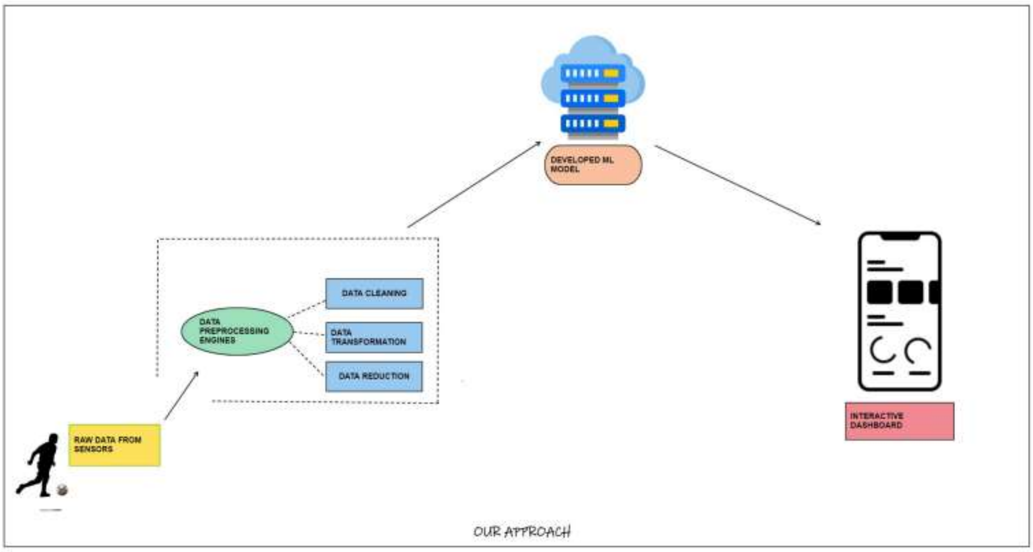
TEAM ID: IBM-Project-25390-1659961381

Heart Disease Prediction

Machine learning is used to diagnose, detect, and forecast many disorders in the medical industry. The primary purpose of this study is to give clinicians a tool to detect cardiac problems at an early stage. As a result, it will be easier to deliver appropriate treatment to patients while avoiding serious effects.In the system of the human heart, the heart's electrical activity is recorded by ECG with various wave forms through skin.electrodes. Our approach begins with acquiring raw data through various sensors attached to the human body, let's say a fitness tracker watch ,the raw data is sent to the preprocessing engine where the data is structured and ready for further processing.The processed data is sent to the developed model for prediction ,after which the results are displayed in a interactive dashboard where the users can keep track of their body status.

In this machine learning project, I have collected the dataset from Kaggle (https://www.kaggle.com/ronitf/heart-disease-uci) and I will be using Machine Learning to predict whether any person is suffering from heart disease



Import libraries

In [25]:
```
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

In [5]:
```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
```

Import dataset

Now that we have all the libraries we will need, we can import the dataset and take a look at it. The dataset is stored in the file dataset.csv . We'll use the pandas read_csv method to read the dataset.

In [6]:
```
df = pd.read_csv('dataset.csv')
```

In [7]:
```
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 [8]:
```
df.describe()
```

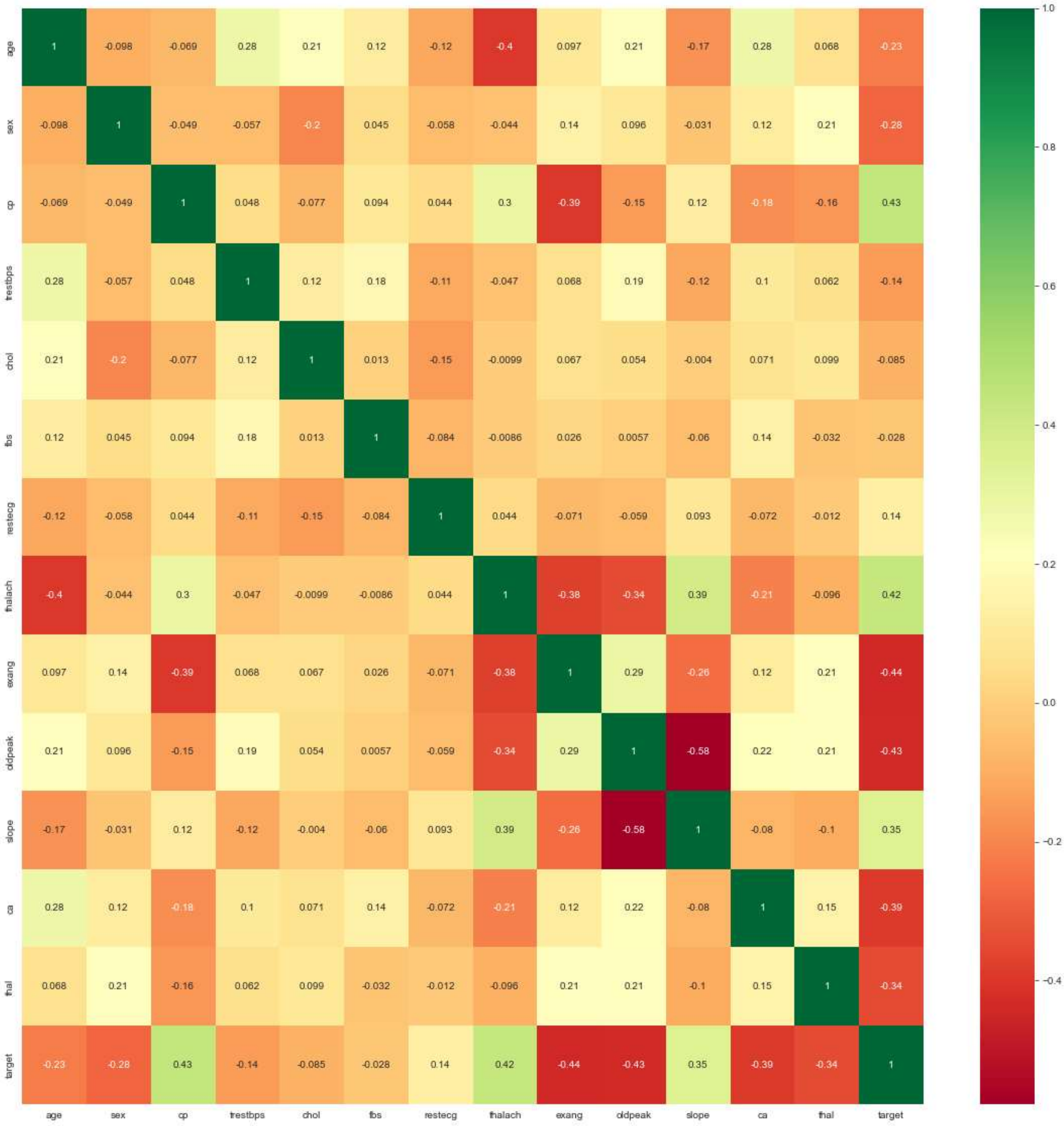
Out[8]:

	age	sex	cp	trestbps	chol	fbs	restecg	th
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.0C
mean	54.366337	0.683168	0.966997	131.623762	246.264026	0.148515	0.528053	149.64
std	9.082101	0.466011	1.032052	17.538143	51.830751	0.356198	0.525860	22.9C
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71.0C
25%	47.500000	0.000000	0.000000	120.000000	211.000000	0.000000	0.000000	133.5C
50%	55.000000	1.000000	1.000000	130.000000	240.000000	0.000000	1.000000	153.0C
75%	61.000000	1.000000	2.000000	140.000000	274.500000	0.000000	1.000000	166.0C
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000	202.0C

