In [138…
```python
#I imported several libraries for the project:
# Basic
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')

# Other libraries
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

# Machine Learning
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
```

In [139…
```python
dataset = pd.read_csv("heart.csv")
dataset
```

Out[139]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	52	1	0	125	212	0	1	168	0	1.0	2	2	3	0
1	53	1	0	140	203	1	0	155	1	3.1	0	0	3	0
2	70	1	0	145	174	0	1	125	1	2.6	0	0	3	0
3	61	1	0	148	203	0	1	161	0	0.0	2	1	3	0
4	62	0	0	138	294	1	1	106	0	1.9	1	3	2	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
298	35	1	1	122	192	0	1	174	0	0.0	2	0	2	1
299	52	1	1	120	325	0	1	172	0	0.2	2	0	2	1
300	46	0	1	105	204	0	1	172	0	0.0	2	0	2	1
301	51	1	2	94	227	0	1	154	1	0.0	2	1	3	1
302	55	0	1	132	342	0	1	166	0	1.2	2	0	2	1

303 rows × 14 columns

In [140…
```python
dataset.isnull().sum()
```

Out[140]:
```
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
```

In [141…
```python
dataset.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 [142…
```python
dataset.tail()
```

Out[142]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
298	35	1	1	122	192	0	1	174	0	0.0	2	0	2	1
299	52	1	1	120	325	0	1	172	0	0.2	2	0	2	1
300	46	0	1	105	204	0	1	172	0	0.0	2	0	2	1
301	51	1	2	94	227	0	1	154	1	0.0	2	1	3	1
302	55	0	1	132	342	0	1	166	0	1.2	2	0	2	1

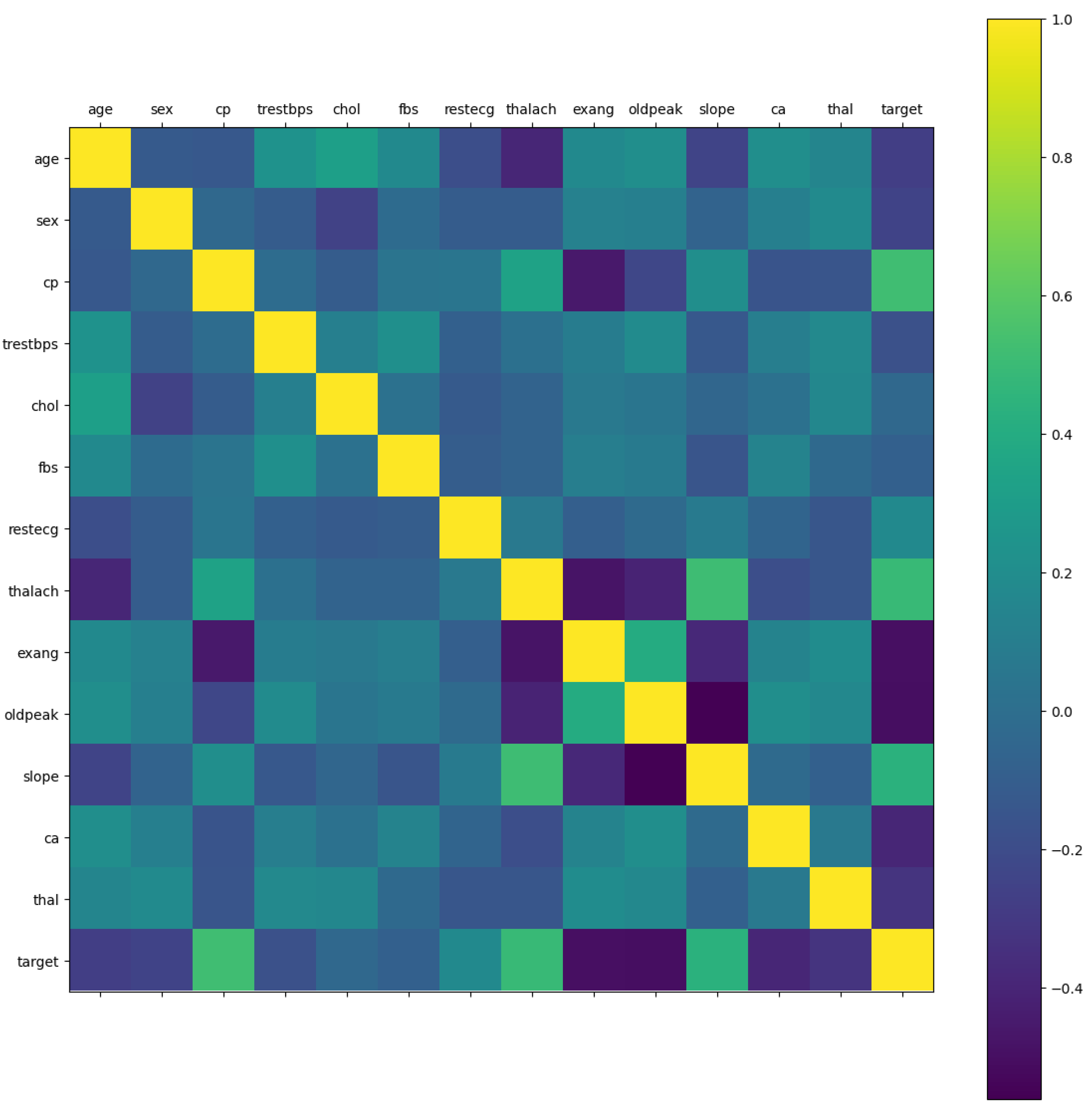
In [143…
```python
dataset.describe()
```

Out[143]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000
mean	53.399340	0.732673	0.953795	131.679868	247.442244	0.165017	0.498350	151.033003	0.343234	1.066337	1.429043	0.696370	2.316832	0.534653
