In this Homework we are asked to research the Exteme Gradient Boosting (XGBoost)

Team Members are: Youssouf Diallo and Walter Diong

1. Importing Libraries

In [2]:
```
import xgboost as xgb
from xgboost import XGBClassifier
import pandas as pd
import numpy as np
import seaborn as sn
import matplotlib.pyplot as plt
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
```

2. Importing the dataset into a DataFrame

we will be using XGBoost as classifier to classify if someone will have heart disease

the data set is dowloaded from Kaggle at this link https://www.kaggle.com/ronitf/heart-disease-uci (https://www.kaggle.com/ronitf/heart-disease-uci)

In [3]:
```
## loading the data
heath_data =pd.read_csv('HW7_XGBoost_datafile.csv')
```

In [4]:
```
## let 's look at the size of the data set
heath_data.shape
```

Out[4]: (303, 14)

we have 303 records and 14 features

3. Data set Description

In [5]:
```
## print the features
heath_data.columns
```

Out[5]:
```
Index(['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach',
       'exang', 'oldpeak', 'slope', 'ca', 'thal', 'target'],
      dtype='object')
```

this is a description of the features . 1 means the first feature and so

1. age

2. sex

3. (CP) chest pain type (4 values)

4. resting blood pressure

5. serum cholestoral in mg/dl

6. fasting blood sugar > 120 mg/dl

7. resting electrocardiographic results (values 0,1,2)

8. maximum heart rate achieved

9. exercise induced angina

10. oldpeak = ST depression induced by exercise relative to rest

11. the slope of the peak exercise ST segment

12. number of major vessels (0-3) colored by flourosopy

13. thal: 3 = normal; 6 = fixed defect; 7 = reversable defec

we will be using these 13 features to predict the target varibale which is coded 1 for heart desease and 0 not

In [6]:
```
## let's describe the data
heath_data.iloc[:,0:6].describe()
```

Out[6]:

	age	sex	cp	trestbps	chol	fbs
count	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
std	9.082101	0.466011	1.032052	17.538143	51.830751	0.356198
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000
25%	47.500000	0.000000	0.000000	120.000000	211.000000	0.000000
50%	55.000000	1.000000	1.000000	130.000000	240.000000	0.000000
75%	61.000000	1.000000	2.000000	140.000000	274.500000	0.000000
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000

In [7]:
```
## let's describe the data
heath_data.iloc[:,7:14].describe()
```

Out[7]:

	thalach	exang	oldpeak	slope	ca	thal	target
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000
mean	149.646865	0.326733	1.039604	1.399340	0.729373	2.313531	0.544554
std	22.905161	0.469794	1.161075	0.616226	1.022606	0.612277	0.498835
min	71.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	133.500000	0.000000	0.000000	1.000000	0.000000	2.000000	0.000000
50%	153.000000	0.000000	0.800000	1.000000	0.000000	2.000000	1.000000
75%	166.000000	1.000000	1.600000	2.000000	1.000000	3.000000	1.000000
max	202.000000	1.000000	6.200000	2.000000	4.000000	3.000000	1.000000

In [8]:
```
## we group the target by mean
heath_data.groupby('target').mean()
```

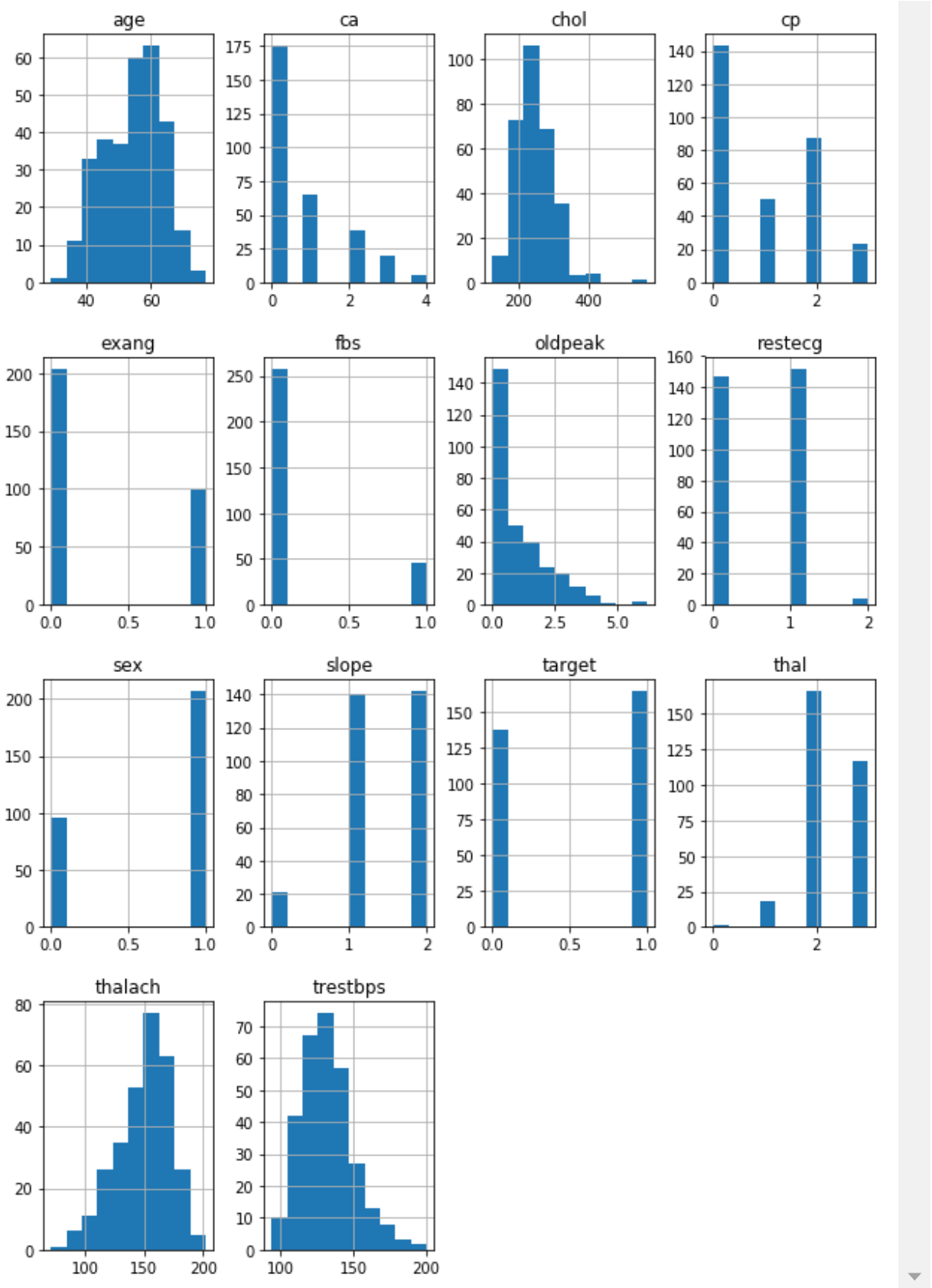
Out[8]:

target	age	sex	cp	trestbps	chol	fbs	restecg	thalach	e
0	56.601449	0.826087	0.478261	134.398551	251.086957	0.159420	0.449275	139.101449	0.5
1	52.496970	0.563636	1.375758	129.303030	242.230303	0.139394	0.593939	158.466667	0.1

4. Data Visualization

In [9]:
```
heath_data.hist(figsize=(10,15))
```

Out[9]:
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x000001F82BBC4C88>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001F82EBFB358>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001F82EC1E8D0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001F82EC47E48>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x000001F82EC79400>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001F82EC9E898>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001F82ECC5E10>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001F82ECF5400>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x000001F82ECF5438>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001F82ED43EB8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001F82ED72470>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001F82ED999E8>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x000001F82EDC3F60>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001F82EDF1518>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001F82EE18A90>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000001F82EE48048>]],
      dtype=object)
```



Based on the histogramme output certains features have normal, and exponential distribution

5. Preprossessing

we will put in x independent variables and y the target or dependent variable

In [10]:
```
x= heath_data.iloc[:,:13]  ## we take the 13 first features
y = heath_data['target'] ## we take the target
```

In [11]:
```
####Converting the dataset into a Dmatrix will allow us to take advantage of the p
dmatrix = xgb.DMatrix(data=x, label=y)
```

```
c:\users\diall\appdata\local\programs\python\python37\lib\site-packages\xgboos
t\core.py:587: FutureWarning: Series.base is deprecated and will be removed in
a future version
  if getattr(data, 'base', None) is not None and \
c:\users\diall\appdata\local\programs\python\python37\lib\site-packages\xgboos
t\core.py:588: FutureWarning: Series.base is deprecated and will be removed in
a future version
  data.base is not None and isinstance(data, np.ndarray) \
```

In [12]:
```
## We will take 80 % for traing and 20 % for testing
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_st
```

6. Creating and training the model

In [13]:
```
model =XGBClassifier()
model.fit(x_train,y_train)
```

Out[13]:
```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
       colsample_bytree=1, gamma=0, learning_rate=0.1, max_delta_step=0,
       max_depth=3, min_child_weight=1, missing=None, n_estimators=100,
       n_jobs=1, nthread=None, objective='binary:logistic', random_state=0,
       reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=None,
       silent=True, subsample=1)
```

7. Making predicting with XGBoost

In [14]:
```
y_pred =model.predict(x_test)
predictions =[round(value) for value in y_pred]
```

8. Test the performance of XGBoost

In [15]:
```
accurary =accuracy_score(y_test,predictions)
print(accurary)
```

```
0.7049180327868853
```

the Model has 70.49 % accuracy

9. k-fold Cross Validation using XGBoost

"In order to build more robust models, it is common to do a k-fold cross validation

where all the entries in the original training dataset are used for both training as well as validation. " Datacamp

In [16]:
```
params = {"objective":"reg:logistic",'colsample_bytree': 0.3,'learning_rate': 0.1,
                'max_depth': 8, 'alpha': 10}

cv_results = xgb.cv(dtrain=dmatrix, params=params, nfold=5,
                    num_boost_round=100,early_stopping_rounds=10,metrics="rmse", a
```

All you have to do is specify the nfolds parameter,

which is the number of cross validation sets you want to build.

Also, it supports many other parameters (check out this link) like:

num_boost_round: denotes the number of trees you build (analogous to n_estimators)

metrics: tells the evaluation metrics to be watched during CV

as_pandas: to return the results in a pandas DataFrame.

early_stopping_rounds: finishes training of the model early

if the hold-out metric ("rmse" in our case) does not improve for a given number of rounds.

cv_results contains train and test RMSE metrics for each boosting round.

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

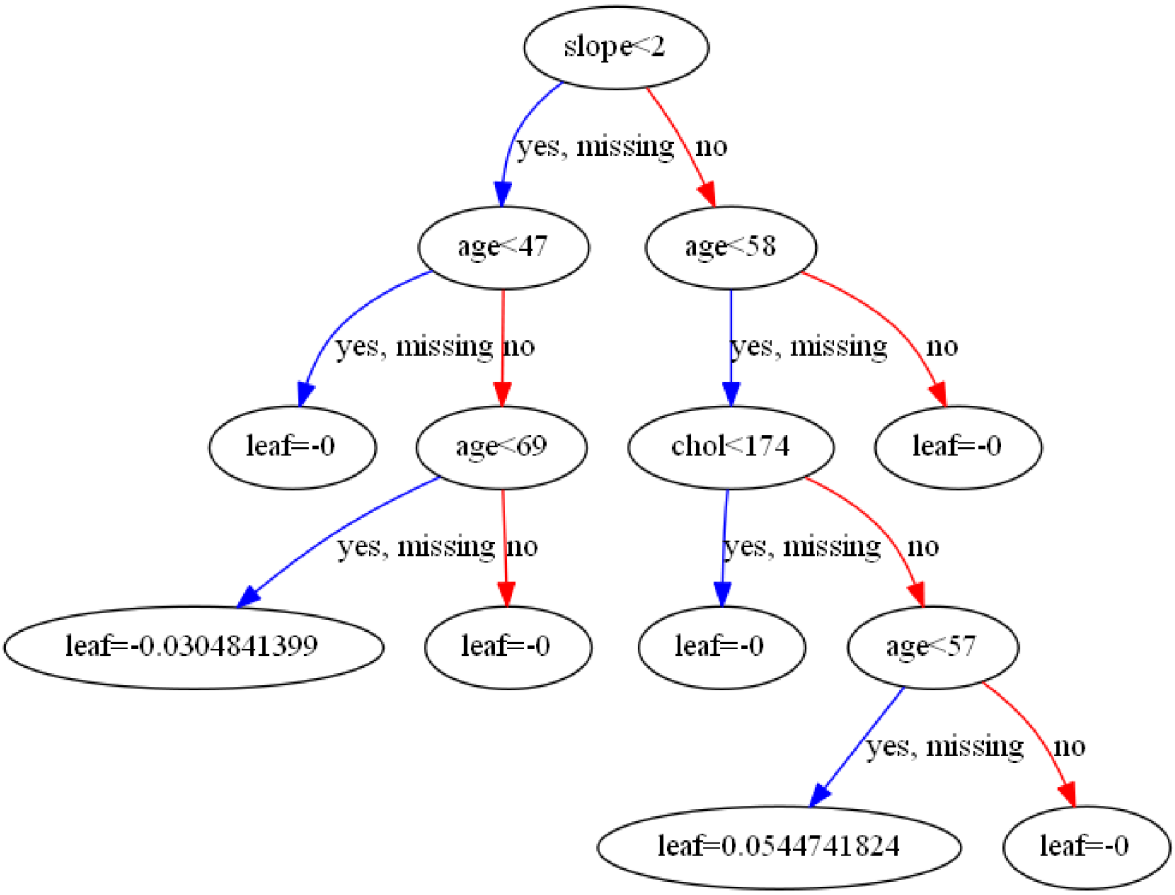
Out[17]:

	train-rmse-mean	train-rmse-std	test-rmse-mean	test-rmse-std
0	0.493998	0.001031	0.495090	0.001113
1	0.486648	0.002976	0.487668	0.001384
2	0.479699	0.003533	0.482182	0.002653
3	0.472716	0.004111	0.475834	0.003627
4	0.466659	0.005860	0.471564	0.004468

10. Visualize Boosting Trees

In [18]:
```
xg_reg = xgb.train(params=params, dtrain=dmatrix, num_boost_round=15)
```

In [20]:
```
xgb.plot_tree(xg_reg,num_trees=10)
plt.rcParams['figure.figsize'] = [30, 30]
plt.show()
```



References

https://xgboost.readthedocs.io/en/latest/python/python_api.xgboost.sklearn (https://xgboost.readthedocs.io/en/latest/python/python_api xgboost.sklearn)

https://www.datacamp.com/community/tutorials/xgboost-in-python (https://www.datacamp.com/community/tutorials/xgboost-in-python)

https://www.kaggle.com/ronitf/heart-disease-uci (https://www.kaggle.com/ronitf/heart-disease-uci)

https://www.dataiku.com/learn/guide/code/python/advanced xgboost-tuning.html (https://www.dataiku.com/learn/guide/code/python/advanced xgboost-tuning.html)

In [ ]: