest Model

The widely-used "machine learning technique" known as "random forest" was created by "Leo Breinman and Adele Cutler". Random Forest can be used to analyze different risk variables in the context of heart disease prediction and produce a forecast of whether a person is at high risk of getting the condition. The approach creates numerous decision trees using a subset of features that are randomly chosen from the dataset. A random subset of the data is used to train each tree in the forest, and each tree utilizes a distinct set of characteristics. In order to arrive at a final forecast, the algorithm integrates the results of all the decision trees in the forest. The ability to handle huge datasets with numerous features, the handling of missing data, and the resistance to overfitting are only a few benefits of random forest for heart disease prediction. The dataset can then be divided into training and testing sets, and the model can be trained using the training set. The Random Forest algorithm will build several decision trees during the training phase using subsets of the training set's samples and features. The algorithm may also shed light on the key elements that affect the likelihood of developing heart disease. In conclusion, the random forest algorithm has been applied successfully in numerous research and is an effective tool for predicting cardiac disease. By utilizing this method, healthcare professionals may create more precise forecasts and superior treatment plans for their patients.

3. Results and Analysis

Importing essential libraries

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import f1_score
from sklearn.model_selection import train_test_split
import warnings
warnings.filterwarnings('ignore')
import sklearn.neighbors
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
```

Reading the Dataset

In [3]:
```
hdf = pd.read_csv("Heart_Disease_Prediction (2).csv")
```

A visual of how the "Heart_Disease_Prediction" dataset has been loaded into Jupyter by using ".read". The figure also shows how the dataset is imported as a data frame "hdf" into

Jupyter Notebook. Additionally, the figure gives a clear picture of the contents of the dataframe created.

Displaying the data frame

In [4]:
```
hdf
```

Out[4]:

	Age	Sex	Chest pain type	BP	Cholesterol	FBS over 120	EKG results	Max HR	Exercise angina	ST depression	Slope of ST	Number of vessels fluro
0	70	1	4	130	322	0	2	109	0	2.4	2	3
1	67	0	3	115	564	0	2	160	0	1.6	2	0
2	57	1	2	124	261	0	0	141	0	0.3	1	0
3	64	1	4	128	263	0	0	105	1	0.2	2	1
4	74	0	2	120	269	0	2	121	1	0.2	1	1
...	...	...	...	...	...	...	...	...	...	...	...	...
265	52	1	3	172	199	1	0	162	0	0.5	1	0
266	44	1	2	120	263	0	0	173	0	0.0	1	0
267	56	0	2	140	294	0	2	153	0	1.3	2	0
268	57	1	4	140	192	0	0	148	0	0.4	2	0
269	67	1	4	160	286	0	2	108	1	1.5	2	3

270 rows × 14 columns

Descriptive Summary

In [5]:
```
hdf.describe()
```

Out[5]:

	Age	Sex	Chest pain type	BP	Cholesterol	FBS over 120	EKG results	Ma
count	270.000000	270.000000	270.000000	270.000000	270.000000	270.000000	270.000000	270.00
mean	54.433333	0.677778	3.174074	131.344444	249.659259	0.148148	1.022222	149.67
std	9.109067	0.468195	0.950090	17.861608	51.686237	0.355906	0.997891	23.16
min	29.000000	0.000000	1.000000	94.000000	126.000000	0.000000	0.000000	71.00
25%	48.000000	0.000000	3.000000	120.000000	213.000000	0.000000	0.000000	133.00
50%	55.000000	1.000000	3.000000	130.000000	245.000000	0.000000	2.000000	153.50
75%	61.000000	1.000000	4.000000	140.000000	280.000000	0.000000	2.000000	166.00
max	77.000000	1.000000	4.000000	200.000000	564.000000	1.000000	2.000000	202.00

Data frame overview

In [6]:
```
hdf.shape
```

Out[6]:
```
(270, 14)
```

In [7]:
```
hdf.head()
```

Out[7]:

	Age	Sex	Chest pain type	BP	Cholesterol	FBS over 120	EKG results	Max HR	Exercise angina	ST depression	Slope of ST	Number of vessels fluro
0	70	1	4	130	322	0	2	109	0	2.4	2	3
1	67	0	3	115	564	0	2	160	0	1.6	2	0
2	57	1	2	124	261	0	0	141	0	0.3	1	0
3	64	1	4	128	263	0	0	105	1	0.2	2	1
4	74	0	2	120	269	0	2	121	1	0.2	1	1

Checking for null values

In [8]:
```
hdf.isnull().sum()
```

Out[8]:
```
Age                     0
Sex                     0
Chest pain type         0
BP                      0
Cholesterol             0
FBS over 120            0
EKG results             0
Max HR                  0
Exercise angina         0
ST depression           0
Slope of ST             0
Number of vessels fluro 0
Thallium                0
Heart Disease           0
dtype: int64
```

The Python code ".isnull.sum()" is applied for checking the "total number of null values" present in the given dataset. It can be noted from the above figure that no null values are present in the dataframe.

Checking for imbalanced data based on "Sex"

In [9]:
```
hdf['Sex'].value_counts()
```

Out[9]:
```
1    183
0     87
Name: Sex, dtype: int64
```

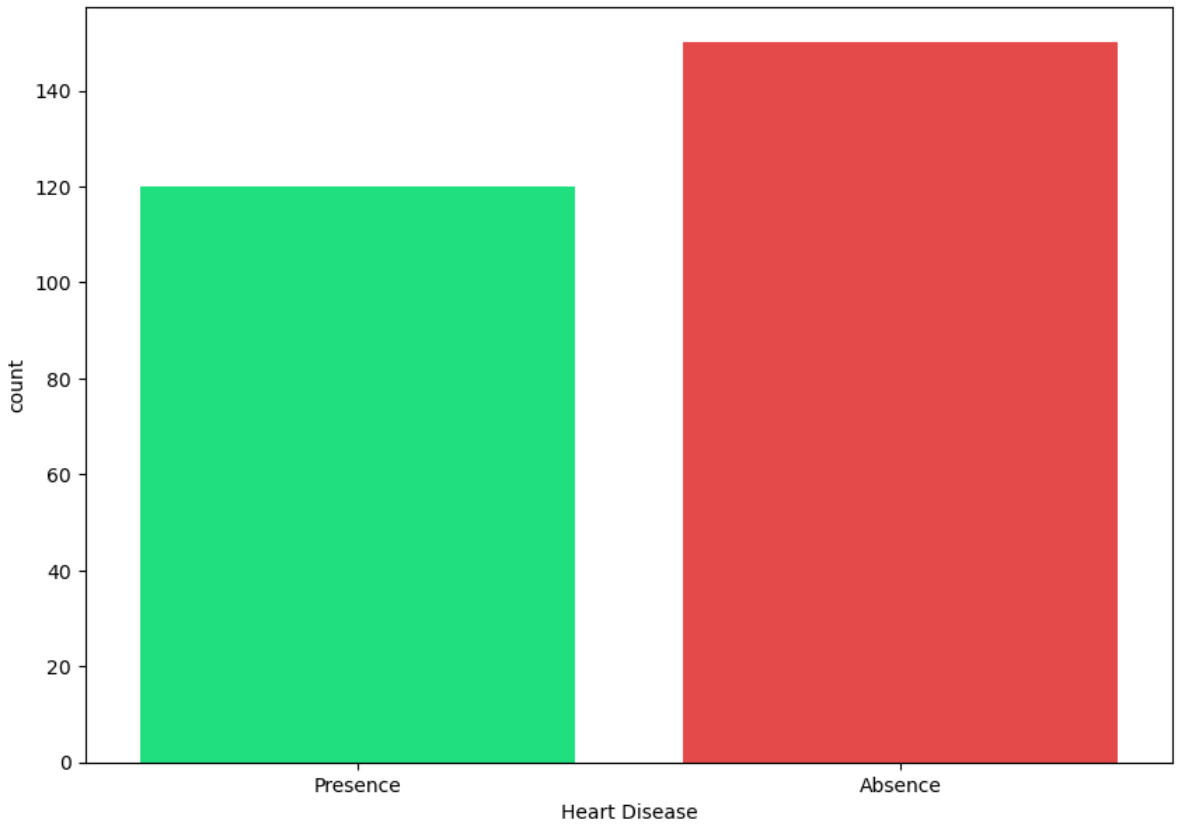
Checking for imbalanced data based on outcome

In [10]:
```
hdf['Heart Disease'].value_counts()
```

Out[10]:
```
Absence     150
Presence    120
Name: Heart Disease, dtype: int64
```

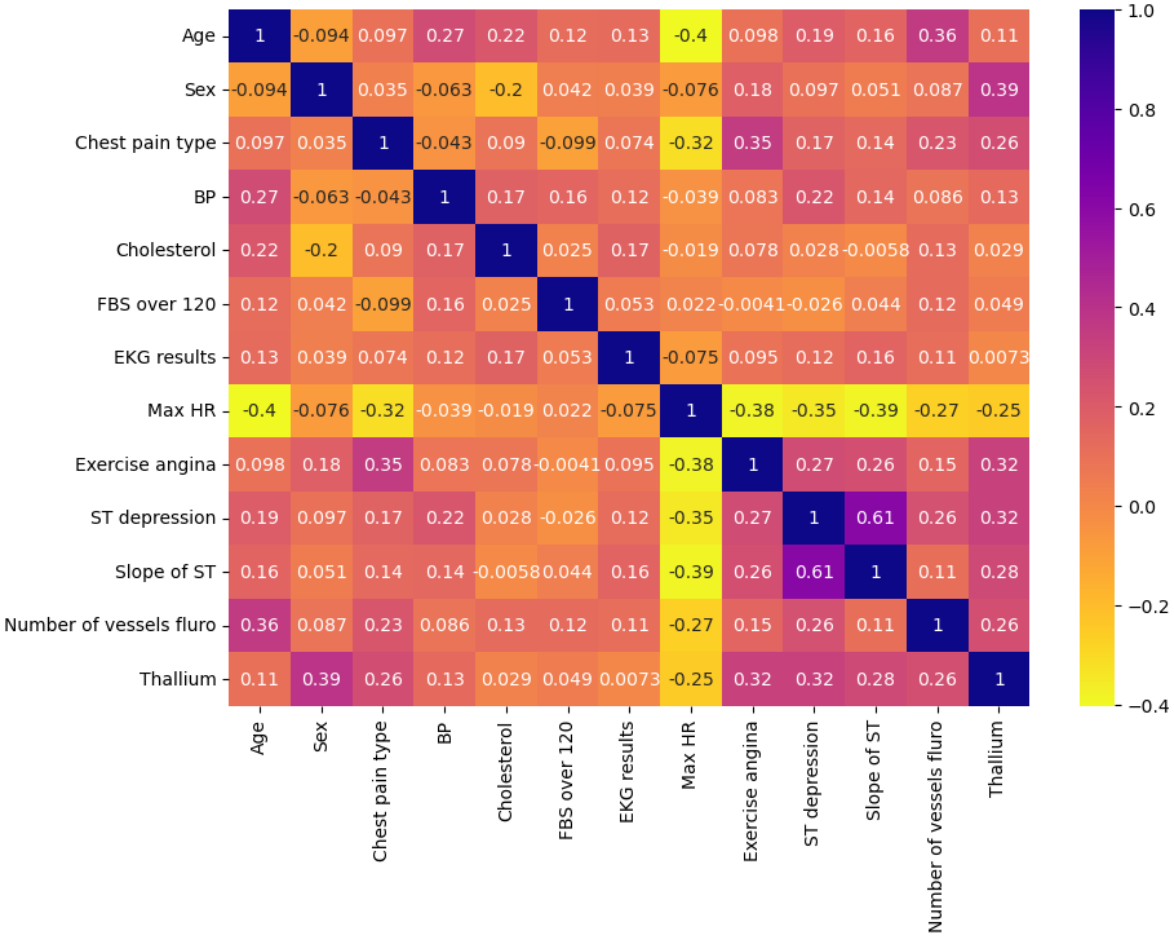
In [11]:
```
fig, ax = plt.subplots(figsize=(10, 7))
sns.countplot(x='Heart Disease', data=hdf, palette=['#00FF7F', '#FF3030'])
```

Out[11]:
```
<Axes: xlabel='Heart Disease', ylabel='count'>
```



Showing correlation heatmap

In [12]:
```
fig, ax = plt.subplots(figsize=(10, 7))
sns.heatmap(hdf.corr(), annot=True, cmap='plasma_r', ax=ax)
plt.show()
```



KNN

Splitting the dataset into training and testing sets

In [13]:
```
x = hdf.iloc[:, :-2]
y = hdf.iloc[:, -1]
x_train, x_test, y_train, y_test = train_test_split(x, y, random_state = 0, test_s
```

Appylying standard scaler as standardization technique

In [14]:
```
scx = StandardScaler()
x_train = scx.fit_transform(x_train)
x_test = scx.transform(x_test)
```

Looking for optimal number of nearest neighbours

In [15]:
```
import math
math.sqrt(len(y_test))
```

Out[15]:
```
9.746794344808963
```

Building KNN Model

In [16]:
```
cf = KNeighborsClassifier(n_neighbors = 9, p = 2, metric = 'euclidean')
cf.fit(x_train,y_train)
KNeighborsClassifier(metric='euclidean', n_neighbors=9)
y_pred = cf.predict(x_test)
y_pred
```

Out[16]:
```
array(['Absence', 'Absence', 'Presence', 'Presence', 'Absence', 'Absence',
       'Absence', 'Absence', 'Presence', 'Absence', 'Absence', 'Absence',
       'Presence', 'Presence', 'Presence', 'Absence', 'Absence',
       'Presence', 'Presence', 'Absence', 'Presence', 'Absence',
       'Presence', 'Presence', 'Absence', 'Presence', 'Absence',
       'Absence', 'Absence', 'Presence', 'Presence', 'Absence',
       'Presence', 'Absence', 'Presence', 'Absence', 'Presence',
       'Presence', 'Absence', 'Absence', 'Absence', 'Absence', 'Absence',
       'Presence', 'Absence', 'Absence', 'Presence', 'Absence',
       'Presence', 'Presence', 'Presence', 'Presence', 'Absence',
       'Absence', 'Absence', 'Absence', 'Absence', 'Presence', 'Absence',
       'Presence', 'Absence', 'Absence', 'Absence', 'Absence', 'Absence',
       'Presence', 'Absence', 'Presence', 'Absence', 'Absence',
       'Presence', 'Presence', 'Presence', 'Absence', 'Presence',
       'Presence', 'Presence', 'Absence', 'Absence', 'Presence',
       'Presence', 'Absence', 'Absence', 'Absence', 'Presence', 'Absence',
       'Presence', 'Absence', 'Presence', 'Absence', 'Presence',
       'Absence', 'Absence', 'Presence', 'Absence'], dtype=object)
```

A model is built using the “K-Nearest Neighbors” algorithm and then shows the accuracy score of the model. From the figure it can be noted that the model has an accuracy score of 75%.

Displaying Accuracy Score

In [17]:
```
print("Accuracy Score of KNN: ", accuracy_score(y_test,y_pred)*100)
```

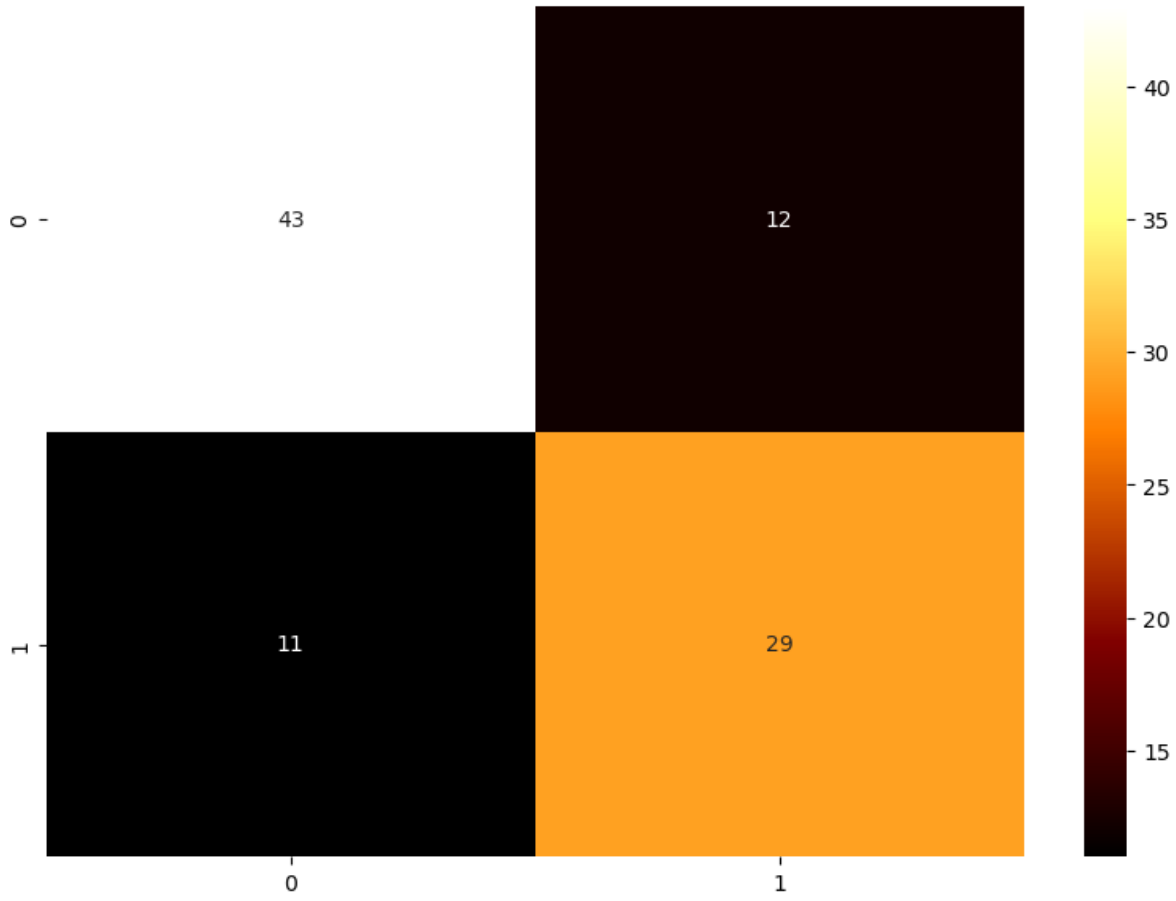
```
Accuracy Score of KNN:  75.78947368421053
```

Showing confusion matrix heatmap

In [18]:
```
cfm = confusion_matrix(y_test,y_pred)
print("Confusion Matrix")
print(cfm)
```

```
Confusion Matrix
[[43 12]
 [11 29]]
```

In [19]:
```
fig, ax = plt.subplots(figsize=(10, 7))
sns.heatmap(cfm, annot=True, cmap='afmhot', ax=ax)
plt.show()
```



Building SVM Model

In [20]:
```
from sklearn import svm
cf = svm.SVC(kernel='rbf')
cf.fit(x_train,y_train)
y_pred = cf.predict(x_test)
y_pred
```

Out[20]:
```
array(['Absence', 'Absence', 'Presence', 'Presence', 'Absence', 'Absence',
       'Absence', 'Absence', 'Presence', 'Absence', 'Absence', 'Absence',
       'Presence', 'Presence', 'Presence', 'Absence', 'Absence',
       'Presence', 'Presence', 'Absence', 'Presence', 'Absence',
       'Presence', 'Presence', 'Presence', 'Absence', 'Absence',
       'Absence', 'Absence', 'Absence', 'Presence', 'Absence', 'Presence',
       'Absence', 'Presence', 'Absence', 'Presence', 'Presence',
       'Absence', 'Absence', 'Absence', 'Absence', 'Absence', 'Absence',
       'Absence', 'Absence', 'Presence', 'Absence', 'Presence',
       'Presence', 'Presence', 'Presence', 'Absence', 'Absence',
       'Absence', 'Absence', 'Absence', 'Presence', 'Absence', 'Presence',
       'Absence', 'Absence', 'Absence', 'Absence', 'Absence', 'Presence',
       'Presence', 'Presence', 'Absence', 'Absence', 'Presence',
       'Absence', 'Presence', 'Absence', 'Presence', 'Absence',
       'Presence', 'Absence', 'Absence', 'Presence', 'Presence',
       'Absence', 'Presence', 'Absence', 'Presence', 'Absence',
       'Presence', 'Presence', 'Presence', 'Absence', 'Presence',
       'Absence', 'Absence', 'Presence', 'Absence'], dtype=object)
```

A model is built using the “Support Vector Machine” SVM algorithm and then shows the accuracy score of the model. From the figure it can be noted that the model has an accuracy score of about 77%.