std	9.458638	0.443296	1.012101	18.462235	56.956145	0.371809	0.526607	23.361801	0.475574	1.229059	0.630947	1.006709	0.596858	0.499623
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	46.000000	0.000000	0.000000	120.000000	208.000000	0.000000	0.000000	138.500000	0.000000	0.000000	1.000000	0.000000	2.000000	0.000000
50%	54.000000	1.000000	1.000000	130.000000	240.000000	0.000000	0.000000	154.000000	0.000000	0.800000	2.000000	0.000000	2.000000	1.000000
75%	60.000000	1.000000	2.000000	140.000000	283.000000	0.000000	1.000000	168.500000	1.000000	1.600000	2.000000	1.000000	3.000000	1.000000
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000	202.000000	1.000000	6.200000	2.000000	4.000000	3.000000	1.000000

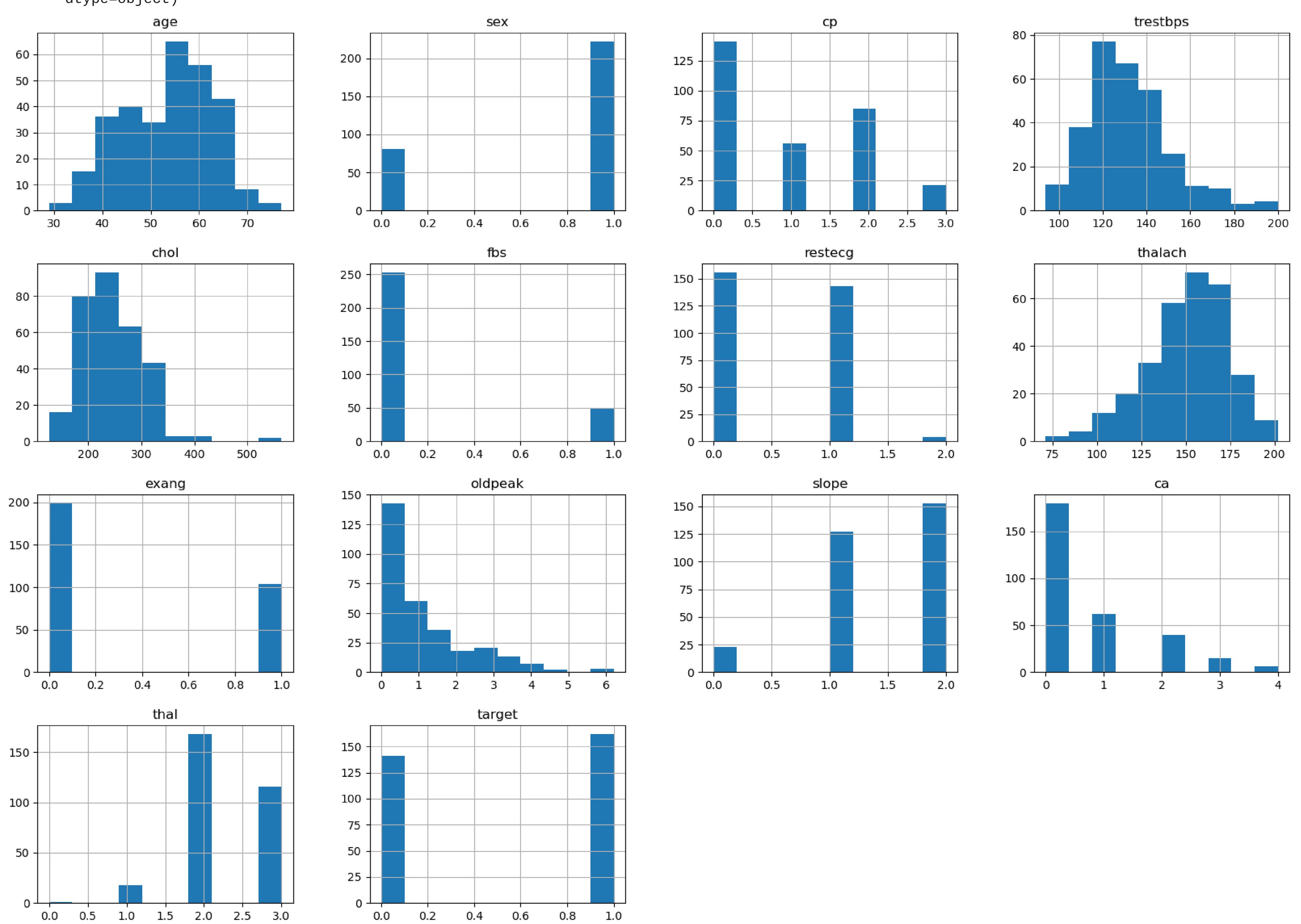
In [144…
```python
rcParams['figure.figsize'] = 20, 14
plt.matshow(dataset.corr())
plt.yticks(np.arange(dataset.shape[1]), dataset.columns)
plt.xticks(np.arange(dataset.shape[1]), dataset.columns)
plt.colorbar()
```

Out[144]:
```
<matplotlib.colorbar.Colorbar at 0x24a3262ab00>
```



In [145…
```python
dataset.hist()#Let’s take a look at the plots. It shows how each feature and label is distributed along different ranges,
```

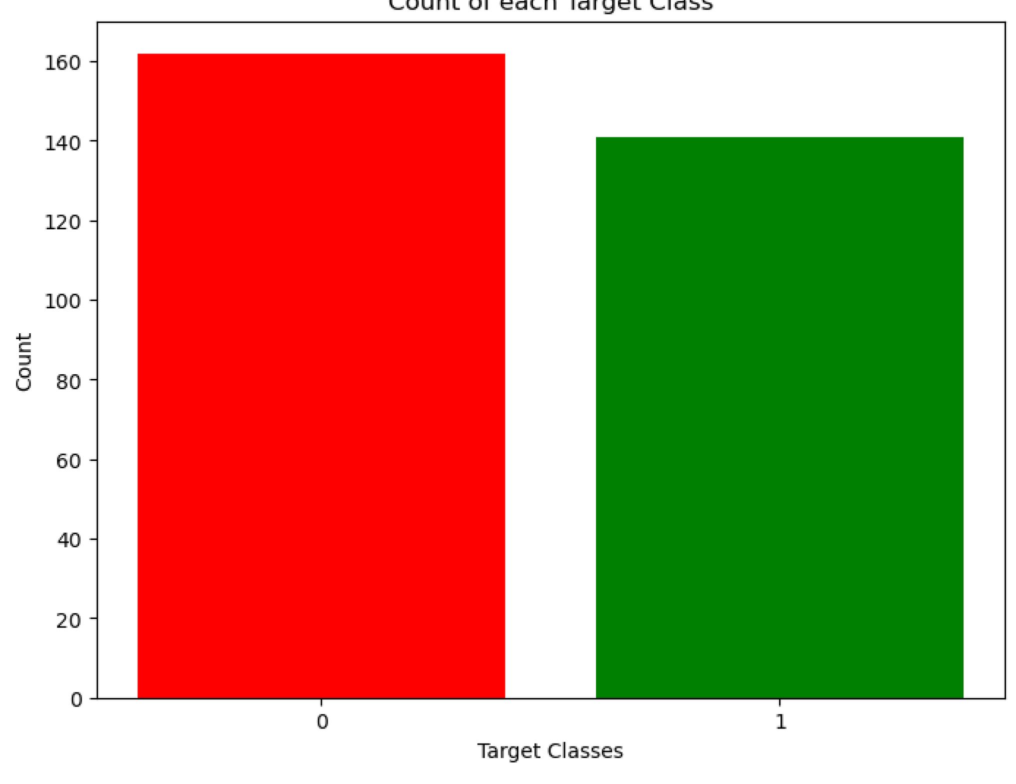
Out[145]:
```
array([[<Axes: title={'center': 'age'}>, <Axes: title={'center': 'sex'}>,
        <Axes: title={'center': 'cp'}>,
        <Axes: title={'center': 'trestbps'}>],
       [<Axes: title={'center': 'chol'}>,
        <Axes: title={'center': 'fbs'}>,
        <Axes: title={'center': 'restecg'}>,
        <Axes: title={'center': 'thalach'}>],
       [<Axes: title={'center': 'exang'}>,
        <Axes: title={'center': 'oldpeak'}>,
        <Axes: title={'center': 'slope'}>,
        <Axes: title={'center': 'ca'}>],
       [<Axes: title={'center': 'thal'}>,
        <Axes: title={'center': 'target'}>, <Axes: >, <Axes: >]],
      dtype=object)
```



In [146…
```python
#Bar Plot for Target Class
#It’s really essential that the dataset we are working on should beapproximately
#balanced. An extremely imbalanced dataset can render the whole model training useless
#and thus, will be of no use. Let’s understand it with an example.
```

In [147…
```python
rcParams['figure.figsize']=8,6
plt.bar(dataset['target'].unique(),dataset['target'].value_counts(),color=['red','green'])
plt.xticks([0,1])
plt.xlabel('Target Classes')
plt.ylabel('Count')
plt.title('Count of each Target Class')
```

Out[147]:
```
Text(0.5, 1.0, 'Count of each Target Class')
```



Data Processing

To work with categorical variables, we should break each categorical column into dummy columns with 1s and 0s.

In [148…
```python
dataset = pd.get_dummies(dataset, columns =
['sex', 'cp', 'fbs', 'restecg', 'exang', 'slope', 'ca', 'thal'])
standardScaler = StandardScaler()
columns_to_scale = ['age', 'trestbps', 'chol', 'thalach', 'oldpeak']
dataset[columns_to_scale] = standardScaler.fit_transform(dataset[columns_to_scale])
```

K Neighbors Classifier

In [159…
```python
y = dataset['target']
X = dataset.drop(['target'], axis = 1)
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.20,random_state=0)
print('Shape of x_train = ',X_train.shape)
print('Shape of y_train = ',y_train.shape)
print('Shape of x_test = ',X_test.shape)
print('Shape of y_train = ',y_test.shape)
```

```
Shape of x_train =  (242, 30)
Shape of y_train =  (242,)
Shape of x_test =  (61, 30)
Shape of y_train =  (61,)
```

In [160…
```python
Classifier=KNeighborsClassifier(n_neighbors = 5)
Classifier.fit(X_train, y_train)
```

Out[160]:
```
▾ KNeighborsClassifier
KNeighborsClassifier()
```

In [161…
```python
Classifier.score(X_test, y_test)
```

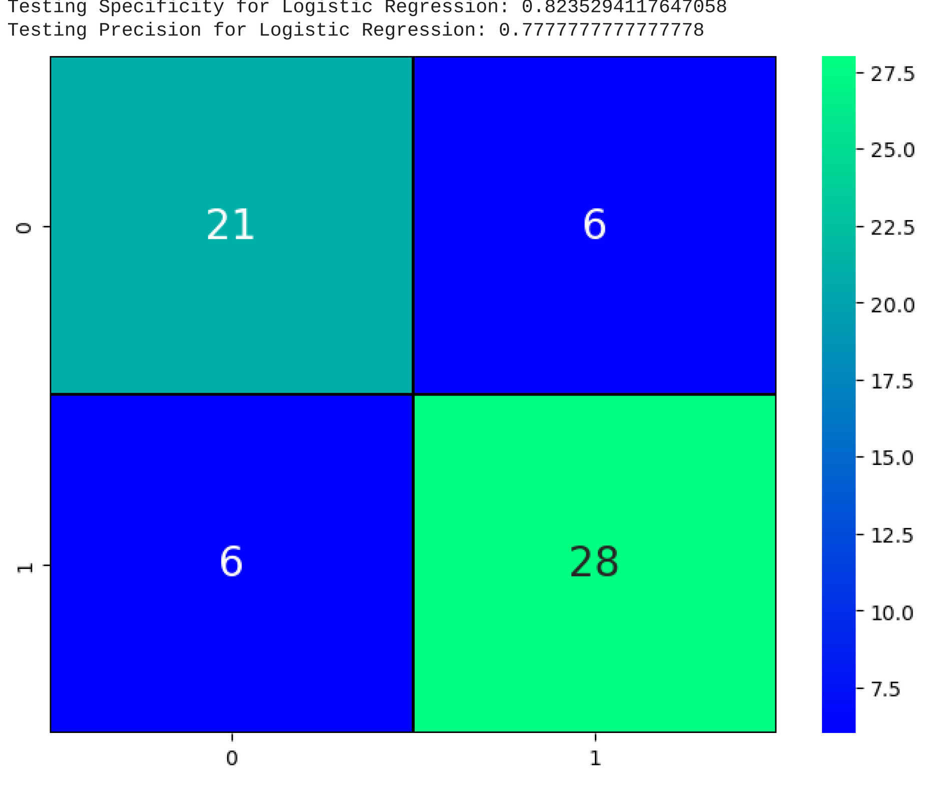
Out[161]:
```
0.8032786885245902
```

In [162…
```python
import seaborn as sns
from sklearn.metrics import confusion_matrix
model1=Classifier.fit(X_train, y_train)
prediction1=model1.predict(X_test)
cm=confusion_matrix(y_test,prediction1)
cm