Feature Selection

In [24]:

```
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")
```

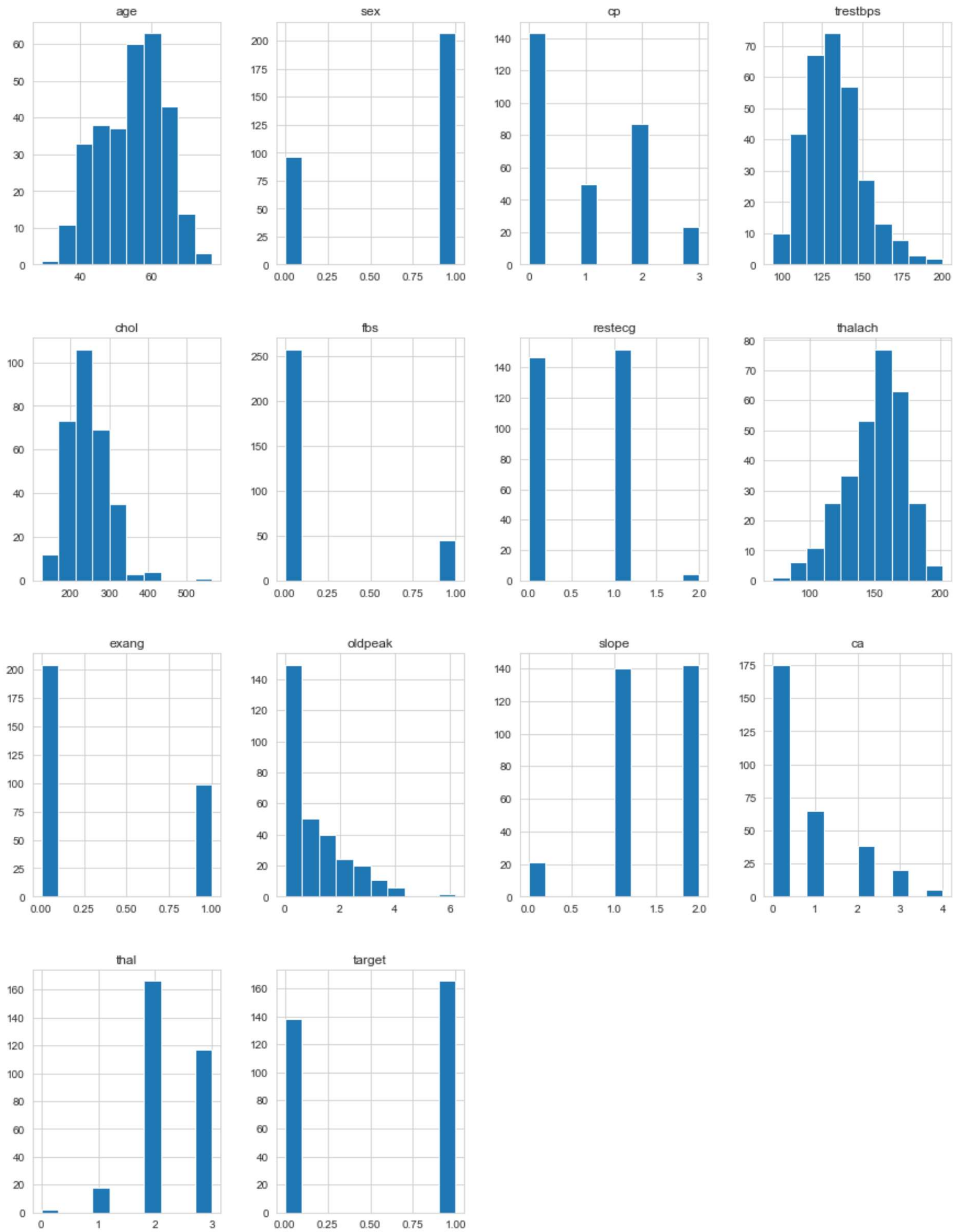


In [28]:

```
df.hist(figsize = (15,20))
```

Out[28]:

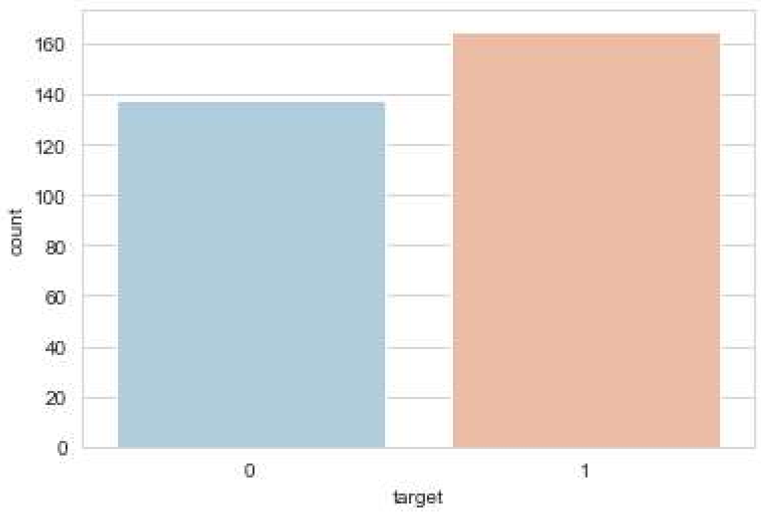
```
array([[<AxesSubplot:title={'center':'age'}>,
        <AxesSubplot:title={'center':'sex'}>,
        <AxesSubplot:title={'center':'cp'}>,
        <AxesSubplot:title={'center':'trestbps'}>],
       [<AxesSubplot:title={'center':'chol'}>,
        <AxesSubplot:title={'center':'fbs'}>,
        <AxesSubplot:title={'center':'restecg'}>,
        <AxesSubplot:title={'center':'thalach'}>],
       [<AxesSubplot:title={'center':'exang'}>,
        <AxesSubplot:title={'center':'oldpeak'}>,
        <AxesSubplot:title={'center':'slope'}>,
        <AxesSubplot:title={'center':'ca'}>],
       [<AxesSubplot:title={'center':'thal'}>,
        <AxesSubplot:title={'center':'target'}>, <AxesSubplot:>,
        <AxesSubplot:>]], 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 [11]:
```
sns.set_style('whitegrid')
sns.countplot(x='target',data=df,palette='RdBu_r')
```

Out[11]:
```
<AxesSubplot:xlabel='target', ylabel='count'>
```



#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 [12]:
```
dataset = pd.get_dummies(df, columns = ['sex', 'cp', 'fbs', 'restecg', 'exang', 's
```

In [13]:
```
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 [14]:
```
dataset.head()
```

Out[14]:

	age	trestbps	chol	thalach	oldpeak	target	sex_0	sex_1	cp_0	cp_1	...	slo
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	...	

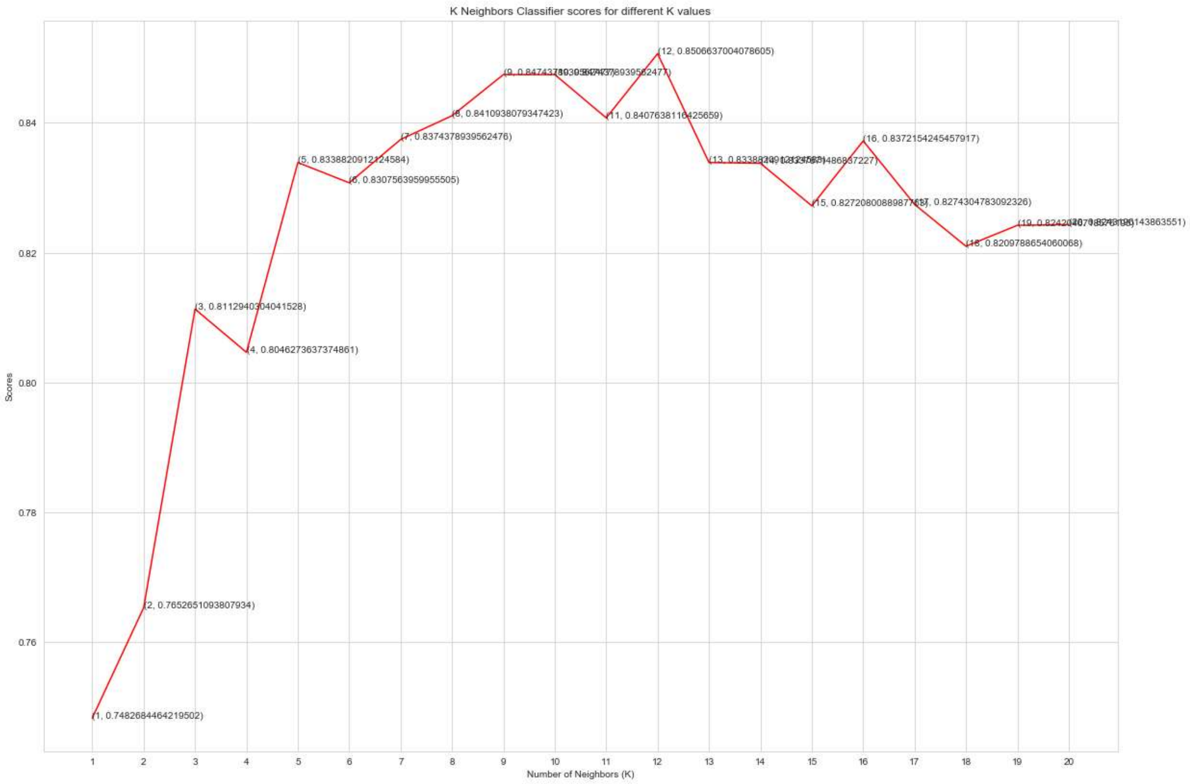
5 rows × 31 columns

In [15]:
```
y = dataset['target']
X = dataset.drop(['target'], axis = 1)
```

In [16]:
```
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 [26]:
```
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[26]:
```
Text(0.5, 1.0, 'K Neighbors Classifier scores for different K values')
```



In [17]:
```
knn_classifier = KNeighborsClassifier(n_neighbors = 12)
score=cross_val_score(knn_classifier,X,y,cv=10)
```

In [18]:
```
score.mean()
```

Out[18]:
```
0.8448387096774195
```

Random Forest Classifier

In [19]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [20]:
```
randomforest_classifier= RandomForestClassifier(n_estimators=10)

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

In [21]:
```
score.mean()
```

Out[21]:
```
0.7981720430107526
```

In [22]:
```
score.mean()
```

Out[22]:
```
0.7981720430107526
```

In [ ]:
```

```