DIABETES MELLITUS PREDICTION USING MACHINE LEARNING TECHNIQUES

Deephan.M, Mytheessvar.R, Monisha.M, Mohammed kaleemullah.A
Students,Department of Computer Science and Engineering,
SonaCollege of Technology,Salem,Tamilnadu,India.

ABSTRACT

Diabetes mellitus is a chronic disease characterized by hyperglycemia. It may cause many complications. According to the growing morbidity in recent years, in 2040, the world's diabetic patients will reach 642 million, which means that one of the ten adults in the future is suffering from diabetes. There is no doubt that this alarming figure needs great attention. With the rapid development of machine learning, machine learning has been applied to many aspects of medical health for accurate predictions.The aim of the project is to build a machine learning model that can efficiently discover the rules to predict the risk level of patients using Machine Learning Algorithms based on the given parameter about their health (In this project we are using random forest to predict diabetes mellitus). Then we evaluate the performance of the model in terms of different parameters like classification accuracy which comes under Supervised.An Web application is built from where the patient health features are entered and depending on the entered parameters, the machine learning model integrated to application will predict the type of diabetes and according to the type of diabetes diet plan for the person will be displayed on the UI.

Keywords: Machine Learning, Supervised, Random Forest.

INTRODUCTION

1.1 OVER VIEW

Diabetes is one of the major health problems of all over the world. Diabetes mellitus is classified into four broad categories: type 1, type 2, gestational diabetes and other specific types. All forms of diabetes increase the risk of long-term complications. These typically develop after many years (10–20), but may be the first symptom in those who have otherwise not received a diagnosis before that time. The criteria for diagnosing diabetes in pregnancy have been given the World Health Organization. criteria are as follows,

1. x fasting plasma glucose ≥ 7.0 mmol/l (126 mg/ dl)
2. 2-hour plasma glucose ≥ 11.1 mmol/l (200 mg/dl) following a 75g oral x glucose load.
3. random plasma glucose ≥ 11.1 mmol/l (200 mg/ dl) in the presence of diabetes symptoms.

Diagnostic criteria for diabetes in non-pregnant individuals are based on the relationship between plasma glucose values and the risk of diabetes-specific Micro vascular complications . People with diabetes have an increased risk of developing a number of serious health problems. Consistently high blood glucose levels can lead to serious diseases affecting the heart and blood vessels, eyes, kidneys, nerves and teeth. In addition, people with diabetes also have a higher risk of developing infections. In almost all developed countries, diabetes is a leading cause of cardiovascular disease, blindness, kidney failure, and lower limb amputation. Now it is very important to develop predictive models using the risk factors for the development of diabetes. Many studies have suggested traditional methods (statistical) as predictors.

Data mining predicts the future by modelling. Predictive modelling is the process by which a model is created to predict an outcome. The data mining process for diagnosis of diabetes can be divided into five steps, though the underlying principles and techniques used for data mining diabetic data bases may differ for different projects in different countries. Data mining is one of the "Knowledge Discovery in Databases" processes. The overall goal of the data mining process is to extract information from a data set and transform it into an

understandable structure for further use. This process has become an increasingly pervasive activity in all areas of medical science research. Data mining problems are often solved using different approaches from both computer sciences, such as multi-dimensional databases, machine learning, soft computing and data visualization; and statistics, including hypothesis testing, clustering, classification, and regression techniques. In recent years, data mining has been used widely in the areas of science and engineering, such as bioinformatics, genetics, medicine, and education.

1.1 PURPOSE:

The aim of the project is to build a machine learning model that can efficiently discover the rules to predict the risk level of patients based on the given parameter about their health. Then we evaluate the performance of the model in terms of different parameter like classification accuracy AUC-ROC Curves.A Web application is built from where the patient health features are entered and depending on the entered parameters, the machine learning model integrated to application will predict the type of diabetes and according to the type of diabetes diet plan for the person will be displayed on the UI.

2.LITERATURE SURVEY:

2.1.EXISTING PROBLEM:

In early predictiion techniques there is no risk factor only it predicts whether the person has diabetes or not,but in our project we the risk factor is known with some alert messages which is more user friendly and also it states for every age goup but in previous analysis it is only for certain age group with less accuracy.

2.1.PROPOSED SOLUTION:

The **Random Forest** algorithms are considered for our comparison analysis for prediction of diabetes.

Random Forest

Random forest algorithm is the statistical and machine learning algorithm which uses multiple learning algorithms to obtain better predictive performance than others. This algorithm has two parts

a. Tree bagging

b. From tree bagging to random forest

Each tree is grown as follows:

1. If the number of cases in the training set is N, sample N cases at random - but with replacement, from the original data. This sample will be the training set for growing the tree.

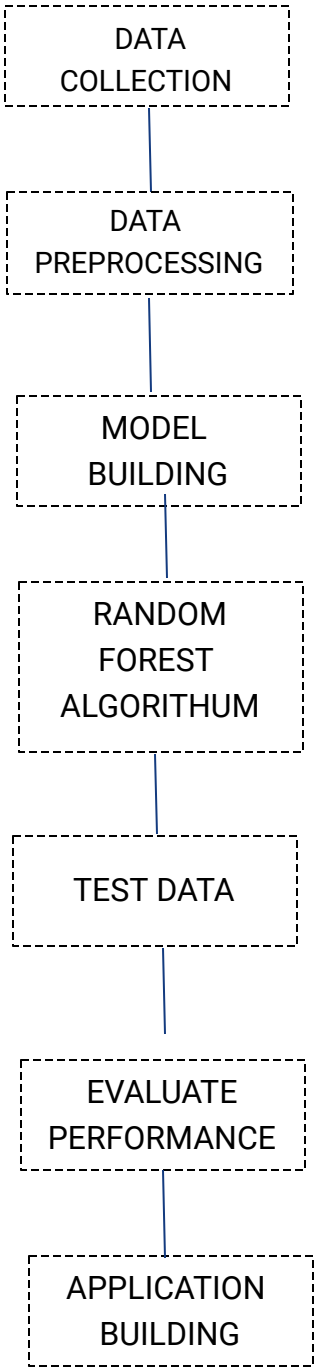
2. If there are M input variables, a random number of attributes are selected and the best \ split used to split the node. The value of M is held constant during the forest growing.

3. Each tree is grown to the largest extent possible. There is no pruning

3.THEORITICAL ANALYSIS

3.1.BLOCK DIAGRAM:

Block diagrams here represents the data collection,data preprocessing and further steps we also it includes the random forest algorithm then at last we done web application.



3.2.HARDWARE AND SOFTWARE DESIGN

A.Hardware requirements :
The following hardware was used for the implementationof the system:

- 4 GB RAM
- 10GB HDD
- Intel 1.66 GHz Processor Pentium 4

B. Software requirements:
The following software was used for the implementationof the system:

- Windows 7 or Windows 10
- Python
- Jupyter Notebook
- Spyder

4.EXPERIMENTAL INVESTIGATIONS

A.DATA COLLECTION

For the purpose of this study, Pima Indian Diabetes Dataset (PIDD) is considered as it is the best dataset for the present study. It contains 768 records. Every record has 9 attributes out of those one attribute is class variable. All 9 attributes contains only numeric data. Each record contains information about single patient. Table-I: Sample recorrd

DATASET:

A	B	C	D	E	F	G	H	I
preg	plas	pres	Skin	test	mass	pedi	age	class
6	148	72	35	0	33.6	0.627	50	1
1	85	66	29	0	26.6	0.351	31	0
8	183	64	0	0	23.3	0.672	32	1
1	89	66	23	94	28.1	0.167	21	0
0	137	40	35	168	43.1	2.288	33	1
5	116	74	0	0	25.6	0.201	30	0
3	78	50	32	88	31	0.248	26	1
10	115	0	0	0	35.3	0.134	29	0
2	197	70	45	543	30.5	0.158	53	1
8	125	96	0	0	0	0.232	54	1
4	110	92	0	0	37.6	0.191	30	0
10	168	74	0	0	38	0.537	34	1
10	139	80	0	0	27.1	1.441	57	0
1	189	60	23	846	30.1	0.398	59	1
5	166	72	19	175	25.8	0.587	51	1

PREG: This column indicates how many times a person is pregnant.
PLAS: This indicates plasma glucose concentration at 2 h in an oral glucose tolerance test.
PRES: This shows diastolic blood pressure.
SKIN: This indicates thickness of skin at triceps.
TEST: This demonstrates insulin level.
MASS: It demonstrates body mass index which is ratio of weight and height.
PEDI: It demonstrates how much probability a person can inherit diabetes from ancestors.
AGE: It provides or shows age of the person.
Class: It is a variable which contains only 0 or 1
1 indicates person having diabetes and 0 indicates person not having diabetes.

B.DATA PREPROCESSING

B1.ImportingThe Libraries:

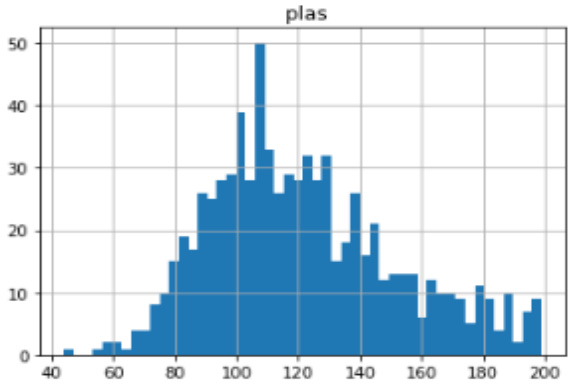
Pandas,Numpy,Matplotlib.pyplot,Seaborn and Sklearn are some llibraries we used here.
Libraries

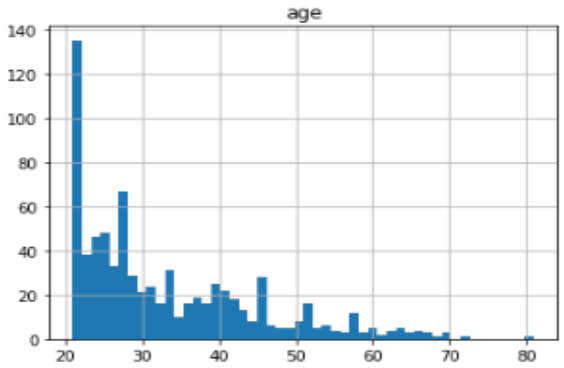
B2.Importing The Dataset :

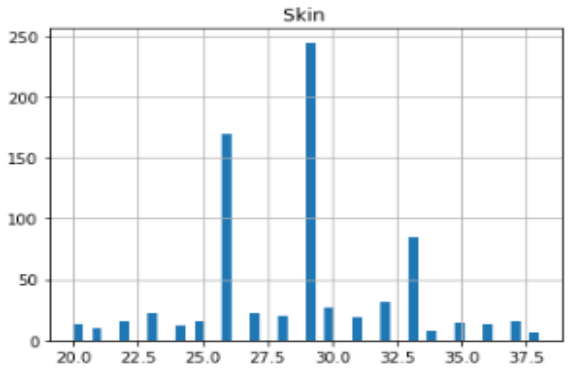
We will need to locate the directory of the CSV file at first (it's more efficient to keep the dataset in the same directory as your program) and read it using a method called read_csv which can be found in the library called pandas.

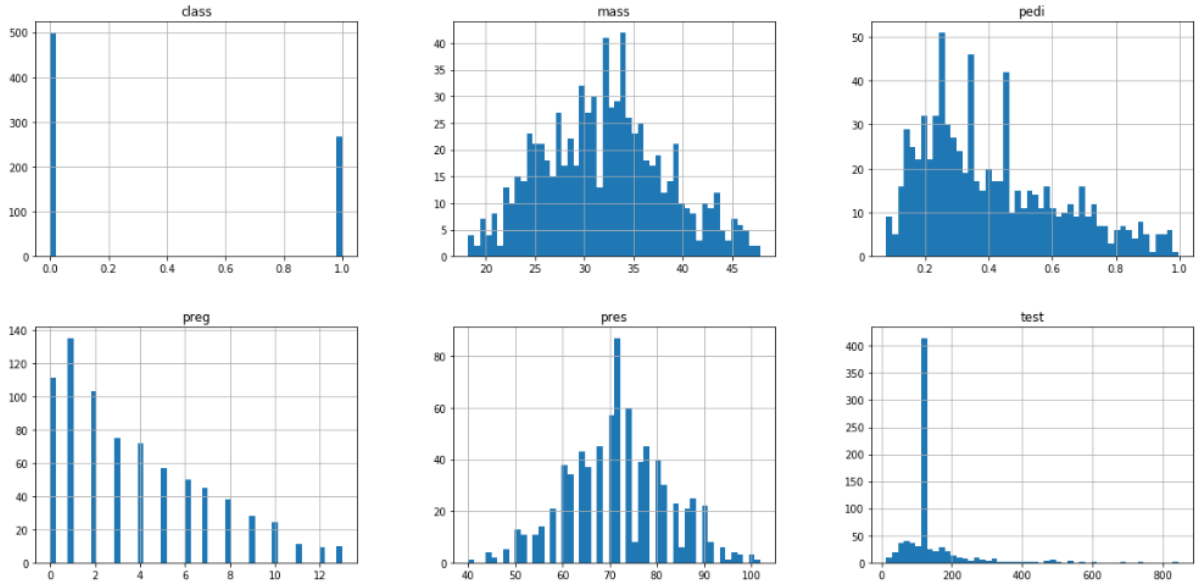
B3.DataVisualization :

To get clear understanding about each column we visualize the data









B4.Taking care of Missing Data:

Sometimes you may find some data are missing in the dataset. We need to be equipped to handle the problem when we come across them. Obviously you could remove the entire line of data but what if you are unknowingly removing crucial information? Of course we would not want to do that. One of the most common ideas to handle the problem is to take a mean of all the values of the same column and have it to replace the missing data.Initatialy there was no missing data but most of the columns are filled with 0, so we tried to replace the 0 with mean initally. When replacing the 0 with mean value tha accuracy was not that good which we expected.Then we tried with median in which it showed better accuracy.

B5. Splitting x and y and Label Encoding and OneHotEncoding

We need to split the dataset where input column which is independent variable and output column(class column) which is dependent variable.In This dataset first 8 columns are stored in x and last column is stored in y. No encoding required becuase the values are not strings.

B6.Feature Scaling:

The final step of data preprocessing is to apply the very important feature scaling. It is a method used to standardize the range of independent variables or features of data. A lot of machine learning models are based on Euclidean distance. If, for example, the values in one column (x) is much higher than the value in another column (y), (x2-x1) squared will give a far greater value than (y2-y1) squared. So clearly, one square difference dominates over the other square difference. In the machine learning equations, the square difference with the lower value in comparison to the far greater value will almost be treated as if it does not exist. We do not want that to happen. That is why it is necessary to transform all our variables into the same scale. There are several ways of scaling the data. One way is called Standardization which may be used. For every observation of the selected column, our program will apply the formula of standardization and fit it to a scale.

B7.Splitting dataset into independent variable and dependent variable :

Now we need to split our dataset into two sets — a Training set and a Test set. A general rule of the thumb is to allocate 80% of the dataset to training set and the remaining 20% to test set. For this task, we will import test_train_split from model_selection library of scikit. Now to build our training and test sets, we will create 4 sets— X_train (training part of the matrix of features), X_test (test part of the matrix of features), Y_train (training part of the dependent variables associated with the X train sets, and therefore also the same indices) , Y_test (test part of the dependent variables associated with the X test sets, and therefore also the same indices). We will assign to them the test_train_split, which takes the parameters — arrays (X and Y), test_size.

C.MODEL BUILDING:

C1.Training And Testing The Model:

A machine learning model can be a mathematical representation of a real-world process. To generate a machine learning model you will need to provide training data to a machine learning algorithm to learn from.Here we implemented Random Forest Classificatiion Algorithm to build our model.The model uses any one of the models that we had chosen. Once the model is trained we can use the same trained model to predict using the testing data i.e. the unseen data. For suppose you are using Linear Regression we will be using Skitlearn library to train the model and test the model.

C2.Evaluation:

Once this is done we can calculate the performance of the Random forest model by calculating accuracy.If the model shows accuracy more than 80 it menas model is trained well.AUC curve is also drawn here.

we also applied other algorithms to calculate the maximum accuracy.

When 0 are replaced with mean values:

ALGORITHM	ACCURACY
SVM	0.7184
KNN	0.7184
Naive Bayes	0.7378
Decision Tree	0.6769
Random Forest	0.7475

When 0 are replaced with median values and also outliers are removed expect test column and age column:

ALGORITHM	ACCURACY
SVM	0.7922
KNN	0.7922
Naive Bayes	0.7978
Decision Tree	0.8246
Random Forest	0.8441

When 0 are replaced with median values and also outliers are removed completely:

The accurcy is high when we remove all the outliers specially on test column and age column.It shows accuracy of 1 which is pretend to be overfit. A good algorithm should never be perfect there is always a small mistake or error. so we just kept outliers on test column and age column as it is.

RANDOM FOREST

In [44]:
```
from sklearn.ensemble import RandomForestClassifier
rf=RandomForestClassifier(n_estimators = 100,criterion='gini',random_state=0)
```

In [45]:
```
rf.fit(x_train,y_train)
```

Out[45]:
```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=200,
                       n_jobs=None, oob_score=False, random_state=0, verbose=0,
                       warm_start=False)
```

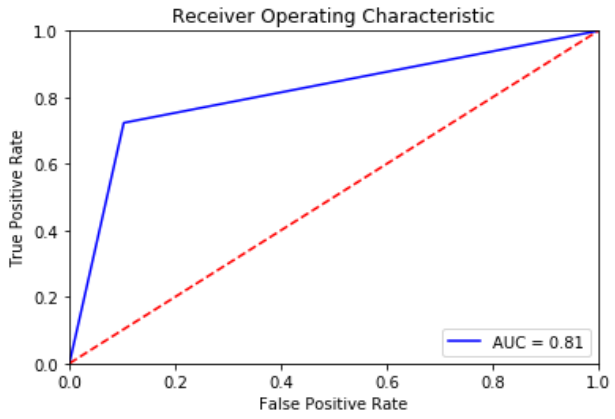
ACCURACY :0.8441

8]:
```
from sklearn.metrics import accuracy_score
ac = accuracy_score(y_test,y_pred)
ac
```

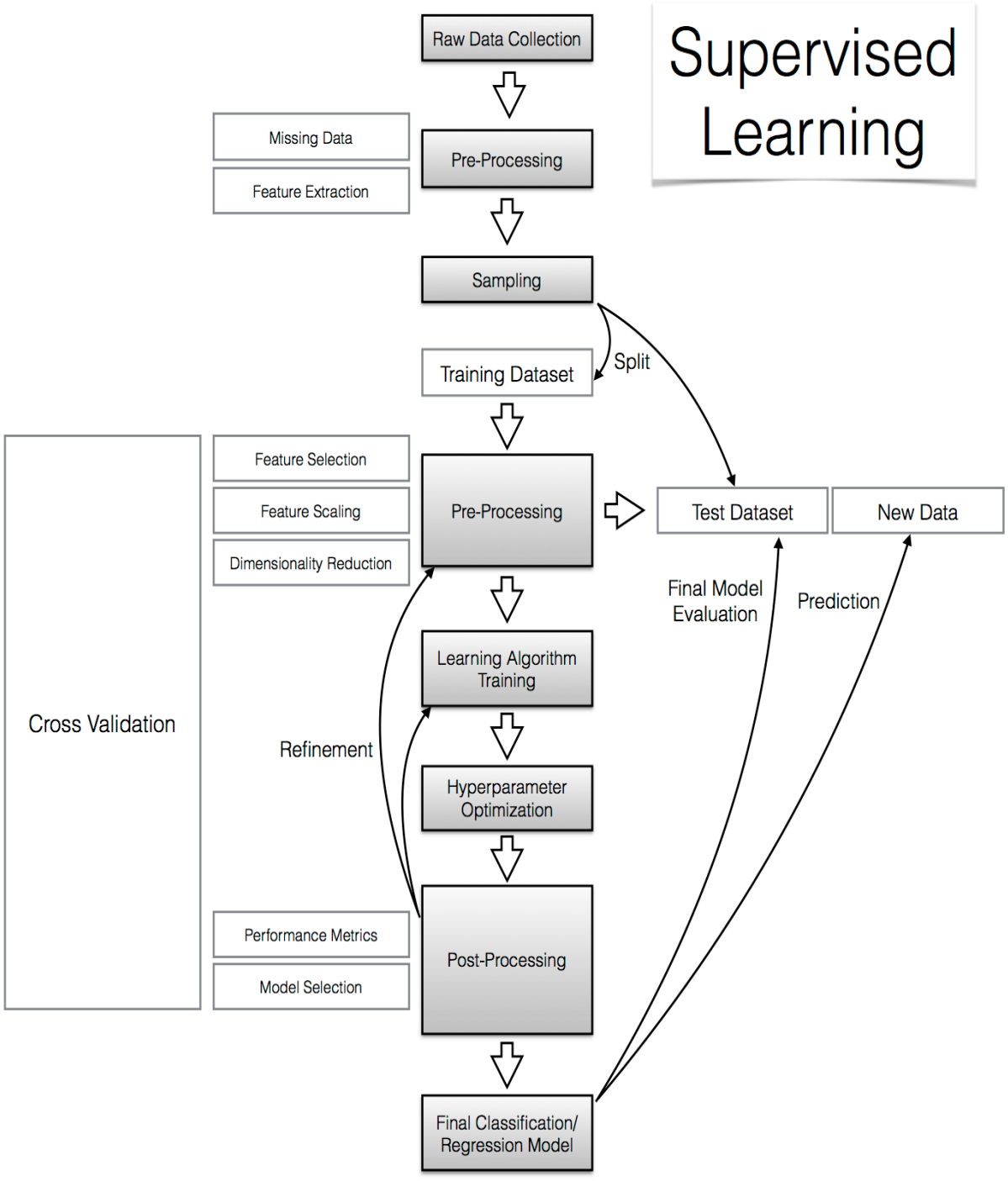
8]:
```
0.8441558441558441
```

9]:
```
import sklearn.metrics as metrics
fpr, tpr, threshold = metrics.roc_curve(y_test,y_pred)
roc_auc = metrics.auc(fpr, tpr)
```

0]:
```
plt.title('Receiver Operating Characteristic')
plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc_auc)
plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
```

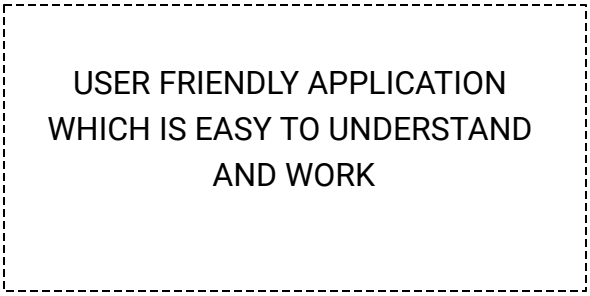


5.FLOWCHART


Supervised Learning
Raw Data Collection
Missing Data
Pre-Processing
Feature Extraction
Sampling
Split
Training Dataset
Feature Selection
Feature Scaling
Dimensionality Reduction
Pre-Processing
Test Dataset
New Data
Cross Validation
Final Model Evaluation
Prediction
Learning Algorithm Training
Refinement
Hyperparameter Optimization
Performance Metrics
Model Selection
Post-Processing
Final Classification/ Regression Model

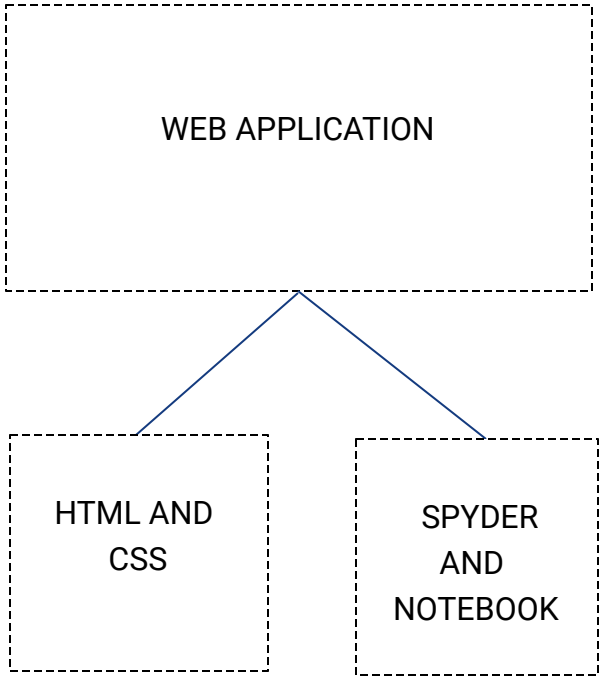
6.RESULT:

We have got the desired results of more than 80% accuracy for prediction of diabetes by using different classifiers. And the AUC curve also shows about 0.81 .



7.ADVANTAGES & DISADVANTAGES:

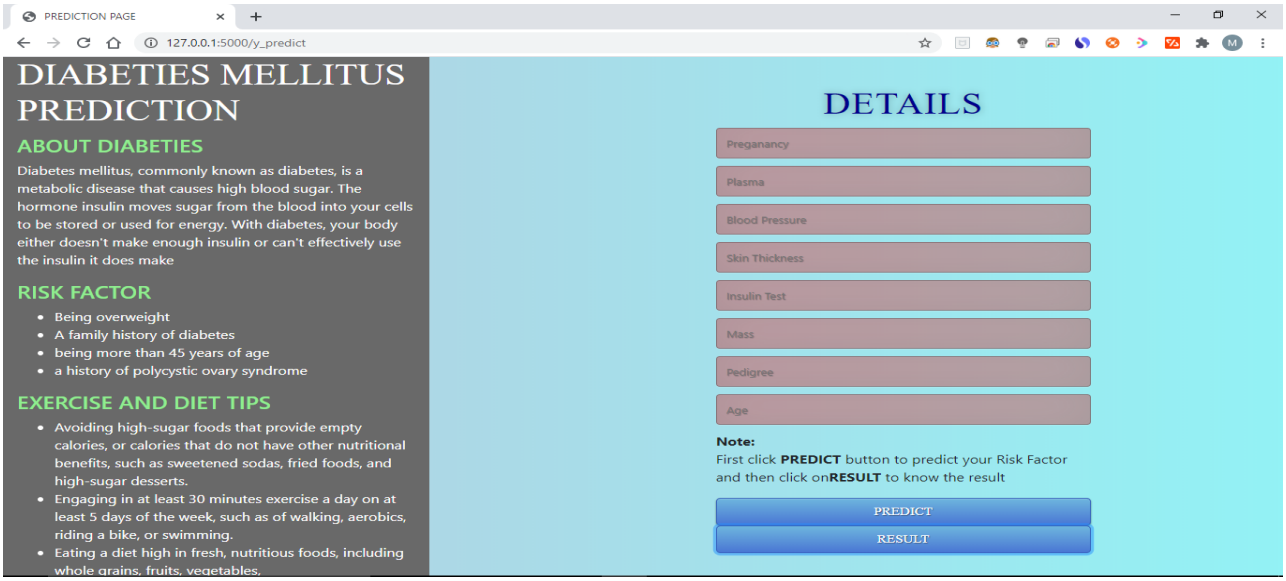
With this we can know our risk factor. Early diagnosis of diabetes and pre-diabetes is important so that patientscan begin to manage the disease early and potentially prevent or delay the serious disease complications that can decrease quality of life. and there is also a disadvantage in this, in some extent there may be wrong prediction.

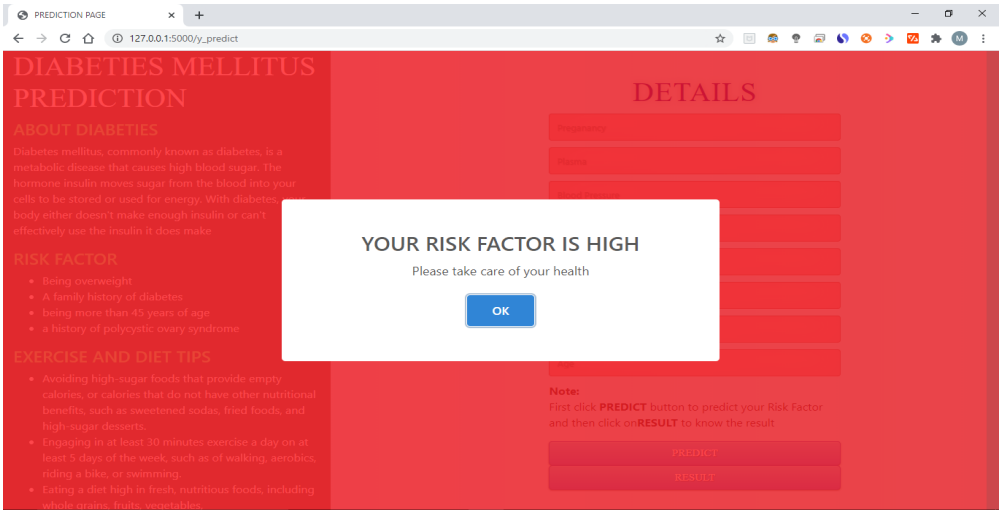


8.APPLICATION:

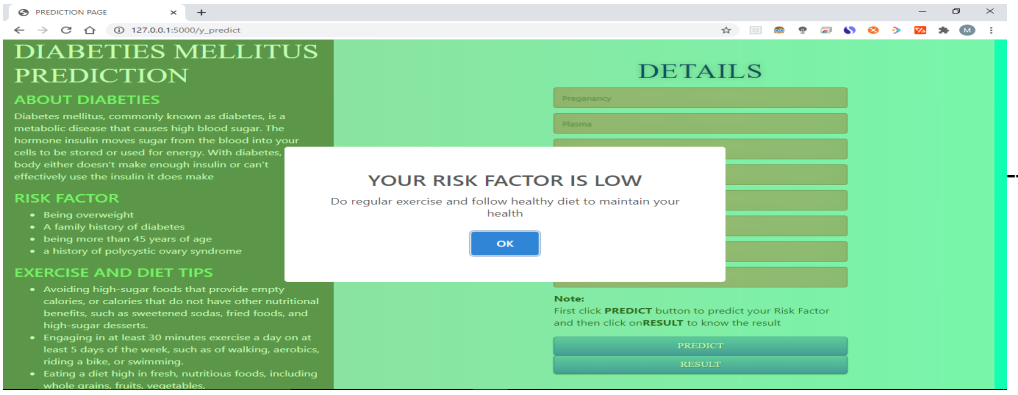
By making the algorithm predict correctly we implemented that in a web applicatioṇ. We have made a user friendly web applicationin which user can know the riskfactor and also there are some messages and tips in our application which will be more helpful to the user.Web application has about diabetes mellitus, Risk factors of diabetes mellitus and also some tips to keep our body health.This is applied in hospitalites in which tech can predict the risk factor of the patient easily.

USER INTERFACE:





-->Risk factor High



-->Risk factor low

9.CONCLUSION:

The machine learning methods can support the doctors to identify and cure diabetic diseases. We shall conclude that the improvement in classification accuracy helps to make the machine learning models get better results. The performance analysis is in terms of accuracy rate among all the classification techniques such as decision tree, logistic regression, k-nearest neighbors, naive bayes, and SVM , random forest. We have also seen that the accuracy of the existing system is less than 70% hence we proposed to use a combination of classifiers . We have found that our system provides us with 84.44 % of accuracy for Random Forest Classifier.

10.FUTURE SCOPE:

In future, if we get a large set of diabetic dataset we can perform comparative analysis for analyzing the performance of each algorithm as well as the Hybrid algorithm so that the best one can be applied for predictive analysis. A particular method to identify diabetes is not very sophisticated way for initial diabetes detection and it is not fully accurate for predicting diseases. That's why we need a smart hybrid predictive analytics diabetes diagnostic system that can effectively work with accuracy and efficiency. We can use data mining , neural network for exploring and utilizing to support medical decision, which improves in diagnosing the risk for pregnant diabetes. Due to the dataset we have till date are not upto the mark , we cannot predict the type of diabetes, so in future we aim to predicting type of diabetes and explore it, which may improve the accuracy of predicting diabetes. We can also study the causes of diabetes and how to avoid having diabetes.

11.BIBILOGRAPHY:

[1] Kaur, H., & Kumari, V. (2018). Predictive modelling and analytics for diabetes using a machine learning approach. Applied Computing and Informatics. [2] Carter, J. A., Long, C. S., Smith, B. P., Smith, T. L., & Donati, G. L. (2019). Combining elemental analysis of toenails and machine learning techniques as a non-invasive diagnostic tool for the robust classification of type-2 diabetes. Expert Systems with Applications, 115, 245-255. [3] Kavakiotis, I., Tsave, O., Salifoglou, A., Maglaveras, N., Vlahavas, I., & Chouvarda, I. (2017). Machine learning and data mining methods in diabetes research. Computational and structural biotechnology journal, 15, 104-116. [4] Mahmud, S. M., Hossin, M. A., Ahmed, M. R., Noori, S. R. H., & Sarkar, M. N. I. (2018, August). Machine Learning Based Unified Framework for Diabetes Prediction. In Proceedings of the 2018 International Conference on Big Data Engineering and Technology (pp. 46-50). ACM. [5] Patil, R., & Tamane, S. (2018). A Comparative Analysis on the Evaluation of Classification Algorithms in the Prediction of Diabetes. International Journal of Electrical and Computer Engineering, 8(5), 3966. [6] Dagliati, A., Marini, S., Sacchi, L., Cogni, G., Teliti, M., Tibollo, V., ... & Bellazzi, R. (2018). Machine learning methods to predict diabetes complications. Journal of diabetes science and technology, 12(2), 295302. [7] Barik, R. K., Priyadarshini, R., Dubey, H., Kumar, V., & Yadav, S. (2018). Leveraging machine learning in mist computing telemonitoring system for diabetes prediction. In Advances in Data and Information Sciences (pp. 95-104). Springer, Singapore. [8] Choudhury, A., & Gupta, D. (2019). A Survey on Medical Diagnosis of Diabetes Using Machine Learning Techniques. In Recent Developments in Machine Learning and Data Analytics (pp. 67-78). Springer, Singapore. [9] Samant, P., & Agarwal, R. (2017). Diagnosis of diabetes using computer methods: soft computing methods for diabetes detection using iris. Threshold, 8, 9.

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

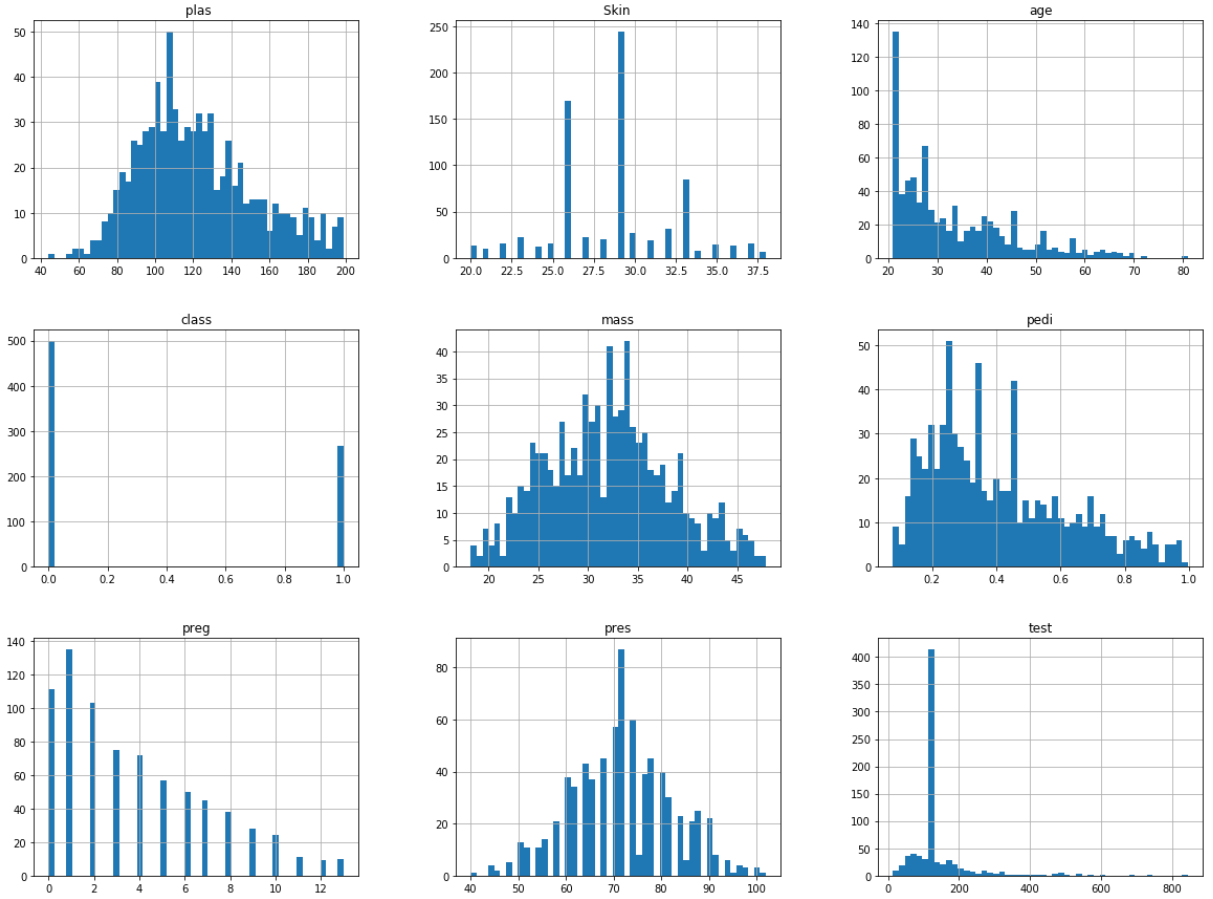
In [2]:
```
ds=pd.read_csv("pima-indians-diabetes.data.csv")
ds
```

Out[2]:

	preg	plas	pres	Skin	test	mass	pedi	age	class
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1
...	...	...	...	...	...	...	...	...	...
763	10	101	76	48	180	32.9	0.171	63	0
764	2	122	70	27	0	36.8	0.340	27	0
765	5	121	72	23	112	26.2	0.245	30	0
766	1	126	60	0	0	30.1	0.349	47	1
767	1	93	70	31	0	30.4	0.315	23	0

768 rows × 9 columns

In [52]:
```
import matplotlib.pyplot as plt
ds.hist(bins=50, figsize=(20, 15))
plt.show()
```



In [3]:
```
ds.isnull().any()#there are no nan values but there are 0
```

Out[3]:
```
preg     False
 plas    False
pres     False
Skin     False
test     False
mass     False
pedi     False
age      False
class    False
dtype: bool
```

In [4]:
```
ds[' plas'] = ds[' plas'].replace([0],[np.nan])
ds['pres'] = ds['pres'].replace([0],[np.nan])
ds['Skin '] = ds['Skin '].replace([0],[np.nan])
ds['mass'] = ds['mass'].replace([0],[np.nan])
ds['test'] = ds['test'].replace([0],[np.nan])
```

In [5]:
```
ds.isnull().sum()
```

Out[5]:
```
preg       0
 plas      5
pres      35
Skin     227
test     374
mass      11
pedi       0
age        0
class      0
dtype: int64
```

In [6]:
```
diabetes_true_count = len(ds.loc[ds['class'] == True])
diabetes_false_count = len(ds.loc[ds['class'] == False])
(diabetes_true_count,diabetes_false_count)
```

Out[6]:
```
(268, 500)
```

In [7]:
```
ds[' plas'].fillna(ds[' plas'].median(),inplace=True)
ds['pres'].fillna(ds['pres'].median(),inplace=True)
ds['Skin '].fillna(ds['Skin '].median(),inplace=True)
ds['mass'].fillna(ds['mass'].median(),inplace=True)
ds['test'].fillna(ds['test'].median(),inplace=True)
```

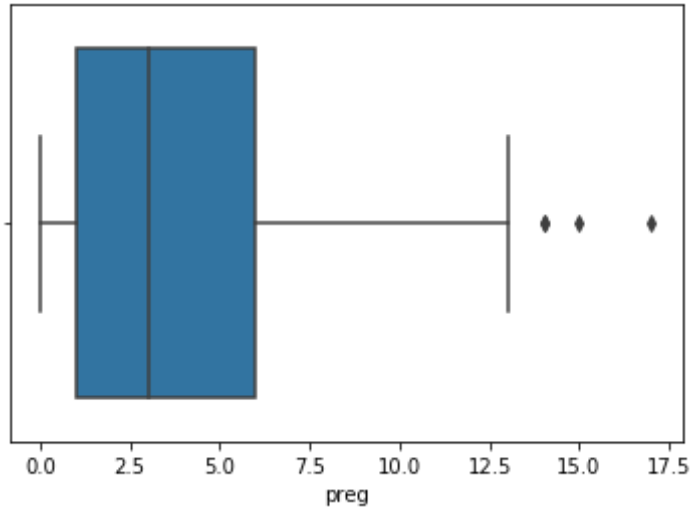
In [8]:
```
ds.isnull().sum()
```

Out[8]:
```
preg     0
 plas    0
pres     0
Skin     0
```

```
test     0
mass     0
pedi     0
age      0
class    0
dtype: int64
```

In [9]:
```
import seaborn as sns
sns.boxplot(x=ds['preg'])
```

Out[9]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x21faee95bc8>
```



In [10]:
```
ds['preg'].value_counts()
```

Out[10]:
```
1     135
0     111
2     103
3      75
4      68
5      57
6      50
7      45
8      38
```

```
9      28
10     24
11     11
13     10
12      9
14      2
15      1
17      1
Name: preg, dtype: int64
```

In [11]:
```
def median_target(var):
    temp = ds[ds[var].notnull()]
    temp = temp[[var, 'class']].groupby(['class'])[[var]].median().rese
t_index()
    return temp
```

In [12]:
```
ds['preg'].median()
```

Out[12]:
```
3.0
```

In [13]:
```
median_target('preg')
```

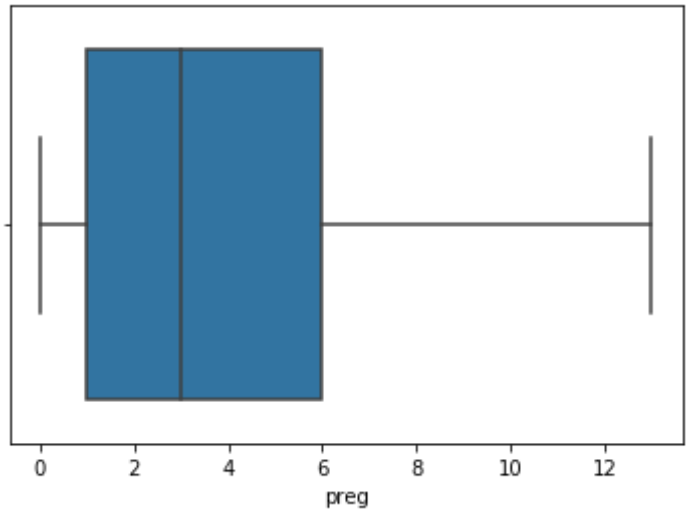
Out[13]:

	class	preg
0	0	2
1	1	4

In [14]:
```
ds.loc[(ds['class'] == 0 ) & (ds['preg']>13), 'preg'] = 2
ds.loc[(ds['class'] == 1 ) & (ds['preg']>13), 'preg'] = 4
```

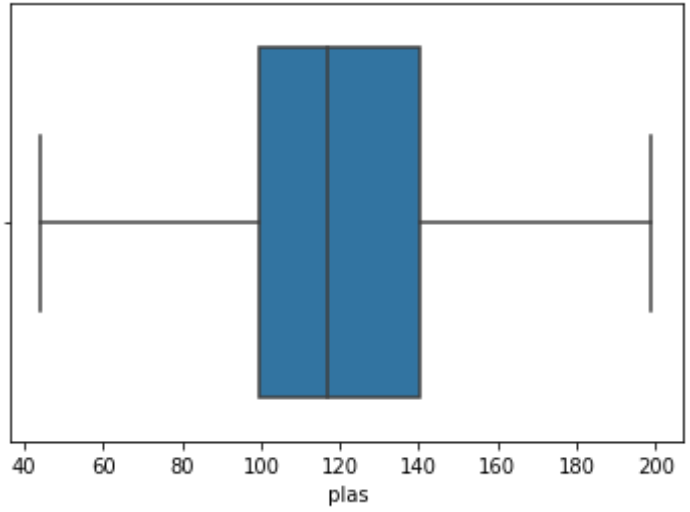
In [15]:
```
sns.boxplot(x=ds['preg'])
```

Out[15]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x21faf6212c8>
```



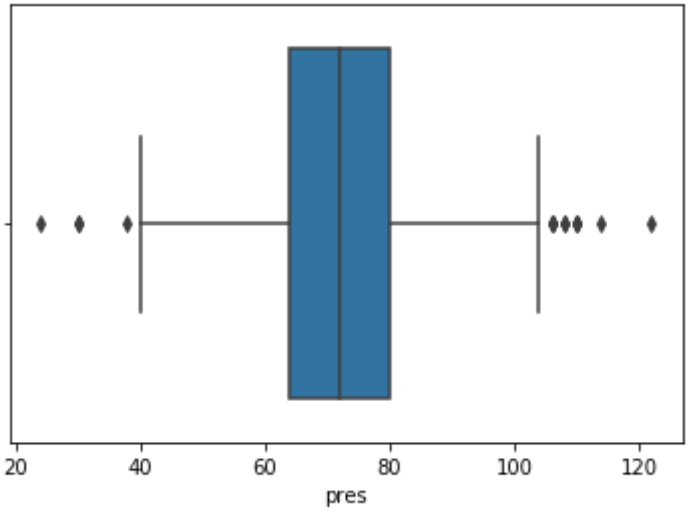
In [16]:
```
sns.boxplot(x=ds[' plas'])
```

Out[16]: <matplotlib.axes._subplots.AxesSubplot at 0x21faf6958c8>



In [17]:
```
sns.boxplot(x=ds['pres'])
```

Out[17]: <matplotlib.axes._subplots.AxesSubplot at 0x21faf6fde08>



In [18]:
```
ds['pres'].median()
```

Out[18]: 72.0

In [19]:
```
median_target('pres')
```

Out[19]:

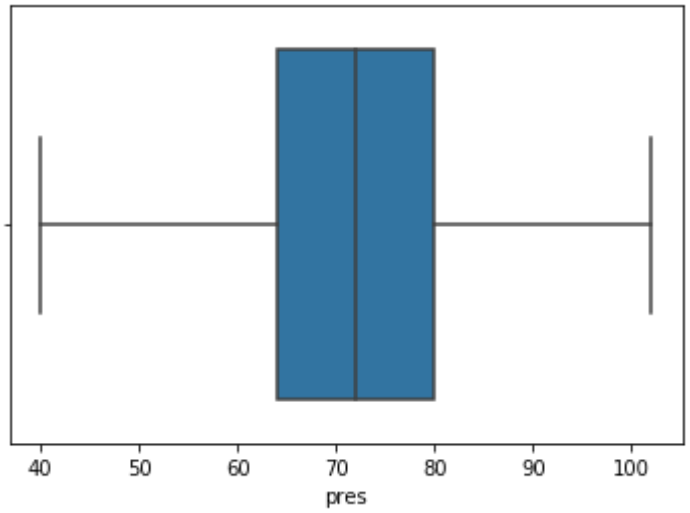
	class	pres
0	0	72.0
1	1	74.0

In [20]:
```
ds.loc[(ds['class'] == 0 ) & (ds['pres']<40), 'pres'] = 72.0
ds.loc[(ds['class'] == 1 ) & (ds['pres']<40), 'pres'] = 74.5
```

In [21]:
```
ds.loc[(ds['class'] == 0 ) & (ds['pres']>103), 'pres'] = 72.0
ds.loc[(ds['class'] == 1 ) & (ds['pres']>103), 'pres'] = 74.5
```

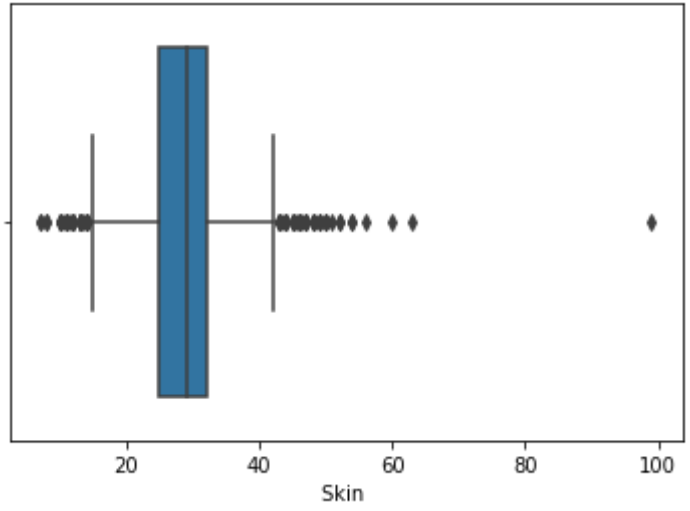
In [22]:
```
sns.boxplot(x=ds['pres'])
```

Out[22]: <matplotlib.axes._subplots.AxesSubplot at 0x21faf786f48>



In [23]:
```
sns.boxplot(x=ds['Skin '])
```

Out[23]: <matplotlib.axes._subplots.AxesSubplot at 0x21faf7e2648>



In [24]:
```
median_target('Skin ')
```

Out[24]:

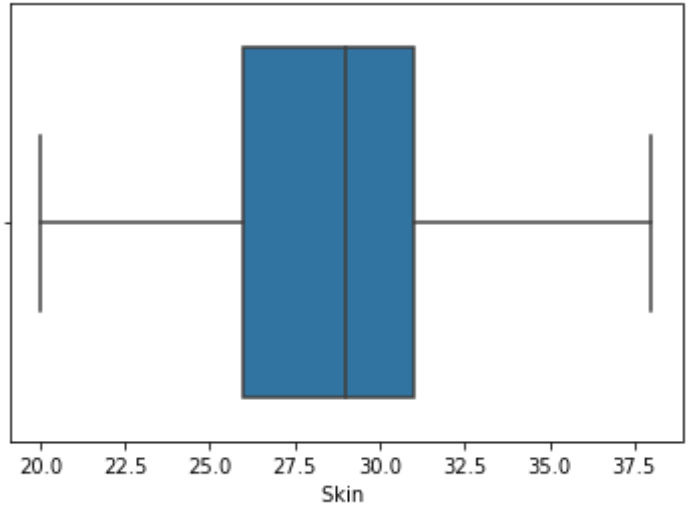
	class	Skin
0	0	29.0
1	1	29.0

In [25]:
```
ds.loc[(ds['class'] == 0 ) & (ds['Skin ']>38), 'Skin '] = 26.0
ds.loc[(ds['class'] == 1 ) & (ds['Skin ']>38), 'Skin '] = 33.0
```

In [26]:
```
ds.loc[(ds['class'] == 0 ) & (ds['Skin ']<20), 'Skin '] = 26.0
ds.loc[(ds['class'] == 1 ) & (ds['Skin ']<20), 'Skin '] = 33.0
```

In [27]:
```
sns.boxplot(x=ds['Skin '])
```

Out[27]: <matplotlib.axes._subplots.AxesSubplot at 0x21faf85d208>



In [28]:
```
median_target('mass')
```

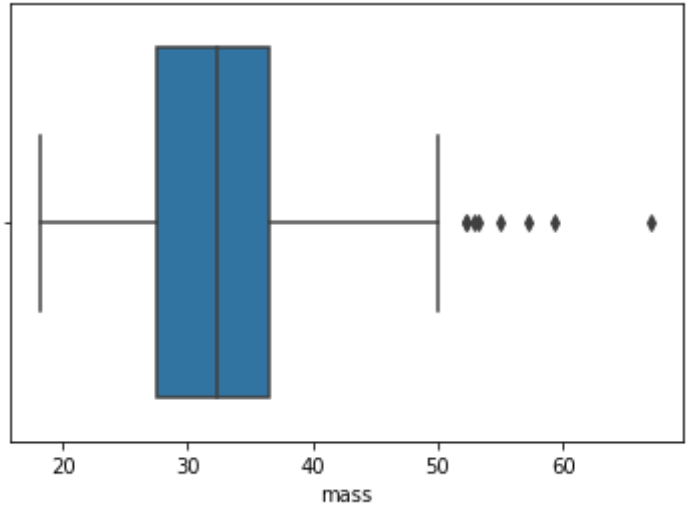
Out[28]:

	class	mass

	class	mass
0	0	30.40
1	1	34.25

In [29]:
```
sns.boxplot(x=ds['mass'])
```

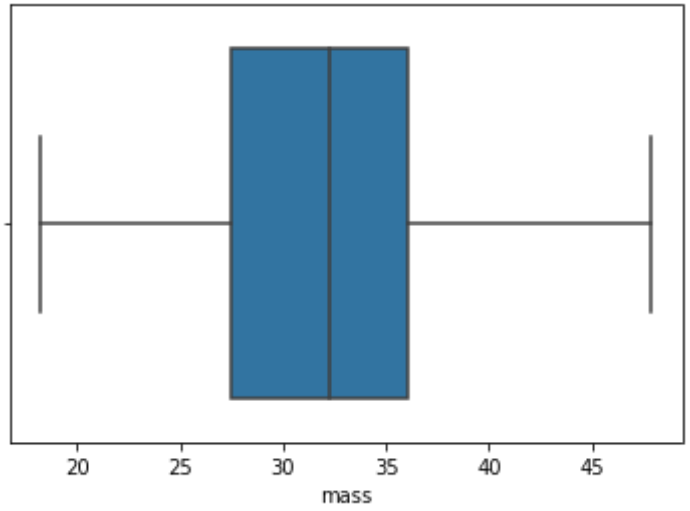
Out[29]: <matplotlib.axes._subplots.AxesSubplot at 0x21faf8a2f88>



In [30]:
```
ds.loc[(ds['class'] == 0 ) & (ds['mass']>48), 'mass'] = 30.40
ds.loc[(ds['class'] == 1 ) & (ds['mass']>48), 'mass'] = 34.23
```

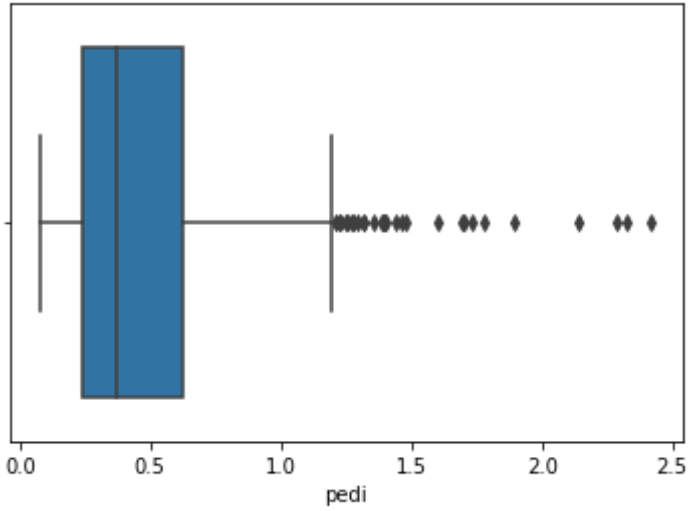
In [31]:
```
sns.boxplot(x=ds['mass'])
```

Out[31]: <matplotlib.axes._subplots.AxesSubplot at 0x21faf90ecc8>



In [32]:
```
sns.boxplot(x=ds['pedi'])
```

Out[32]: <matplotlib.axes._subplots.AxesSubplot at 0x21faf83b3c8>



In [33]:
```
median_target('pedi')
```

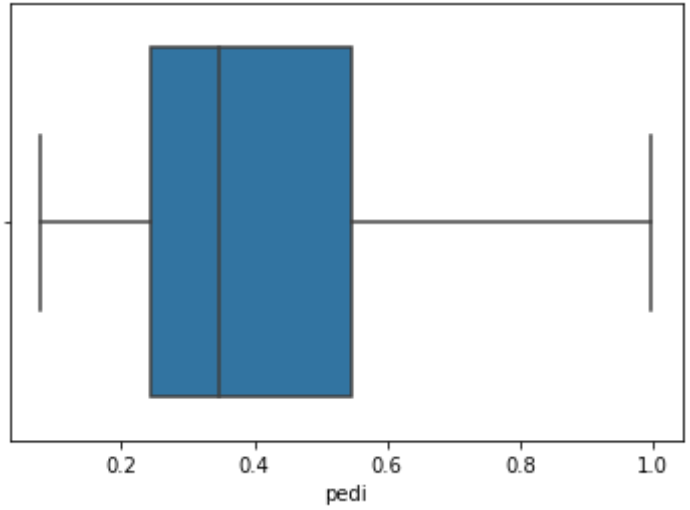
Out[33]:

	class	pedi
0	0	0.336
1	1	0.449

In [34]:
```
ds.loc[(ds['class'] == 0 ) & (ds['pedi']>1), 'pedi'] = 0.336
ds.loc[(ds['class'] == 1 ) & (ds['pedi']>1), 'pedi'] = 0.449
```

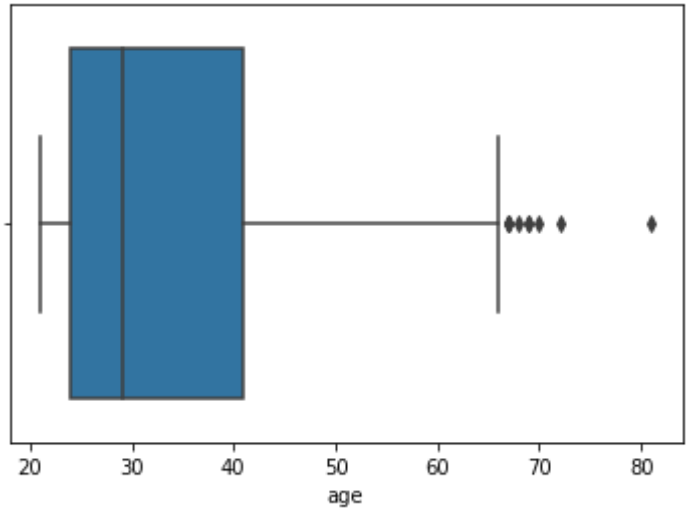
In [35]:
```
sns.boxplot(x=ds['pedi'])
```

Out[35]: <matplotlib.axes._subplots.AxesSubplot at 0x21faf9f0f48>



In [36]:
```
sns.boxplot(x=ds['age'])
```

Out[36]: <matplotlib.axes._subplots.AxesSubplot at 0x21fafa40648>



In [37]:
```
median_target('age')
```

Out[37]:

	class	age
0	0	27
1	1	36

In [38]:
```
ds
```

Out[38]:

	preg	plas	pres	Skin	test	mass	pedi	age	class
0	6	148.0	72.0	35.0	125.0	33.6	0.627	50	1
1	1	85.0	66.0	29.0	125.0	26.6	0.351	31	0
2	8	183.0	64.0	29.0	125.0	23.3	0.672	32	1
3	1	89.0	66.0	23.0	94.0	28.1	0.167	21	0
4	0	137.0	40.0	35.0	168.0	43.1	0.449	33	1
...	...	...	...	...	...	...	...	...	...

	preg	plas	pres	Skin	test	mass	pedi	age	class
763	10	101.0	76.0	26.0	180.0	32.9	0.171	63	0
764	2	122.0	70.0	27.0	125.0	36.8	0.340	27	0
765	5	121.0	72.0	23.0	112.0	26.2	0.245	30	0
766	1	126.0	60.0	29.0	125.0	30.1	0.349	47	1
767	1	93.0	70.0	31.0	125.0	30.4	0.315	23	0

768 rows × 9 columns

In [39]:
```
x=ds.iloc[:,0:8].values
x
```

Out[39]:
```
array([[  6.   , 148.   ,  72.   , ...,  33.6  ,   0.627,  50.   ],
       [  1.   ,  85.   ,  66.   , ...,  26.6  ,   0.351,  31.   ],
       [  8.   , 183.   ,  64.   , ...,  23.3  ,   0.672,  32.   ],
       ...,
       [  5.   , 121.   ,  72.   , ...,  26.2  ,   0.245,  30.   ],
       [  1.   , 126.   ,  60.   , ...,  30.1  ,   0.349,  47.   ],
       [  1.   ,  93.   ,  70.   , ...,  30.4  ,   0.315,  23.   ]])
```

In [40]:
```
y=ds.iloc[:,-1].values
y
```

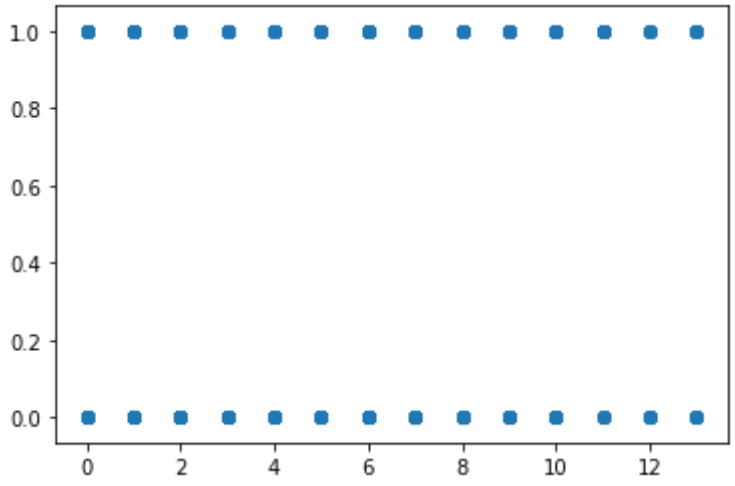
Out[40]:
```
array([1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0,
0,
       1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0,
1,
       0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1,
0,
       1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0,
0,
       1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0,
1,
       1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1,
1,
       1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1,
```

```
0,
        1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0,
1,
        0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0,
1,
        1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1,
1,
        1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0,
0,
        1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0,
0,
        1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1,
0,
        0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1,
0,
        1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1,
0,
        0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
0,
        0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0,
0,
        0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1,
0,
        0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0,
1,
        0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0,
0,
        1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0,
0,
        0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0,
0,
        1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0,
0,
        1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
0,
        0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0,
0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0,
0,
```

```
       0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1,
0,
       0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1,
0,
       0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0,
0,
       1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
1,
       0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0,
1,
       0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1,
0,
       0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0,
0,
       0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1,
0,
       1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0],
      dtype=int64)
```

In [41]:
```
plt.scatter(x[:,0],y)
```

Out[41]:
```
<matplotlib.collections.PathCollection at 0x21fafaeb4c8>
```



In [42]:
```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x = sc.fit_transform(x)
x
```

Out[42]:
```
array([[ 0.67692751,  0.86604475, -0.00472509, ...,  0.24092116,
         1.015903  ,  1.4259954 ],
       [-0.85303226, -1.20506583, -0.56301241, ..., -0.88255837,
        -0.26647172, -0.19067191],
       [ 1.28891142,  2.01666174, -0.74910818, ..., -1.41219872,
         1.22498584, -0.10558415],
       ...,
       [ 0.37093556, -0.02157407, -0.00472509, ..., -0.9467572 ,
        -0.75897795, -0.27575966],
       [-0.85303226,  0.14279979, -1.12129972, ..., -0.32081861,
        -0.27576429,  1.17073215],
       [-0.85303226, -0.94206766, -0.19082086, ..., -0.27266948,
        -0.43373799, -0.87137393]])
```

In [43]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2,rand
om_state=0)
```

RANDOM FOREST

In [44]:
```
from sklearn.ensemble import RandomForestClassifier
rf=RandomForestClassifier(n_estimators = 100,criterion='gini',random_st
ate=0)
```

In [45]:
```
rf.fit(x_train,y_train)
```

Out[45]:
```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=Non
e,
                       criterion='gini', max_depth=None, max_features
='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=No
```

```
ne,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=200,
                       n_jobs=None, oob_score=False, random_state=0, ve
rbose=0,
                       warm_start=False)
```

In [46]:
```
y_pred = rf.predict(x_test)
y_pred
```

Out[46]:
```
array([1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1,
0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0,
1,
       1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1,
1,
       1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0,
1,
       0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1,
0,
       0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0,
0],
      dtype=int64)
```

In [47]:
```
y_test
```

Out[47]:
```
array([1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1,
1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0,
1,
       1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1,
1,
       1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
0,
       1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0,
1,
       0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
```

```
       0,
              0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0,
       0],
             dtype=int64)
```

In [48]:
```
from sklearn.metrics import accuracy_score
ac = accuracy_score(y_test,y_pred)
ac
```

Out[48]:
```
0.8441558441558441
```

In [49]:
```
import sklearn.metrics as metrics
fpr, tpr, threshold = metrics.roc_curve(y_test,y_pred)
roc_auc = metrics.auc(fpr, tpr)
```

In [50]:
```
plt.title('Receiver Operating Characteristic')
plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc_auc)
plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
```