sns.heatmap(cm, annot=True,cmap='winter',linewidths=0.3, linecolor='black',annot_kws={"size": 20})
TP=cm[0][0]
TN=cm[1][1]
FN=cm[1][0]
FP=cm[0][1]

print('Testing Accuracy for Logistic Regression:',(TP+TN)/(TP+TN+FN+FP))
print('Testing Sensitivity for Logistic Regression:',(TP/(TP+FN)))
print('Testing Specificity for Logistic Regression:',(TN/(TN+FP)))
print('Testing Precision for Logistic Regression:',(TP/(TP+FP)))
```

```
Testing Accuracy for Logistic Regression: 0.8032786885245902
Testing Sensitivity for Logistic Regression: 0.7777777777777778
Testing Specificity for Logistic Regression: 0.8235294117647058
Testing Precision for Logistic Regression: 0.7777777777777778
```



In [163…
```python
#DecisionTreeClassifier
```

In [166…
```python
classifier_entropy=DecisionTreeClassifier(criterion="entropy")
classifier_entropy.fit(X_train,y_train)
```

Out[166]:
```
▾ DecisionTreeClassifier
DecisionTreeClassifier(criterion='entropy')
```

In [167…
```python
classifier_entropy.score(X_test,y_test)
```

Out[167]:
```
0.8524590163934426
```

random forest

In [156…
```python
from sklearn.ensemble import RandomForestClassifier
classifier=RandomForestClassifier(n_estimators=100,criterion='gini')
classifier.fit(X_train,y_train)


classifier.score(X_test,y_test)
```

Out[156]:
```
0.9180327868852459
```

In [ ]:

In [ ]:

In [ ]: