In [2]:
```
import pandas as pd
```

In [3]:
```
heart_dataset=pd.read_csv("heart.csv")
```

In [4]:
```
heart_dataset
```

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
...	...	...	...	...	...	...	...	...	...	...	...	...	...	..
298	57	0	0	140	241	0	1	123	1	0.2	1	0	3	0
299	45	1	3	110	264	0	1	132	0	1.2	1	0	3	0
300	68	1	0	144	193	1	1	141	0	3.4	1	2	3	0
301	57	1	0	130	131	0	1	115	1	1.2	1	1	3	0
302	57	0	1	130	236	0	0	174	0	0.0	1	1	2	0

303 rows × 14 columns

In [5]:
```
# 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
```

Import Dataset

Info()method

In [8]:
```
heart_dataset.info() #Before any analysis, I just wanted to take a look
 at the data. So, I used the info() method.
```

```
<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
```

describe() method

In [9]:
```
heart_dataset.describe()
```

Out[9]:

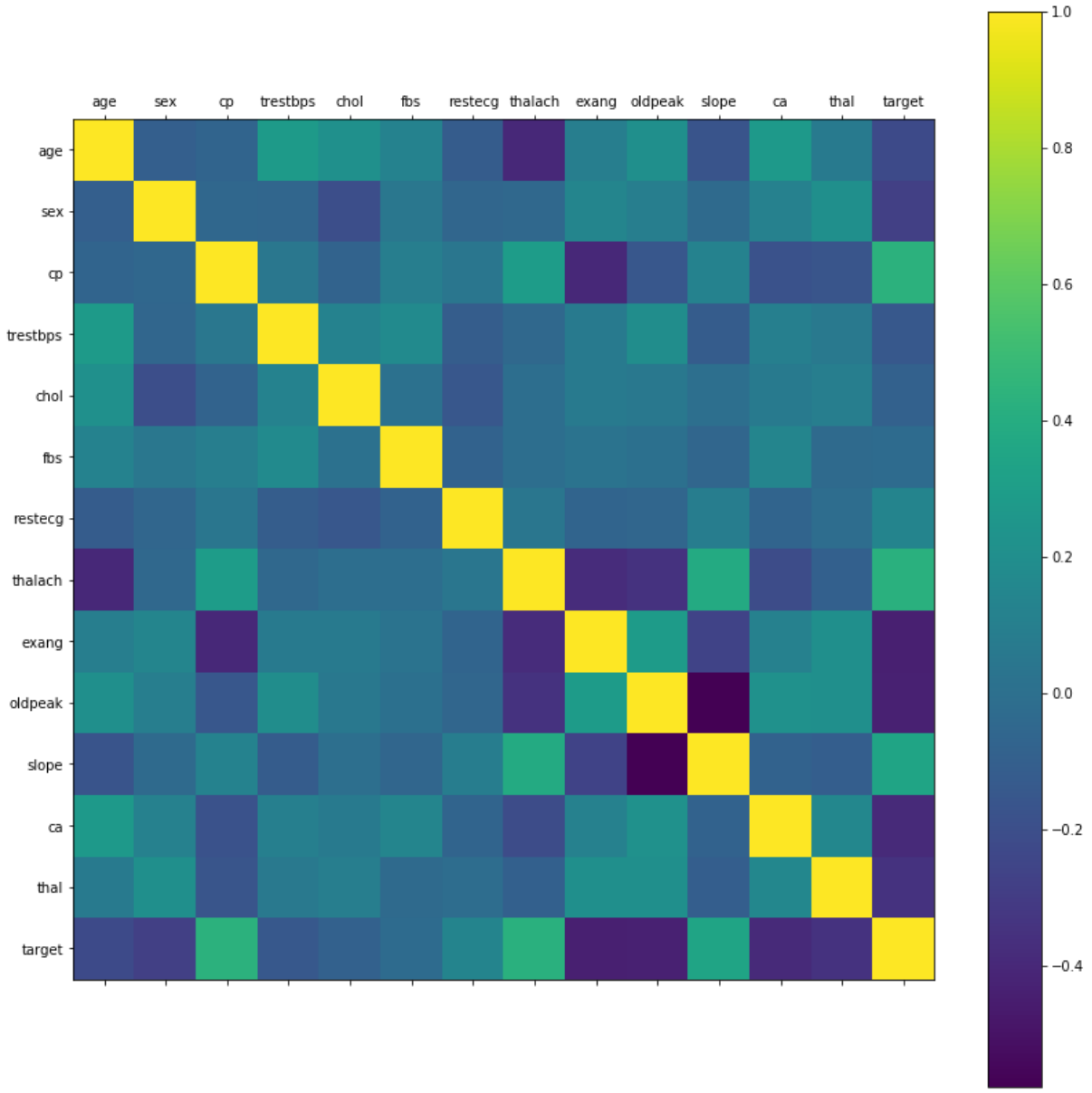
	age	sex	cp	trestbps	chol	fbs	restecg	t
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.(
mean	54.366337	0.683168	0.966997	131.623762	246.264026	0.148515	0.528053	149.6
std	9.082101	0.466011	1.032052	17.538143	51.830751	0.356198	0.525860	22.9
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71.(
25%	47.500000	0.000000	0.000000	120.000000	211.000000	0.000000	0.000000	133.5
50%	55.000000	1.000000	1.000000	130.000000	240.000000	0.000000	1.000000	153.(
75%	61.000000	1.000000	2.000000	140.000000	274.500000	0.000000	1.000000	166.(
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000	202.(

Understanding the data

Correlation Matrix

In [11]:
```
rcParams['figure.figsize'] = 20, 14   #The figure size is defined to 12
 x 8 by using rcParams
plt.matshow(heart_dataset.corr())
plt.yticks(np.arange(heart_dataset.shape[1]), heart_dataset.columns) #U
sing xticks and yticks,
plt.xticks(np.arange(heart_dataset.shape[1]), heart_dataset.columns) #
I've added names to the correlation matrix.
plt.colorbar()  # colorbar() shows the colorbar for the matrix.
```

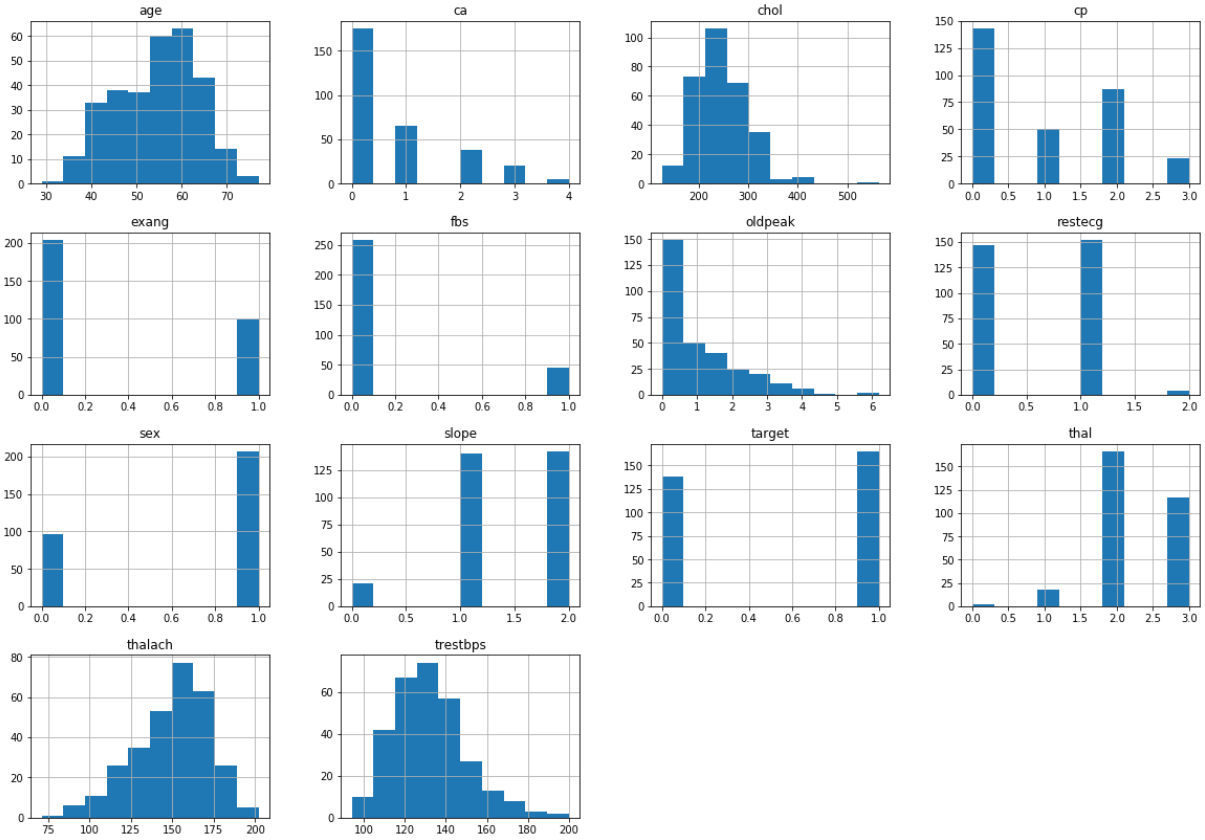
Out[11]: <matplotlib.colorbar.Colorbar at 0x208d0db5748>



Histogram-It just takes a Single Command To Draw the Plots

In [12]:
```
heart_dataset.hist() #It provides so much information in return
```

Out[12]:
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x00000208D1E5
7688>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x00000208D1DE
9C88>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x00000208D1EB
64C8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x00000208D1EE
B248>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x00000208D1F1
AF88>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x00000208D1F5
6C88>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x00000208D1F8
DC48>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x00000208D1FC
4D48>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x00000208D1FD
0E08>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x00000208D201
1048>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x00000208D207
40C8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x00000208D20B
4E88>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x00000208D20E
8308>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x00000208D211
E408>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x00000208D215
7508>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x00000208D219
1708>]],
      dtype=object)
```



Bar Plot for Target Class

In [15]:
```
rcParams['figure.figsize'] = 8,6
plt.bar(heart_dataset['target'].unique(), heart_dataset['target'].value
_counts(), color = ['red', 'green'])
#For x-axis I used the unique() values from the target column and then
 set their name using xticks.
#For y-axis, I used value_count() to get the values for each class. I c
olored the bars as green and red.
plt.xticks([0, 1])
plt.xlabel('Target Classes')
plt.ylabel('Count')
plt.title('Count of each Target Class')
```

Out[15]:
```
Text(0.5, 1.0, 'Count of each Target Class')
```

Data Processing

In [16]:
```
heart_dataset = pd.get_dummies(heart_dataset, columns = ['sex', 'cp',
'fbs', 'restecg', 'exang', 'slope', 'ca', 'thal'])
# we use the get_dummies() method from pandas
standardScaler = StandardScaler()  # we need to scale the dataset for w
