Heart Disease Prediction:

In this blog, let us see how machine learning is used so as to predict if a patient is prone to heart disease or not.

Classification:

Machine learning could be split into three categories:

1.Supervised Learning. 2.Unsupervised Learning. 3.Reinforcement Learning.

When the target variable or the dependent variable is available then it is known as a supervised learning problem.

Supervised Learning could again be split into two categories:

1.Regression :

```
        When target variable or the dependent variable is continuous in nature (exa
mple: temperature), it is known as a regression problem.
```

2.Classification:

```
            When target variable or the dependent variable is discrete in nature (e
xample: 0 & 1,yes or no etc...), it is known as a classification problem.
```

In this problem we are trying to classify whether a person is prone to heart disease or not. Hence, we use the classification models to classify and to predict if a person has heart disease or not.

Dataset:

The dataset is available on the UCI machine learning repository. The link to the dataset is given below:

[https://archive.ics.uci.edu/ml/datasets/Heart+Disease (https://archive.ics.uci.edu/ml/datasets/Heart+Disease)](https://archive.ics.uci.edu/ml/datasets/Heart+Disease)

In this dataset the information about patients are given, here the target variable or the dependent variable is attribute "target" and we will have to fit a model so as to classify the target as 1 if a person has heart disease and 0 if a person is not having heart disease.

Python code:

Let's use the pandas package and import the dataset.

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

In [28]:
```
a=pd.read_csv('Z:\ml-python\heart.csv')
```

Here the first five observations of the dataset are shown:

In [29]:
```
a.head()
```

Out[29]:

	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

Null values:

The dataframe.isnull( ) function is used to check if the data frame consists of null values.

All the attributes show the false value which means that there are no null values in the dataset.

In [30]:
```
a.isnull().any()#no null values.
```

Out[30]:
```
age         False
sex         False
cp          False
trestbps    False
chol        False
fbs         False
restecg     False
thalach     False
exang       False
oldpeak     False
slope       False
ca          False
thal        False
target      False
dtype: bool
```

Now that the preprocessing step is over, but note that i haven't normalized the dataset also i haven't performed any EDA (which is an important step) in here.

The dataset can now be fitted onto a classification model.

Training and test splitting:

The dataset is being split into a training set(70%) and test set(30%), here I have specified random_state=12 so as to split values at random of 12 counts.

In [31]:
```
from sklearn.model_selection import train_test_split,cross_val_score
```

In [32]:
```
x=a.drop(['target'],axis=1)
y=a['target']
```

In [33]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_stat
e=12)
```

Next, we import all classification models to cross-validate on the training dataset.

In [34]:
```
from sklearn.naive_bayes import MultinomialNB
from sklearn.svm import LinearSVC
from sklearn.ensemble import AdaBoostClassifier,GradientBoostingClassifier,R
andomForestClassifier
from sklearn.linear_model import LogisticRegression,RidgeClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
```

Cross-validation:

The imported classification algorithms are now being run upon the training dataset with 'cv = 5', that is it will split the training dataset into five halves and check for the accuracy by keeping one part as the test dataset. This is an iterative process where each sample gets its part in both training and testing. Also, this is an efficient way of choosing the best model for the dataset, however, time complexity exists yet this is a worth doing step.

In [35]:
```
#cross validation:
nb=cross_val_score(MultinomialNB(),x_train,y_train,cv=5)
sv=cross_val_score(LinearSVC(),x_train,y_train,cv=5)
abd=cross_val_score(AdaBoostClassifier(),x_train,y_train,cv=5)
gb=cross_val_score(GradientBoostingClassifier(),x_train,y_train,cv=5)
rf=cross_val_score(RandomForestClassifier(),x_train,y_train,cv=5)
lr=cross_val_score(LogisticRegression(),x_train,y_train,cv=5)
rc=cross_val_score(RidgeClassifier(),x_train,y_train,cv=5)
dt=cross_val_score(DecisionTreeClassifier(),x_train,y_train,cv=5)
kn=cross_val_score(KNeighborsClassifier(),x_train,y_train,cv=5)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\svm\base.py:922: Convergen
ceWarning: Liblinear failed to converge, increase the number of iterations.
  "the number of iterations.", ConvergenceWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\svm\base.py:922: Convergen
ceWarning: Liblinear failed to converge, increase the number of iterations.
  "the number of iterations.", ConvergenceWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\svm\base.py:922: Convergen
ceWarning: Liblinear failed to converge, increase the number of iterations.
  "the number of iterations.", ConvergenceWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\svm\base.py:922: Convergen
ceWarning: Liblinear failed to converge, increase the number of iterations.
  "the number of iterations.", ConvergenceWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\svm\base.py:922: Convergen
ceWarning: Liblinear failed to converge, increase the number of iterations.
  "the number of iterations.", ConvergenceWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\ensemble\forest.py:246: Fu
tureWarning: The default value of n_estimators will change from 10 in version
0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\ensemble\forest.py:246: Fu
tureWarning: The default value of n_estimators will change from 10 in version
0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\ensemble\forest.py:246: Fu
tureWarning: The default value of n_estimators will change from 10 in version
0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\ensemble\forest.py:246: Fu
tureWarning: The default value of n_estimators will change from 10 in version
0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\ensemble\forest.py:246: Fu
tureWarning: The default value of n_estimators will change from 10 in version
0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:4
33: FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify
a solver to silence this warning.
  FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:4
33: FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify
a solver to silence this warning.
  FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:4
33: FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify
a solver to silence this warning.
  FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:4
33: FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify
a solver to silence this warning.
  FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:4
33: FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify
a solver to silence this warning.
  FutureWarning)
```

In this cell the cross-validation scores of each model are shown, out of all these models the logistic regression performs the best with an average accuracy of 86%.hence the logistic regression model will be as the model in heart disease prediction.

In [36]:
```
print("Cross validation scores : ")
print('naive_bayes : ',nb,nb.mean())
print('support_vector:',sv,sv.mean())
print('adaboost : ',abd,abd.mean())
print('randomforest : ',rf,rf.mean())
print('logistic_reg : ' ,lr,lr.mean())
print('gradient boost : ' ,gb,gb.mean())
print('ridge classifier : ',rc,rc.mean())
print('decision tree : ' ,dt,dt.mean())

print('knn : ' ,kn,kn.mean())

#print()
#print()
```

```
Cross validation scores :
naive_bayes :  [0.79069767 0.62790698 0.76744186 0.73809524 0.80487805] 0.745
8039597007267
support_vector: [0.44186047 0.74418605 0.65116279 0.76190476 0.73170732] 0.66
61642762607027
adaboost :  [0.8372093  0.81395349 0.76744186 0.80952381 0.80487805] 0.806601
3018934175
randomforest :  [0.81395349 0.79069767 0.76744186 0.80952381 0.90243902] 0.81
68111714339734
logistic_reg :  [0.88372093 0.88372093 0.79069767 0.88095238 0.87804878] 0.86
34281392647815
gradient boost :  [0.8372093  0.76744186 0.72093023 0.83333333 0.87804878] 0.
8073927018339951
ridge classifier :  [0.86046512 0.81395349 0.69767442 0.83333333 0.87804878]
0.8166950274153905
decision tree :  [0.74418605 0.65116279 0.74418605 0.83333333 0.7804878 ] 0.7
506712043864626
knn :  [0.69767442 0.51162791 0.6744186  0.69047619 0.58536585] 0.63191259487
34571
```

Logistic regression:

The training data is passed onto the logistic regression and a model is being fitted and the test data is passed to predict the target values.

In [37]:
```
#logistic regresssion model:
lr=LogisticRegression()
model=lr.fit(x_train,y_train)
pred=model.predict(x_train)
pred_test=model.predict(x_test)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:4
33: FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify
a solver to silence this warning.
  FutureWarning)
```

The predicted values are as follows:

In [38]:
```
pred_test=pd.DataFrame(pred_test)
pred_test
```

Out[38]:

	0
0	1
1	1
2	1
3	0
4	0
5	1
6	1
7	0
8	0
9	0
10	1
11	0
12	0
13	1
14	1
15	1
16	0
17	0
18	1
19	0
20	1
21	1
22	0
23	1
24	1
25	0
26	1
27	1
28	1
29	0
...	...
61	0
62	0
63	1
64	0
65	1
66	0
67	1
68	0
69	1
70	1
71	0

Accuracy and error of model:

Now, let's check the accuracy and the error term of the model. for this, I have imported the mean squared error to compute the error term, confusion matrix, and classification report to compute the accuracy of the model.

In [39]:
```
from sklearn.metrics import mean_squared_error,classification_report,confusi
on_matrix
```

Training data:

```
      The training data shows a minimal error of 0.12 which shows that the trainin
g data is fit well onto the model.
      The confusion matrix shows that 74 values are truly positive and 112 values a
re false negative whereas 19 values are truly negative and 7 values are false-posit
ive these values have been wrongly classified by the model.
      The classification report shows a good f-score and support count also the rec
all and precision values are above 80% which says that the training data is well fi
t in this model.
```

In [40]:
```
print("training data:")
print("error = ",mean_squared_error(y_pred=pred,y_true=y_train))
print(confusion_matrix(y_pred=pred,y_true=y_train))
print(classification_report(y_pred=pred,y_true=y_train))
```

```
training data:
error =  0.12264150943396226
[[ 74  19]
 [  7 112]]
              precision    recall  f1-score   support

           0       0.91      0.80      0.85        93
           1       0.85      0.94      0.90       119

   micro avg       0.88      0.88      0.88       212
   macro avg       0.88      0.87      0.87       212
weighted avg       0.88      0.88      0.88       212
```

Test data:

```
     The test data shows an error of 0.16 which is slightly higher than the trainin
g data but still, it is a minimal error.
      The confusion matrix shows that 35 values are truly positive and 41 values ar
e false negative whereas 10 values are truly negative and 5 values are false-positi
ve these values have been wrongly classified by the model.
      The classification report shows a good f-score and support count also the rec
all and precision values are above 80% which says that the test data is well fit in
this model.
```

In [41]:
```
print("testing data:")
print("error = ",mean_squared_error(y_pred=pred_test,y_true=y_test))
print(confusion_matrix(y_pred=pred_test,y_true=y_test))
print(classification_report(y_pred=pred_test,y_true=y_test))
```

```
testing data:
error =  0.16483516483516483
[[35 10]
 [ 5 41]]
              precision    recall  f1-score   support

           0       0.88      0.78      0.82        45
           1       0.80      0.89      0.85        46

   micro avg       0.84      0.84      0.84        91
   macro avg       0.84      0.83      0.83        91
weighted avg       0.84      0.84      0.83        91
```

Conclusion:

The logistic regression model has fit both the training and test dataset well, this model has the highest accuracy since the size of the dataset is minimal. Thus, we can use logistic regression to classify and predict if a person is prone to heart disease or not.

Thanks for reading this blog.

In [ ]:
```
```