Displaying Accuracy Score

In [21]:
```
print("Accuracy Score of SVM: ", accuracy_score(y_test,y_pred)*100)
```

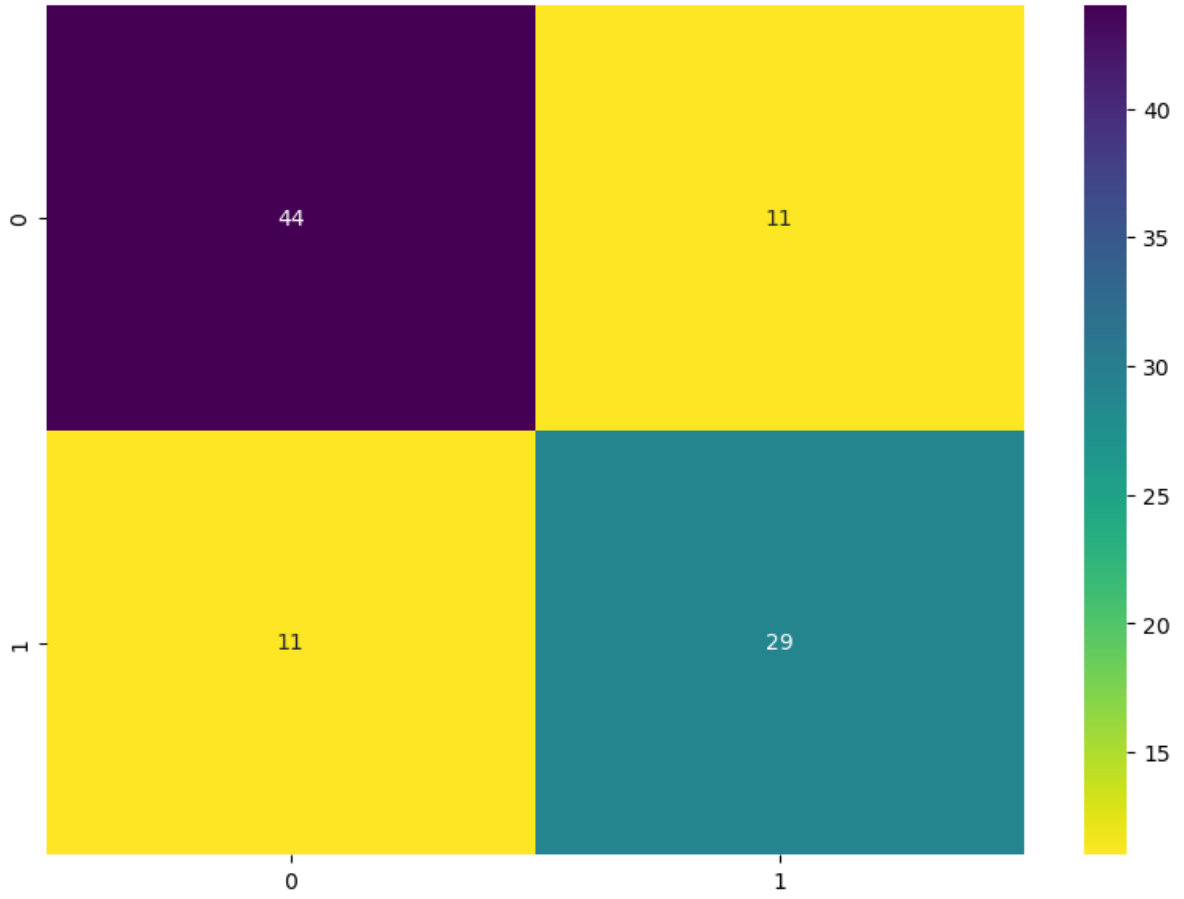
```
Accuracy Score of SVM:  76.84210526315789
```

Showing confusion matrix heatmap

In [22]:
```
cfm = confusion_matrix(y_test,y_pred)
print("Confusion Matrix")
print(cfm)
```

```
Confusion Matrix
[[44 11]
 [11 29]]
```

In [23]:
```
fig, ax = plt.subplots(figsize=(10, 7))
sns.heatmap(cfm, annot=True, cmap='viridis_r', ax=ax)
plt.show()
```



Building Random Forest Model

In [24]:
```
cf = RandomForestClassifier(n_estimators=500, random_state=12, max_depth=5)
cf.fit(x_train,y_train)
cf_pred = cf.predict(x_test)
cf_pred
```

Out[24]:
```
array(['Absence', 'Absence', 'Presence', 'Presence', 'Absence',
       'Presence', 'Absence', 'Absence', 'Presence', 'Absence', 'Absence',
       'Absence', 'Presence', 'Presence', 'Presence', 'Absence',
       'Absence', 'Presence', 'Presence', 'Absence', 'Presence',
       'Absence', 'Presence', 'Presence', 'Presence', 'Absence',
       'Absence', 'Absence', 'Presence', 'Presence', 'Absence',
       'Presence', 'Presence', 'Absence', 'Presence', 'Absence',
       'Presence', 'Presence', 'Absence', 'Absence', 'Absence', 'Absence',
       'Absence', 'Presence', 'Absence', 'Absence', 'Presence', 'Absence',
       'Presence', 'Presence', 'Presence', 'Absence', 'Absence',
       'Presence', 'Absence', 'Absence', 'Absence', 'Presence', 'Absence',
       'Presence', 'Absence', 'Absence', 'Absence', 'Absence', 'Absence',
       'Presence', 'Absence', 'Presence', 'Absence', 'Absence', 'Absence',
       'Presence', 'Presence', 'Absence', 'Presence', 'Absence',
       'Presence', 'Absence', 'Absence', 'Presence', 'Presence',
       'Presence', 'Absence', 'Absence', 'Presence', 'Absence',
       'Presence', 'Presence', 'Presence', 'Absence', 'Presence',
       'Absence', 'Absence', 'Presence', 'Absence'], dtype=object)
```

Demonstrated above is how a model is built using the “Random Forest” algorithm and then shows the accuracy score of the model. From the figure it can be noted that the model has an accuracy score of about 82%.

Displaying Accuracy Score

In [25]:
```
acs = accuracy_score(y_test, cf_pred)
print("Accuracy of Random Forest:", acs*100)
```

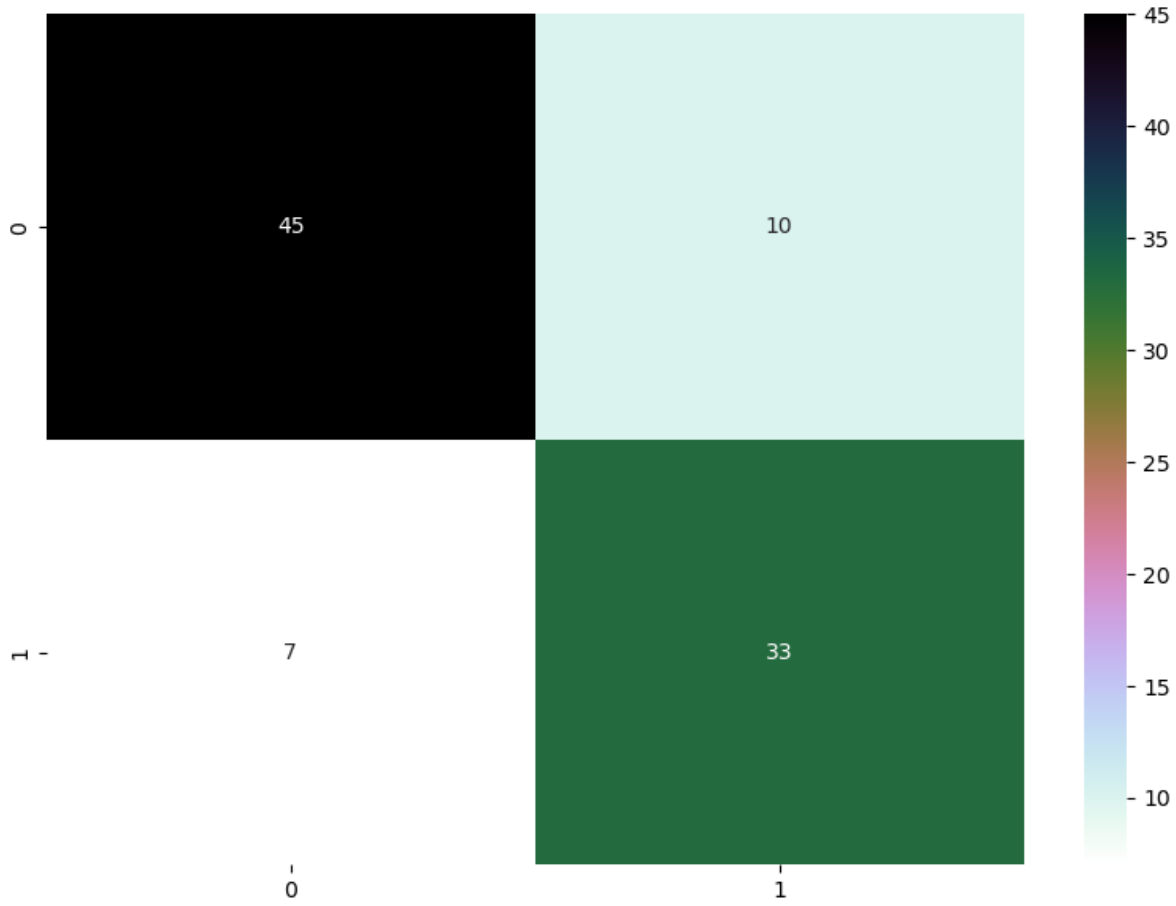
```
Accuracy of Random Forest: 82.10526315789474
```

Showing confusion matrix heatmap

In [26]:
```
rf_cfm = confusion_matrix(y_test, cf_pred)
print("Confusion Matrix")
print(rf_cfm)
```

```
Confusion Matrix
[[45 10]
 [ 7 33]]
```

In [27]:
```
fig, ax = plt.subplots(figsize=(10, 7))
sns.heatmap(rf_cfm, annot=True, cmap='cubehelix_r', ax=ax)
plt.show()
```



Classification Report

In [28]:
```
from sklearn.metrics import classification_report
print(classification_report(y_test,cf_pred))
```

```
              precision    recall  f1-score   support

     Absence       0.87      0.82      0.84        55
    Presence       0.77      0.82      0.80        40

    accuracy                           0.82        95
   macro avg       0.82      0.82      0.82        95
weighted avg       0.82      0.82      0.82        95
```

The method by which the predictions of the constructed categorical models are used to generate the “classification report for the given dataset” is clearly depicted in the aforementioned figure. The parameters of the aforementioned report include “precision”, “f1-score”, “recall”, and “support”.

1. Conclusion

The conclusion report of this study compares the performance of three popular machine learning algorithms, KNN, SVM, and “Random Forest, for predicting heart disease”. The findings of the study show that the Random Forest algorithm achieved the highest accuracy and F1-score, followed by SVM and KNN. Additionally, the results also indicate that “feature selection” using “correlation-based feature selection” (CFS) and “principal component

analysis" (PCA) can significantly improve the performance of all three algorithms in the areas of heart disease prediction

In [ ]:

In [ ]: