**CODE OF HEART DISEASE PREDICTION USING MACHINE LEARNING AND DATA ANALYSIS APPROACH :-**

**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')

Here we will be experimenting with 3 algorithms

1. KNeighborsClassifier
2. DecisionTreeClassifier
3. RandomForestClassifier

**from** sklearn.neighbors **import** KNeighborsClassifier

**from** sklearn.tree **import** DecisionTreeClassifier

**from** sklearn.ensemble **import** RandomForestClassifier

df **=** pd**.**read\_csv('dataset.csv')

df**.**info()

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

df**.**describe()

|  | **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** | 54.366337 | 0.683168 | 0.966997 | 131.623762 | 246.264026 | 0.148515 | 0.528053 | 149.646865 | 0.326733 | 1.039604 | 1.399340 | 0.729373 | 2.313531 | 0.544554 |
| **std** | 9.082101 | 0.466011 | 1.032052 | 17.538143 | 51.830751 | 0.356198 | 0.525860 | 22.905161 | 0.469794 | 1.161075 | 0.616226 | 1.022606 | 0.612277 | 0.498835 |
| **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%** | 47.500000 | 0.000000 | 0.000000 | 120.000000 | 211.000000 | 0.000000 | 0.000000 | 133.500000 | 0.000000 | 0.000000 | 1.000000 | 0.000000 | 2.000000 | 0.000000 |
| **50%** | 55.000000 | 1.000000 | 1.000000 | 130.000000 | 240.000000 | 0.000000 | 1.000000 | 153.000000 | 0.000000 | 0.800000 | 1.000000 | 0.000000 | 2.000000 | 1.000000 |
| **75%** | 61.000000 | 1.000000 | 2.000000 | 140.000000 | 274.500000 | 0.000000 | 1.000000 | 166.000000 | 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 |

## **Feature Selection**

**import** seaborn **as** sns

*#get correlations of each features in dataset*

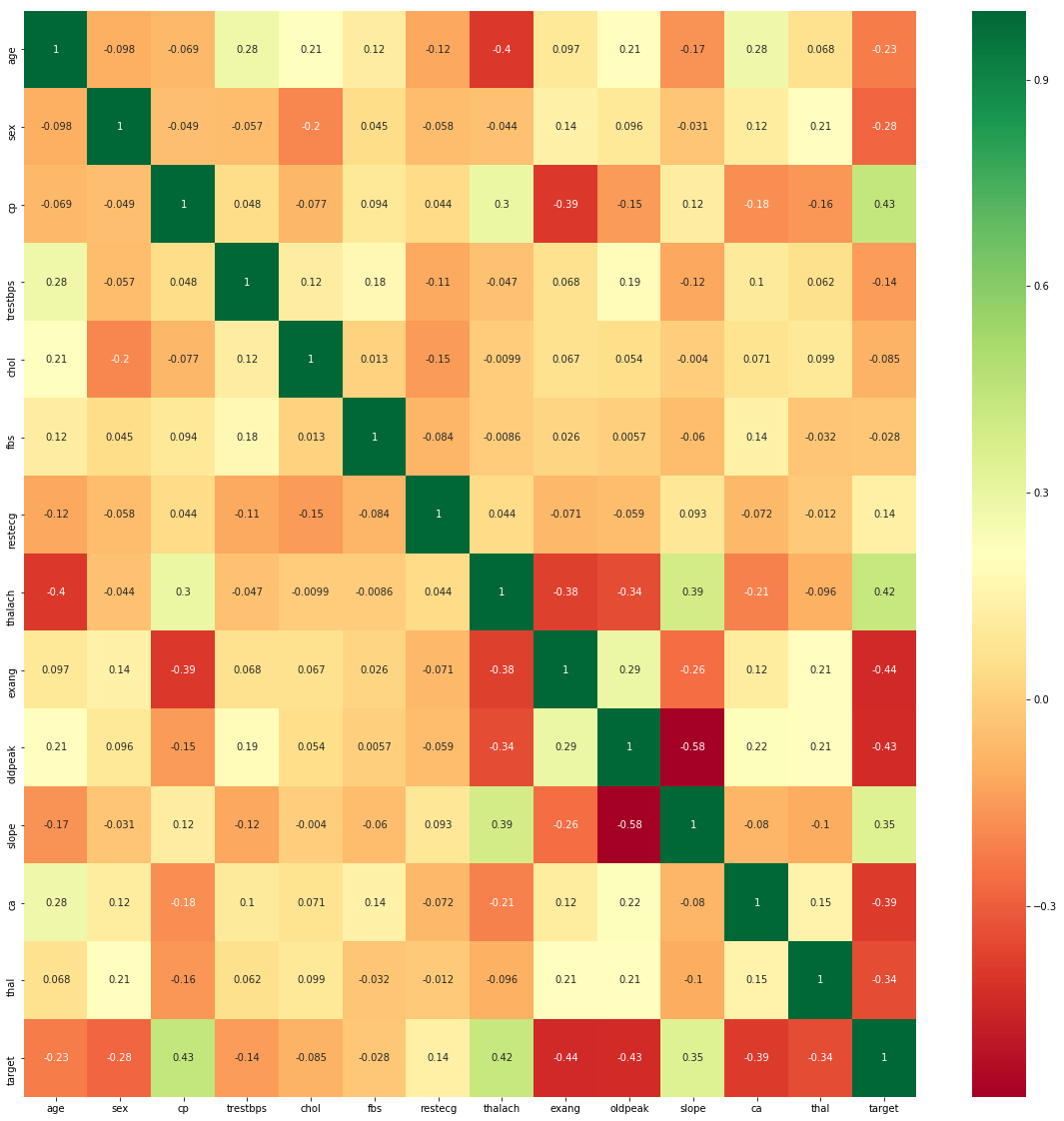
corrmat **=** df**.**corr()

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")



In [14]:

df**.**hist()

array([[,

,

,

],

[,

,

,

],

[,

,

,

],

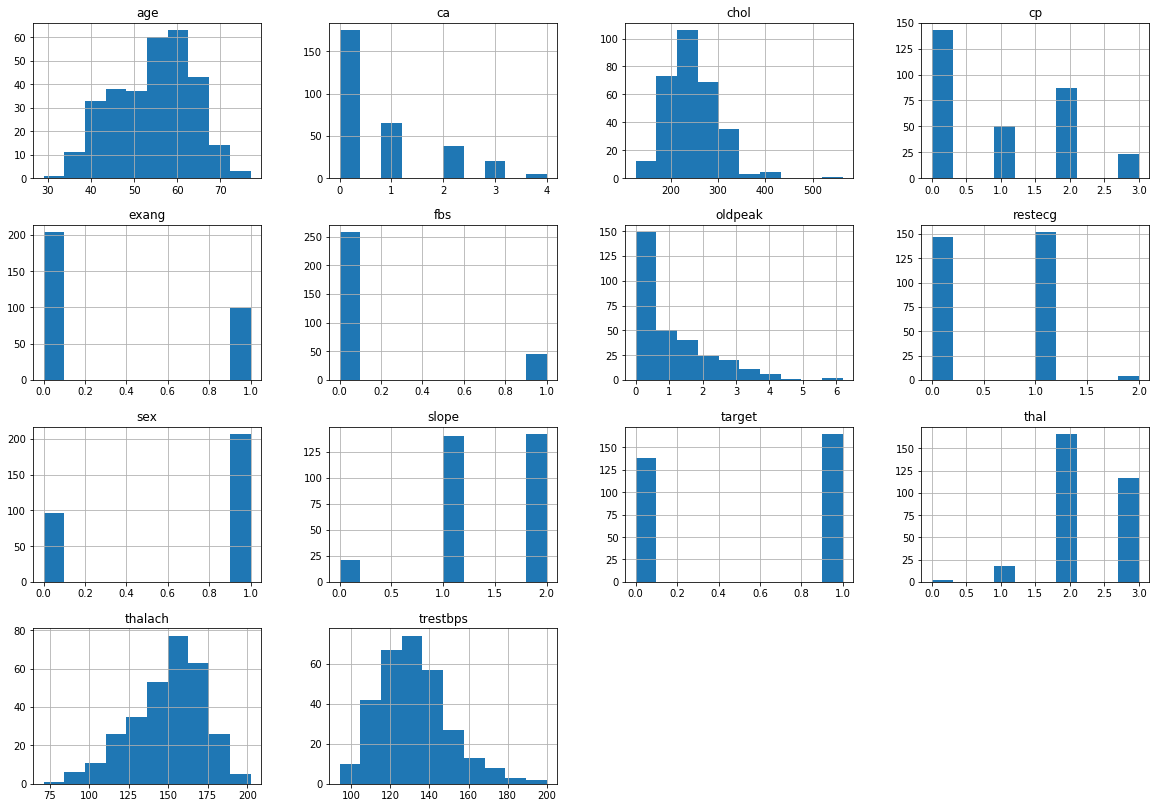
[,

,

,

]],

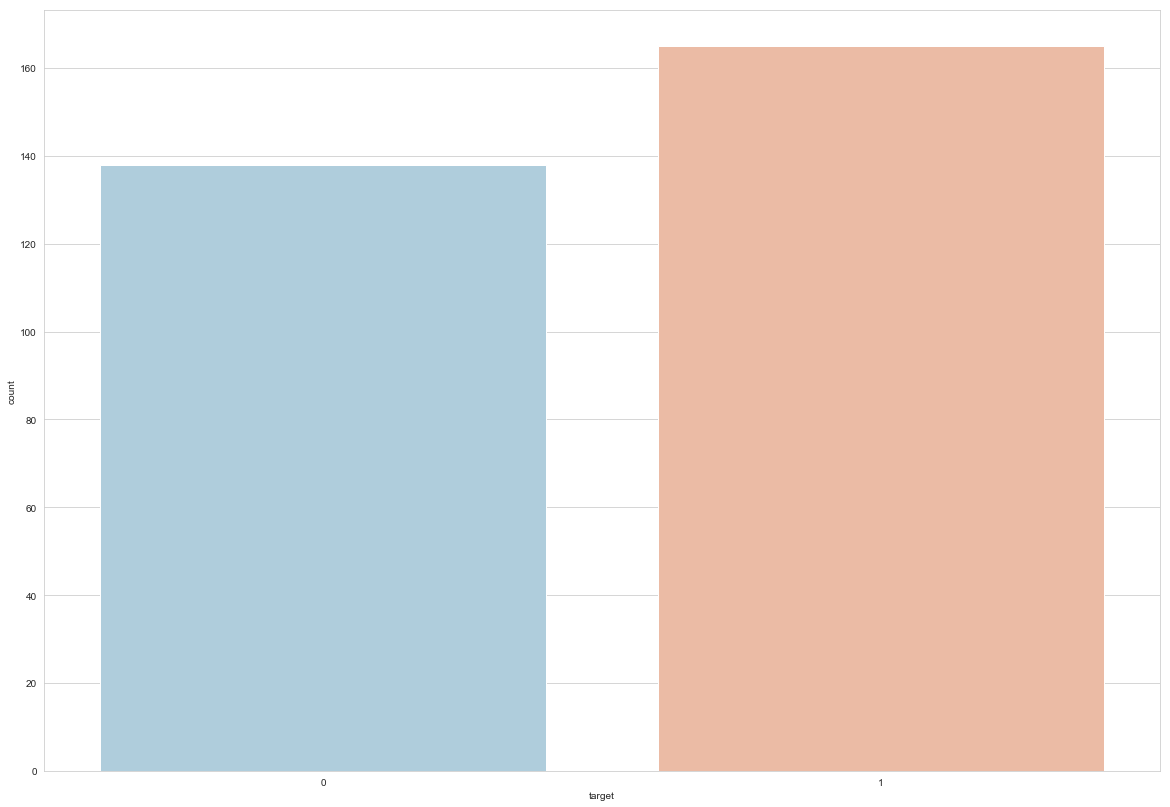
dtype=object)



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.

sns**.**set\_style('whitegrid')

sns**.**countplot(x**=**'target',data**=**df,palette**=**'RdBu\_r')



### **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.

dataset **=** pd**.**get\_dummies(df, columns **=** ['sex', 'cp', 'fbs', 'restecg', 'exang', 'slope', 'ca', 'thal'])

**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])

C:\Users\krish.naik\AppData\Local\Continuum\anaconda3\envs\myenv\lib\site-packages\sklearn\preprocessing\data.py:625: DataConversionWarning: Data with input dtype int64, float64 were all converted to float64 by StandardScaler.

return self.partial\_fit(X, y)

C:\Users\krish.naik\AppData\Local\Continuum\anaconda3\envs\myenv\lib\site-packages\sklearn\base.py:462: DataConversionWarning: Data with input dtype int64, float64 were all converted to float64 by StandardScaler.

return self.fit(X, \*\*fit\_params).transform(X)

dataset**.**head()

|  | **age** | **trestbps** | **chol** | **thalach** | **oldpeak** | **target** | **sex\_0** | **sex\_1** | **cp\_0** | **cp\_1** | **...** | **slope\_2** | **ca\_0** | **ca\_1** | **ca\_2** | **ca\_3** | **ca\_4** | **thal\_0** | **thal\_1** | **thal\_2** | **thal\_3** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 0.952197 | 0.763956 | -0.256334 | 0.015443 | 1.087338 | 1 | 0 | 1 | 0 | 0 | ... | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| **1** | -1.915313 | -0.092738 | 0.072199 | 1.633471 | 2.122573 | 1 | 0 | 1 | 0 | 0 | ... | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| **2** | -1.474158 | -0.092738 | -0.816773 | 0.977514 | 0.310912 | 1 | 1 | 0 | 0 | 1 | ... | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| **3** | 0.180175 | -0.663867 | -0.198357 | 1.239897 | -0.206705 | 1 | 0 | 1 | 0 | 1 | ... | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| **4** | 0.290464 | -0.663867 | 2.082050 | 0.583939 | -0.379244 | 1 | 1 | 0 | 1 | 0 | ... | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |

5 rows × 31 columns

y **=** dataset['target']

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

**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())

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]))

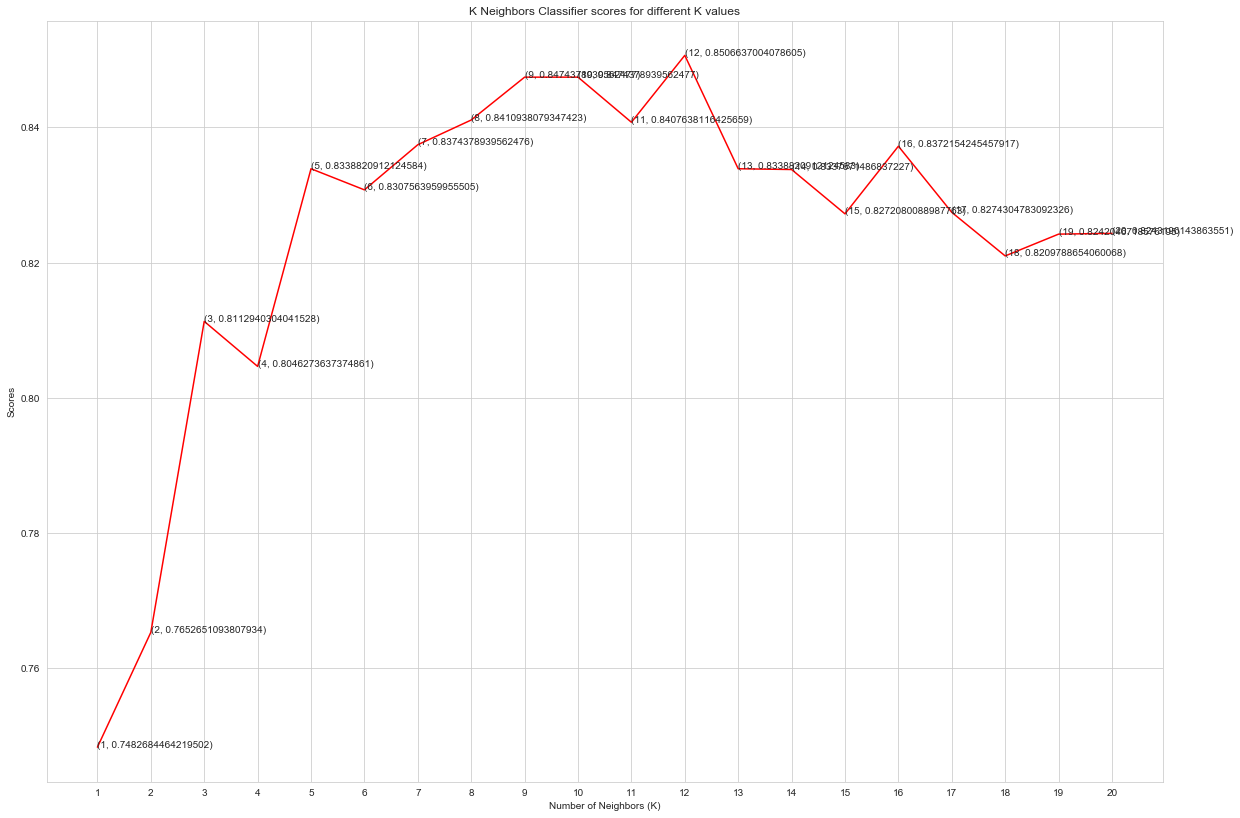
plt**.**xticks([i **for** i **in** range(1, 21)])

plt**.**xlabel('Number of Neighbors (K)')

plt**.**ylabel('Scores')

plt**.**title('K Neighbors Classifier scores for different K values')

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



knn\_classifier **=** KNeighborsClassifier(n\_neighbors **=** 12)

score**=**cross\_val\_score(knn\_classifier,X,y,cv**=**10)

score**.**mean()

0.9892506637004078605

## **Random Forest Classifier**

**from** sklearn.ensemble **import** RandomForestClassifier

randomforest\_classifier**=** RandomForestClassifier(n\_estimators**=**10)

score**=**cross\_val\_score(randomforest\_classifier,X,y,cv**=**10)

score**.**mean()

0.8798887645294772