Heart Attack Predictor

In this machine learning project, I have collected the dataset from Kaggle and I will be using Machine Learning to predict whether any person is suffering from heart disease

In [1]:

```
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')
```

In [2]:

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
```

In [3]:

```
df = pd.read_csv('dataset.csv')
```

In [4]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 14 columns):
age         303 non-null int64
sex         303 non-null int64
cp          303 non-null int64
trestbps    303 non-null int64
chol        303 non-null int64
fbs         303 non-null int64
restecg     303 non-null int64
thalach     303 non-null int64
exang       303 non-null int64
oldpeak     303 non-null float64
slope       303 non-null int64
ca          303 non-null int64
thal        303 non-null int64
target      303 non-null int64
dtypes: float64(1), int64(13)
memory usage: 33.2 KB
```

In [5]:

```
df.describe()
```

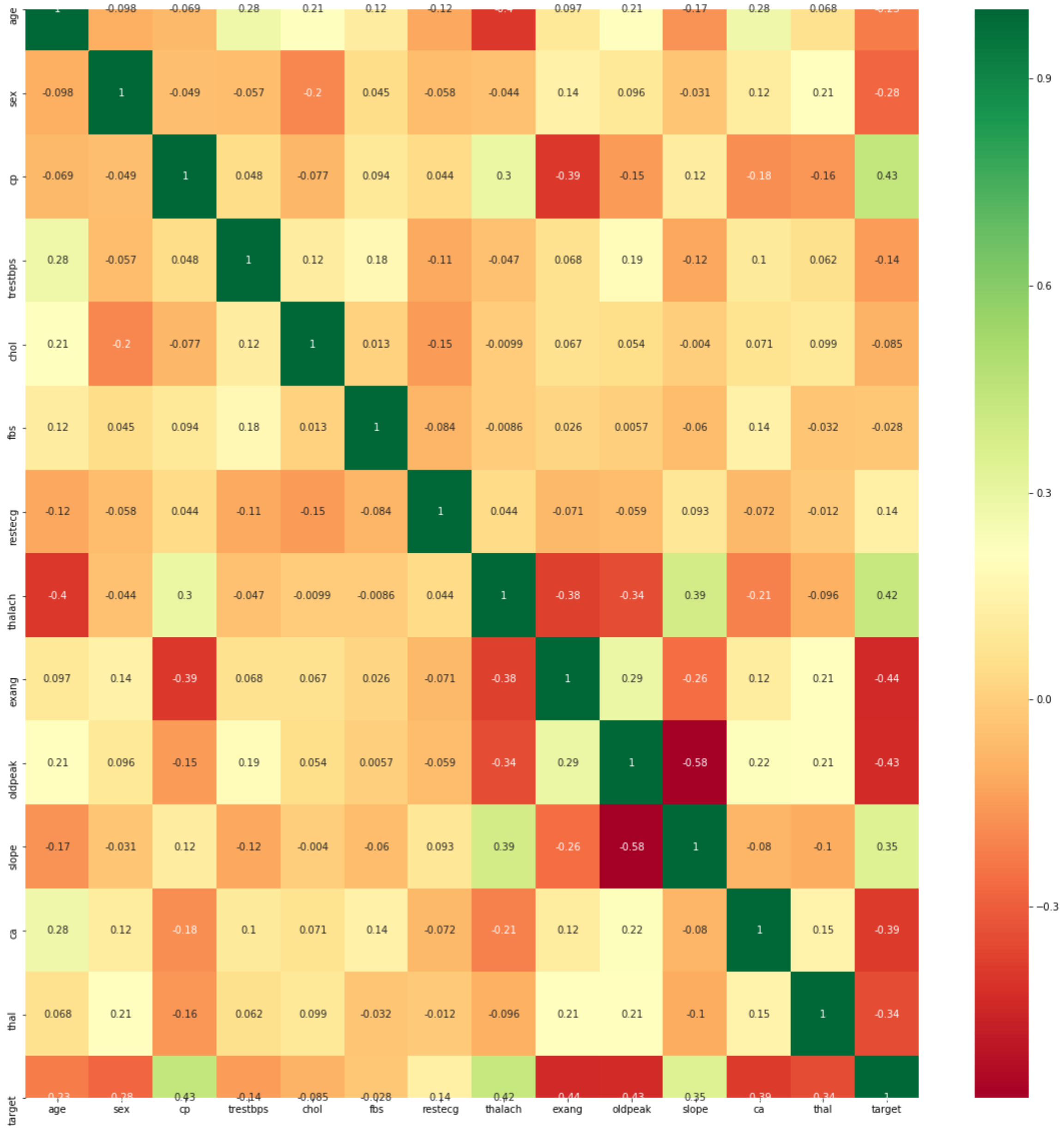
Out[5]:

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

Feature Selection

In [6]:

```
import seaborn as sns
#get correlations of each features in dataset
corrmat = df.corr()
top_corr_features = corrmat.index
plt.figure(figsize=(20,20))
#plot heat map
g=sns.heatmap(df[top_corr_features].corr(),annot=True,cmap="RdYlGn")
```

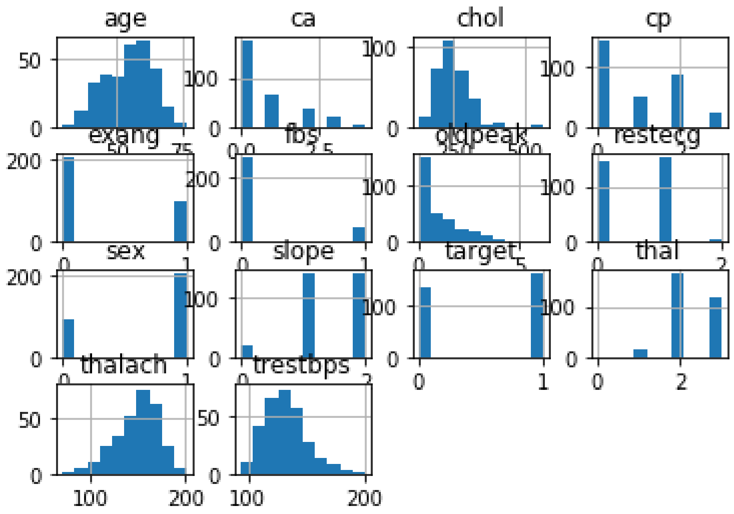

age
sex
cp
trestbps
chol
fbs
restecg
thalach
exang
oldpeak
slope
ca
thal
target
1 -0.098 -0.069 0.28 0.21 0.12 -0.12 -0.4 0.097 0.21 -0.17 0.28 0.068 -0.23
-0.098 1 -0.049 -0.057 -0.2 0.045 -0.058 -0.044 0.14 0.096 -0.031 0.12 0.21 -0.28
-0.069 -0.049 1 0.048 -0.077 0.094 0.044 0.3 -0.39 -0.15 0.12 -0.18 -0.16 0.43
0.28 -0.057 0.048 1 0.12 0.18 -0.11 -0.047 0.068 0.19 -0.12 0.1 0.062 -0.14
0.21 -0.2 -0.077 0.12 1 0.013 -0.15 -0.0099 0.067 0.054 -0.004 0.071 0.099 -0.085
0.12 0.045 0.094 0.18 0.013 1 -0.084 -0.0086 0.026 0.0057 -0.06 0.14 -0.032 -0.028
-0.12 -0.058 0.044 -0.11 -0.15 -0.084 1 0.044 -0.071 -0.059 0.093 -0.072 -0.012 0.14
-0.4 -0.044 0.3 -0.047 -0.0099 -0.0086 0.044 1 -0.38 -0.34 0.39 -0.21 -0.096 0.42
0.097 0.14 -0.39 0.068 0.067 0.026 -0.071 -0.38 1 0.29 -0.26 0.12 0.21 -0.44
0.21 0.096 -0.15 0.19 0.054 0.0057 -0.059 -0.34 0.29 1 -0.58 0.22 0.21 -0.43
-0.17 -0.031 0.12 -0.12 -0.004 -0.06 0.093 0.39 -0.26 -0.58 1 -0.08 -0.1 0.35
0.28 0.12 -0.18 0.1 0.071 0.14 -0.072 -0.21 0.12 0.22 -0.08 1 0.15 -0.39
0.068 0.21 -0.16 0.062 0.099 -0.032 -0.012 -0.096 0.21 0.21 -0.1 0.15 1 -0.34
-0.23 -0.28 0.43 -0.14 -0.085 -0.028 0.14 0.42 -0.44 -0.43 0.35 -0.39 -0.34 1
0.9
0.6
0.3
0.0
−0.3

In [7]:

```
df.hist()
```

Out[7]:

```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x0E7DAC90>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0E7F5E90>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0E80CF70>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0E5885F0>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x0E5A86D0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0E5CA7B0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0E5EB910>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0E60A3D0>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x0E60A970>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0E62BB10>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0E66CC10>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0E68FCF0>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x0ECCCDD0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0ECF0EB0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0ED0FF90>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0ED30AD0>]],
      dtype=object)
```



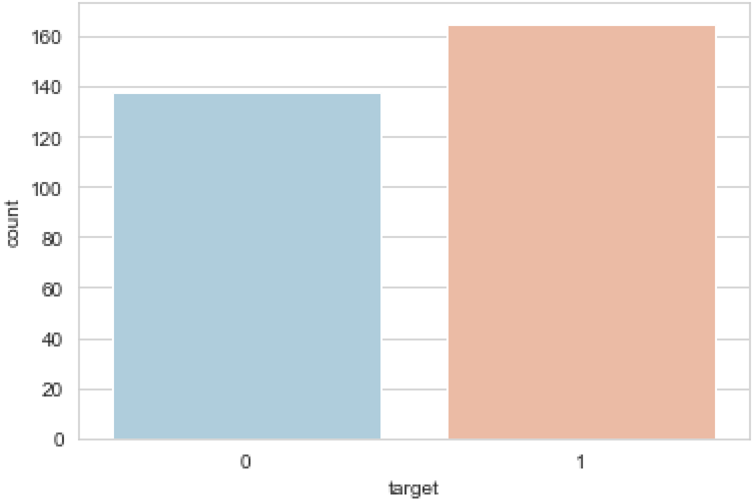
It's always a good practice to work with a dataset where the target classes are of approximately equal size. Thus, let's check for the same.

In [8]:

```
sns.set_style('whitegrid')
sns.countplot(x='target',data=df,palette='RdBu_r')
```

Out[8]:

```
<matplotlib.axes._subplots.AxesSubplot at 0xee6fc90>
```



Data Processing

After exploring the dataset, I observed that I need to convert some categorical variables into dummy variables and scale all the values before training the Machine Learning models. First, I'll use the `get_dummies` method to create dummy columns for categorical variables.

In [9]:

```
dataset = pd.get_dummies(df, columns = ['sex', 'cp', 'fbs', 'restecg', 'exang', 'slope'
, 'ca', 'thal'])
```

In [10]:

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
standardScaler = StandardScaler()
columns_to_scale = ['age', 'trestbps', 'chol', 'thalach', 'oldpeak']
dataset[columns_to_scale] = standardScaler.fit_transform(dataset[columns_to_scale])
```

In [11]:

```
dataset.head()
```

Out[11]:

	age	trestbps	chol	thalach	oldpeak	target	sex_0	sex_1	cp_0	cp_1	..
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 [12]:

```
y = dataset['target']
X = dataset.drop(['target'], axis = 1)
```

In [13]:

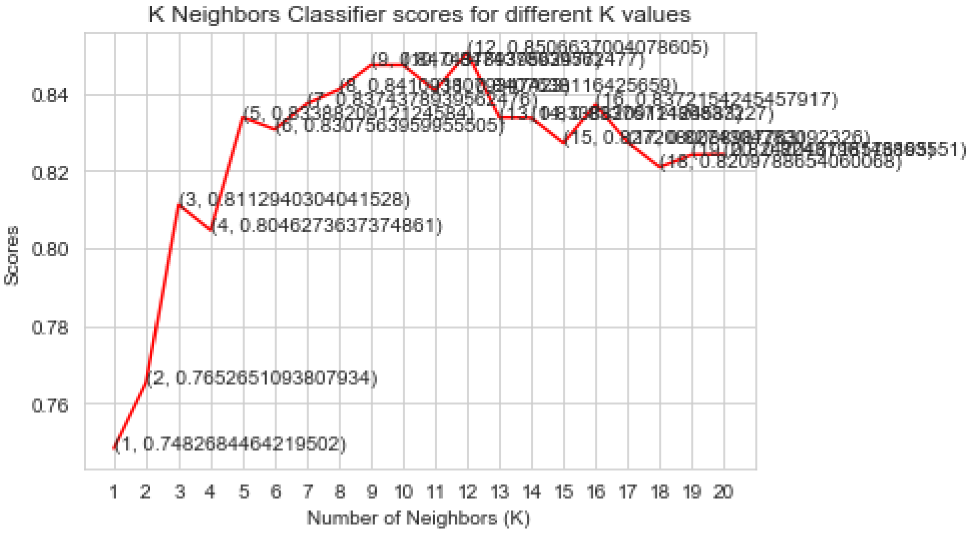
```
from sklearn.model_selection import cross_val_score
knn_scores = []
for k in range(1,21):
    knn_classifier = KNeighborsClassifier(n_neighbors = k)
    score=cross_val_score(knn_classifier,X,y,cv=10)
    knn_scores.append(score.mean())
```

In [14]:

```
plt.plot([k for k in range(1, 21)], knn_scores, color = 'red')
for i in range(1,21):
    plt.text(i, knn_scores[i-1], (i, knn_scores[i-1]))
plt.xticks([i for i in range(1, 21)])
plt.xlabel('Number of Neighbors (K)')
plt.ylabel('Scores')
plt.title('K Neighbors Classifier scores for different K values')
```

Out[14]:

```
Text(0.5, 1.0, 'K Neighbors Classifier scores for different K values')
```



In [15]:

```
knn_classifier = KNeighborsClassifier(n_neighbors = 12)
score=cross_val_score(knn_classifier,X,y,cv=10)
```

In [16]:

```
score.mean()
```

Out[16]:

```
0.8506637004078605
```

Random Forest Classifier

In [17]:

```
from sklearn.ensemble import RandomForestClassifier
```

In [18]:

```
randomforest_classifier= RandomForestClassifier(n_estimators=10)

score=cross_val_score(randomforest_classifier,X,y,cv=10)
```

In [19]:

```
score.mean()
```

Out[19]:

```
0.8144197256210605
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: