In [1]:
```
import os
import numpy as np
import pandas as pd
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns
```

In [3]:
```
df=pd.read_csv('C:/Users/User/Downloads/heart.csv')
```

In [4]:
```
df.head()
```

Out[4]:

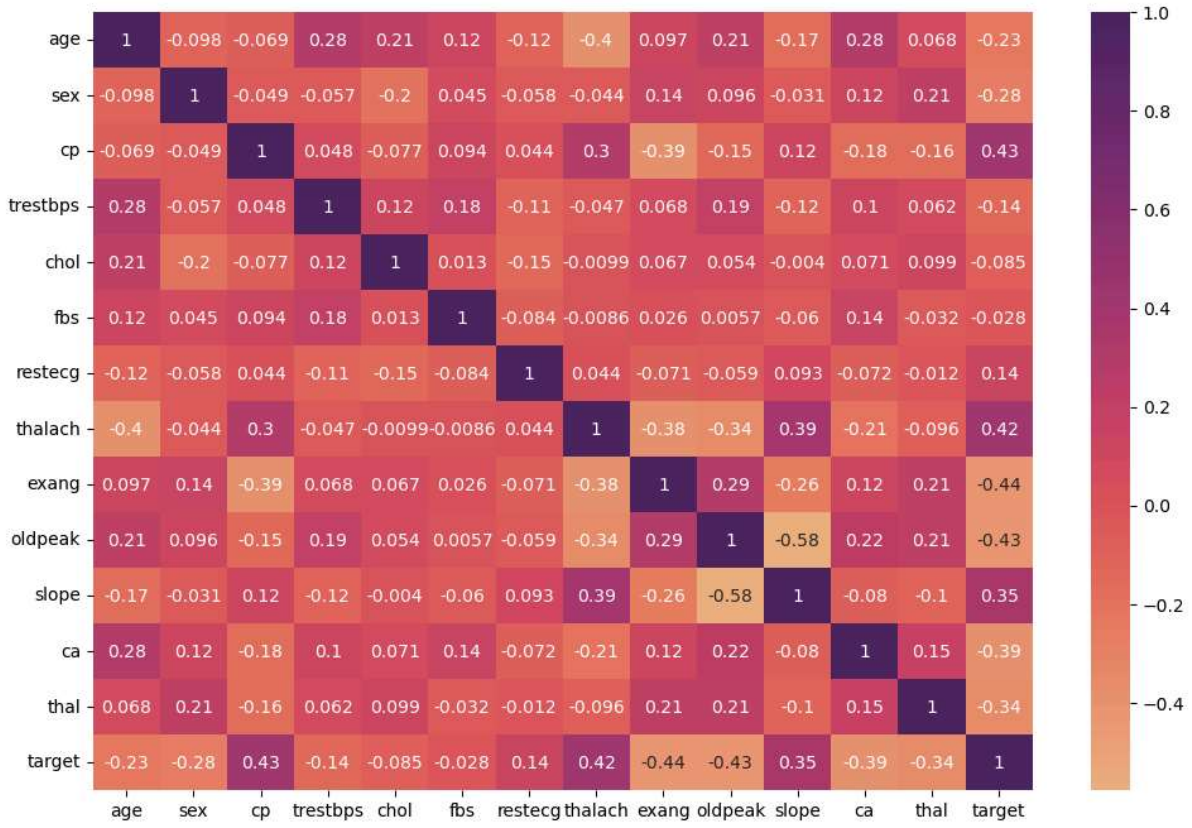
	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1