hich we will use the StandardScaler
columns_to_scale = ['age', 'trestbps', 'chol', 'thalach', 'oldpeak']
heart_dataset[columns_to_scale] = standardScaler.fit_transform(heart_da
taset[columns_to_scale])
# fit_transform() method of the scaler scales the data and we update th
e columns.
```

In [17]:
```
heart_dataset.head()
```

Out[17]:

	age	trestbps	chol	thalach	oldpeak	target	sex_0	sex_1	cp_0	cp_1	...	sl
0	0.952197	0.763956	-0.256334	0.015443	1.087338	1	0	1	0	0	...	
1	-1.915313	-0.092738	0.072199	1.633471	2.122573	1	0	1	0	0	...	
2	-1.474158	-0.092738	-0.816773	0.977514	0.310912	1	1	0	0	1	...	
3	0.180175	-0.663867	-0.198357	1.239897	-0.206705	1	0	1	0	1	...	
4	0.290464	-0.663867	2.082050	0.583939	-0.379244	1	1	0	1	0	...	

5 rows × 31 columns

In [ ]:
```
#The dataset is now ready. We can begin with training our models.
#I split the dataset into 67% training data and 33% testing data.
```

In [18]:
```
y = heart_dataset['target']
X = heart_dataset.drop(['target'], axis = 1)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size =
0.33, random_state = 0)
```

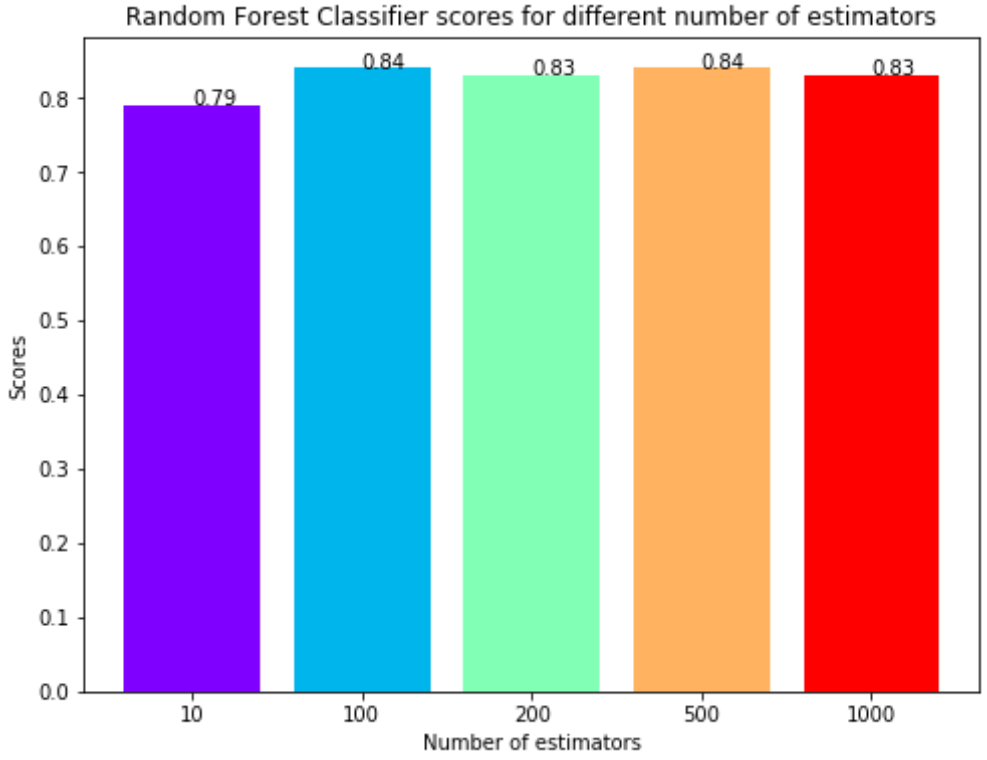
Random Forest Algorithm

In [20]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [21]:
```
rf_scores = []
estimators = [10, 100, 200, 500, 1000]
for i in estimators:
    rf_classifier = RandomForestClassifier(n_estimators = i, random_sta
te = 0)
    rf_classifier.fit(X_train, y_train)
    rf_scores.append(rf_classifier.score(X_test, y_test))
```

In [22]:
```
colors = rainbow(np.linspace(0, 1, len(estimators)))
plt.bar([i for i in range(len(estimators))], rf_scores, color = colors,
 width = 0.8)
for i in range(len(estimators)):
    plt.text(i, rf_scores[i], rf_scores[i])
plt.xticks(ticks = [i for i in range(len(estimators))], labels = [str(e
stimator) for estimator in estimators])
plt.xlabel('Number of estimators')
plt.ylabel('Scores')
plt.title('Random Forest Classifier scores for different number of esti
mators')
```

Out[22]:
```
Text(0.5, 1.0, 'Random Forest Classifier scores for different number of
estimators')
```



conclusion

In [ ]:
```
#Taking a look at the bar graph, we can see that the maximum score of 8
4% was achieved for both 100 and 500 trees.
```