In [6]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   age       303 non-null    int64  
 1   sex       303 non-null    int64  
 2   cp        303 non-null    int64  
 3   trestbps  303 non-null    int64  
 4   chol      303 non-null    int64  
 5   fbs       303 non-null    int64  
 6   restecg   303 non-null    int64  
 7   thalach   303 non-null    int64  
 8   exang     303 non-null    int64  
 9   oldpeak   303 non-null    float64
 10  slope     303 non-null    int64  
 11  ca        303 non-null    int64  
 12  thal      303 non-null    int64  
 13  target    303 non-null    int64  
dtypes: float64(1), int64(13)
memory usage: 33.3 KB
```

In [5]:
```
plt.figure(figsize=(12,8))
sns.heatmap(df.corr(),square=False,cmap='flare',annot=True)
```

Out[5]: <Axes: >



In [7]:
```
#creating features (x) and targets(y) variables
x=df.iloc[:,:-1]
y=df.iloc[:,-1]
```

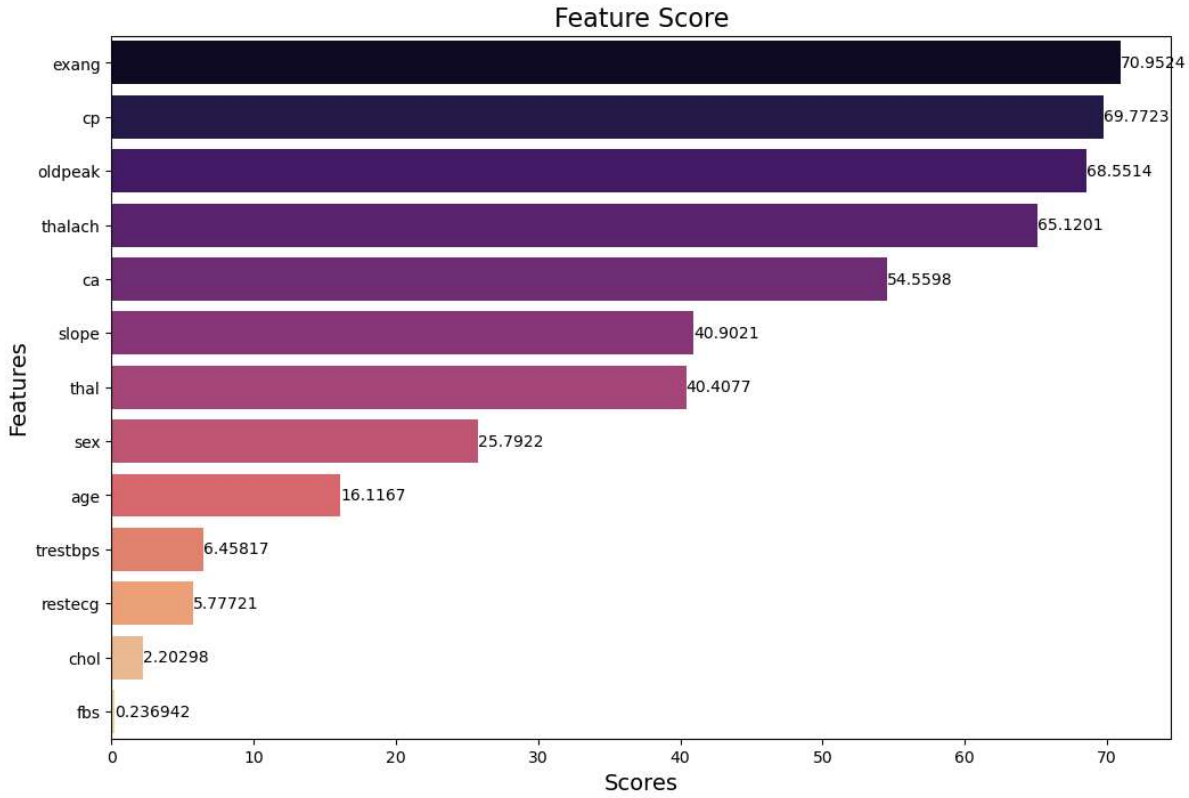
In [15]:
```
#Finding the best features for the model
from sklearn.feature_selection import SelectKBest, f_classif
selector=SelectKBest(f_classif,k=13)
x_selected=selector.fit_transform(x,y)
```

In [18]:
```
#list of all column ans score inx variable
selected_features=x.columns[selector.get_support()]
feature_scores=selector.scores_[selector.get_support()]
```

In [19]:
```
#Creating dataframe for feature and scores
feature_score_df=pd.DataFrame({'Features': selected_features, 'Scores': feature
```

In [21]:
```
#sort the dataframe in descending order
feature_score_df=feature_score_df.sort_values(by='Scores', ascending=False)
```

In [25]:
```
#plotting a barplot for better understanding
plt.figure(figsize=(12,8))
ax=sns.barplot(x=feature_score_df['Scores'],y=feature_score_df['Features'], pal
plt.title('Feature Score', fontsize=16)
plt.xlabel('Scores',fontsize=14)
plt.ylabel('Features',fontsize=14)
for lab in ax.containers:
  ax.bar_label(lab)
```



In [26]:
```
#dropping the low scored values
x=x.drop(['fbs','chol','restecg','trestbps'],axis=1)
```

In [31]:
```
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
x=scaler.fit_transform(x)
```

In [29]:
```
x.head
```

Out[29]:
```
<bound method NDFrame.head of           age       sex        cp   thalach
exang   oldpeak     slope  \
0    0.952197  0.681005  1.973123  0.015443 -0.696631  1.087338 -2.274579
1   -1.915313  0.681005  1.002577  1.633471 -0.696631  2.122573 -2.274579
2   -1.474158 -1.468418  0.032031  0.977514 -0.696631  0.310912  0.976352
3    0.180175  0.681005  0.032031  1.239897 -0.696631 -0.206705  0.976352
4    0.290464 -1.468418 -0.938515  0.583939  1.435481 -0.379244  0.976352
..        ...       ...       ...       ...       ...       ...       ...
298  0.290464 -1.468418 -0.938515 -1.165281  1.435481 -0.724323 -0.649113
299 -1.033002  0.681005  1.973123 -0.771706 -0.696631  0.138373 -0.649113
300  1.503641  0.681005 -0.938515 -0.378132 -0.696631  2.036303 -0.649113
301  0.290464  0.681005 -0.938515 -1.515125  1.435481  0.138373 -0.649113
302  0.290464 -1.468418  0.032031  1.064975 -0.696631 -0.896862 -0.649113

           ca      thal
0   -0.714429 -2.148873
1   -0.714429 -0.512922
2   -0.714429 -0.512922
3   -0.714429 -0.512922
4   -0.714429 -0.512922
..        ...       ...
298 -0.714429  1.123029
299 -0.714429  1.123029
300  1.244593  1.123029
301  0.265082  1.123029
302  0.265082 -0.512922

[303 rows x 9 columns]>
```

In [32]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=4
```

In [37]:
```
from sklearn.neighbors import KNeighborsClassifier
knn=KNeighborsClassifier(n_neighbors=11)
knn.fit(x_train,y_train)
```

Out[37]:
```
KNeighborsClassifier(n_neighbors=11)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [38]:
```
y_predict=knn.predict(x_test)
```

In [39]:
```
from sklearn.metrics import accuracy_score
print(accuracy_score(y_test,y_predict))
```

```
0.9016393442622951
```

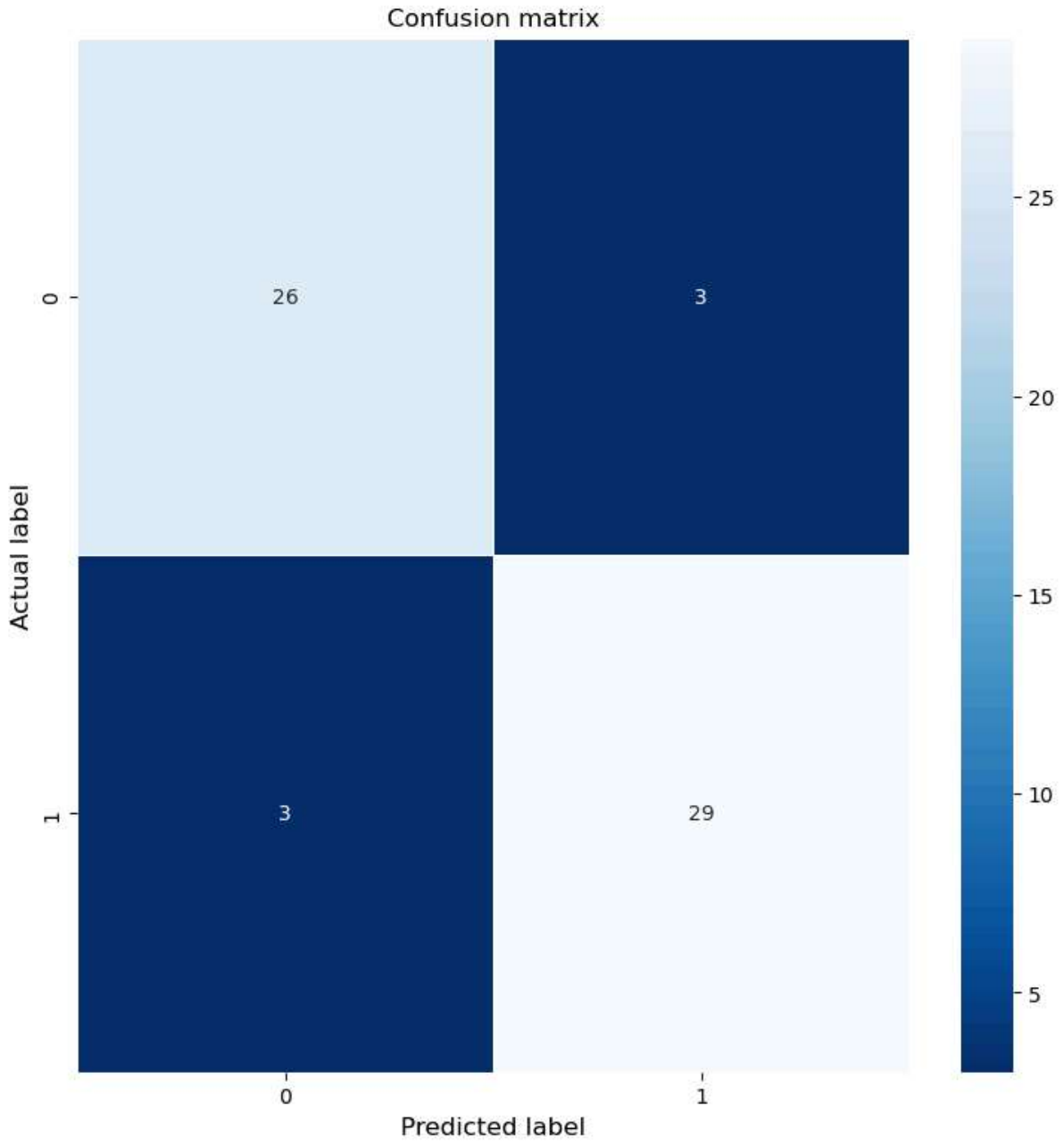
In [40]:
```
from sklearn.metrics import classification_report
print(classification_report(y_test,y_predict))
```

```
              precision    recall  f1-score   support

           0       0.90      0.90      0.90        29
           1       0.91      0.91      0.91        32

    accuracy                           0.90        61
   macro avg       0.90      0.90      0.90        61
weighted avg       0.90      0.90      0.90        61
```

In [42]:
```
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(y_test,y_predict)
print(cm)
```

```
[[26  3]
 [ 3 29]]
```

In [47]:
```
plt.figure(figsize=(9,9))
plt.title('Confusion matrix')
sns.heatmap(cm,annot=True,fmt='g',linewidth=0.5,cmap='Blues_r',)
plt.xlabel('Predicted label',fontsize=12)
plt.ylabel('Actual label',fontsize=12)
plt.show()
```



In [ ]:
```

```