DATA MINING ASSIGNMENT

MALARIA PREDICTION USING CELL IMAGES

Manipal University Jaipur

By-

Name: Jyoti Tuli

Reg No.: 179302068

Background of The Study

Data Mining is rapidly growing to occupy all the industries of the world today. This is just the beginning. Medical disease prediction is regarded as one of the most important subjects in data analysis which is done using exploratory analysis. The health care industries collect huge amounts of data containing hidden information useful for making effective decisions by providing appropriate results using data science techniques.

Data mining knowledge is used to give a user-oriented approach to new and hidden patterns in the data which can be used by the healthcare experts for predicting malaria which can improve the entire research and prevention process, making sure that more people can live healthy lives.

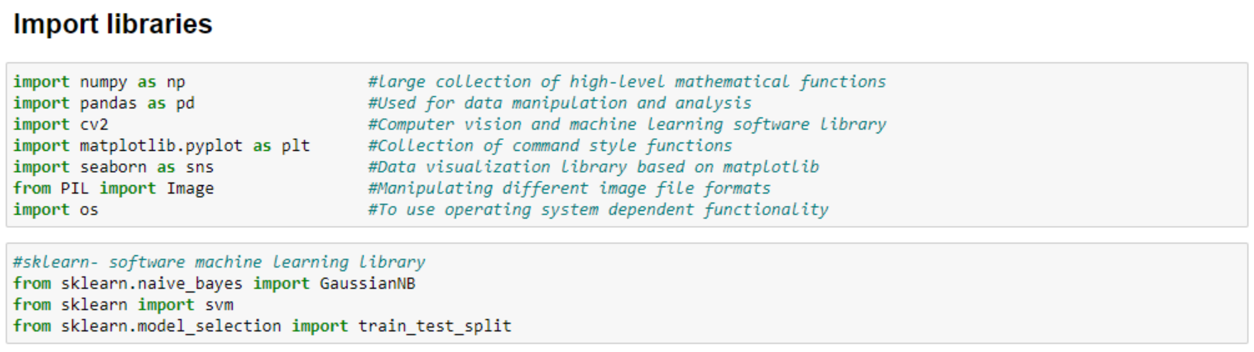
CONTENTS

Introduction

Among all fatal diseases, malaria is considered as the most prevalent. Medical practitioners conduct different surveys on cells images. In this study, Malaria Prediction Analysis is developed using Naïve Bayes and Support Vector Machine algorithms.

The code predicts the likelihood of having malaria with the help of cell images. It helps us establish a relationship between medical factors related to malaria and form patterns. Through this project we aim to exploit data mining techniques on cell images to assist in the prediction of the malaria.

Libraries Used

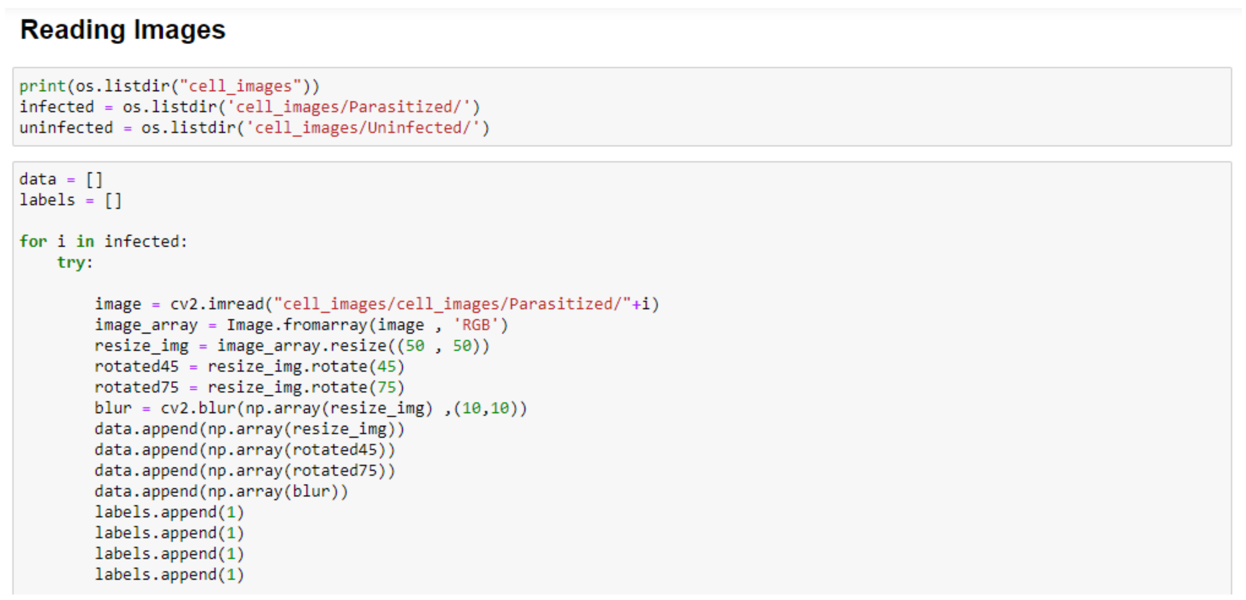


Import libraries

```
import numpy as np                 #Large collection of high-level mathematical functions
import pandas as pd                #Used for data manipulation and analysis
import cv2                         #Computer vision and machine learning software library
import matplotlib.pyplot as plt    #Collection of command style functions
import seaborn as sns              #Data visualization library based on matplotlib
from PIL import Image              #Manipulating different image file formats
import os                          #To use operating system dependent functionality
```

```
#sklearn- software machine learning library
from sklearn.naive_bayes import GaussianNB
from sklearn import svm
from sklearn.model_selection import train_test_split
```

Loading Dataset



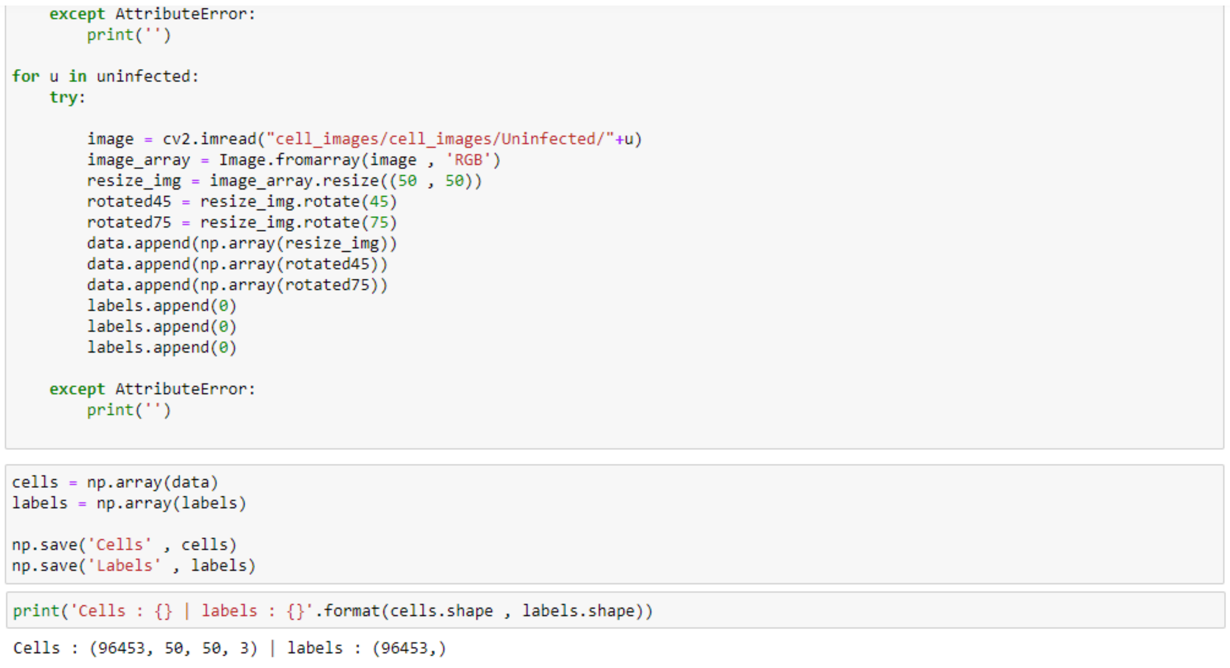
Reading Images

```
print(os.listdir("cell_images"))
infected = os.listdir('cell_images/Parasitized/')
uninfected = os.listdir('cell_images/Uninfected/')
```

```
data = []
labels = []

for i in infected:
    try:

        image = cv2.imread("cell_images/cell_images/Parasitized/"+i)
        image_array = Image.fromarray(image , 'RGB')
        resize_img = image_array.resize((50 , 50))
        rotated45 = resize_img.rotate(45)
        rotated75 = resize_img.rotate(75)
        blur = cv2.blur(np.array(resize_img) ,(10,10))
        data.append(np.array(resize_img))
        data.append(np.array(rotated45))
        data.append(np.array(rotated75))
        data.append(np.array(blur))
        labels.append(1)
        labels.append(1)
        labels.append(1)
        labels.append(1)
```



```
    except AttributeError:
        print('')

for u in uninfected:
    try:

        image = cv2.imread("cell_images/cell_images/Uninfected/"+u)
        image_array = Image.fromarray(image , 'RGB')
        resize_img = image_array.resize((50 , 50))
        rotated45 = resize_img.rotate(45)
        rotated75 = resize_img.rotate(75)
        data.append(np.array(resize_img))
        data.append(np.array(rotated45))
        data.append(np.array(rotated75))
        labels.append(0)
        labels.append(0)
        labels.append(0)

    except AttributeError:
        print('')
```

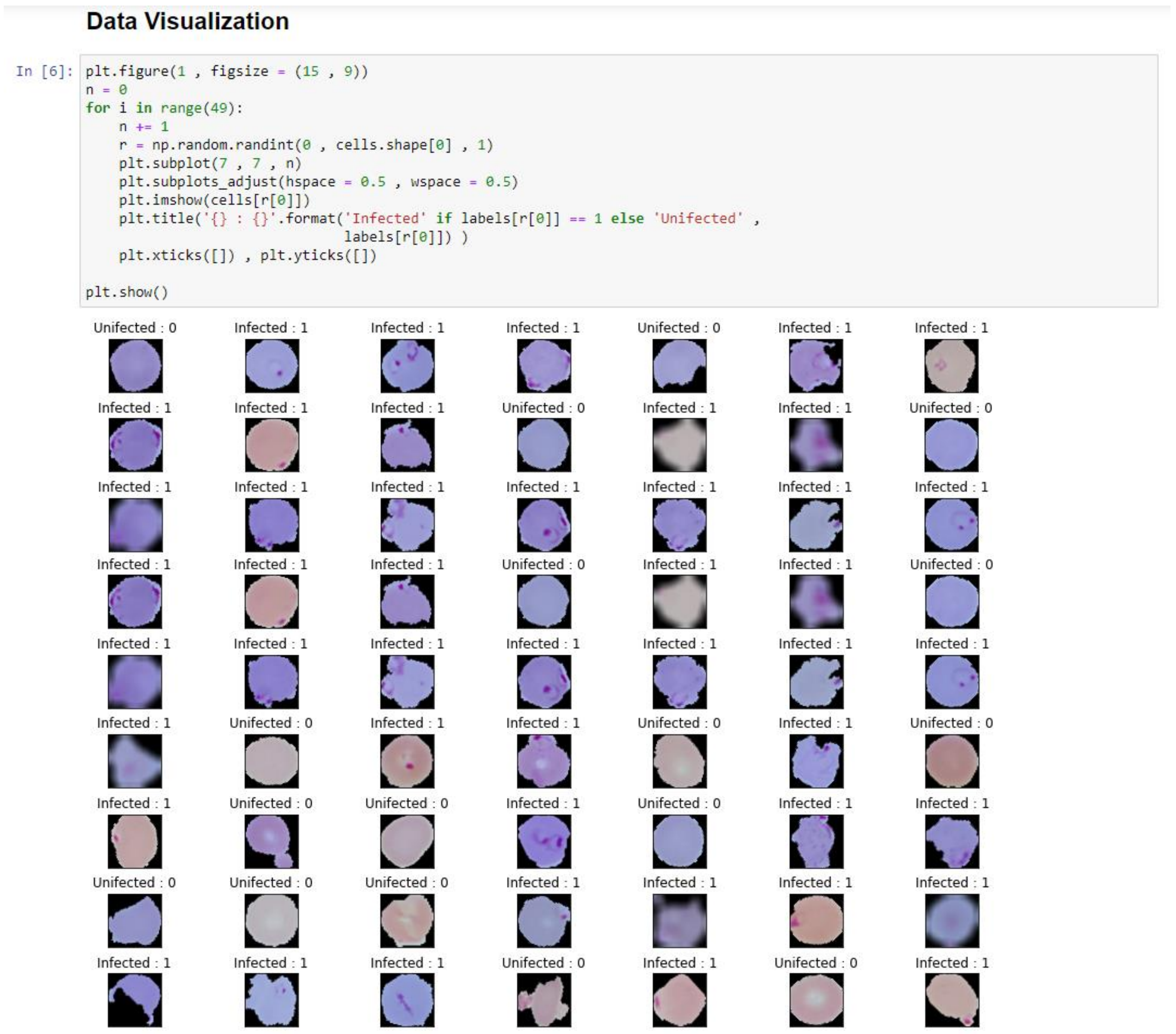
```
cells = np.array(data)
labels = np.array(labels)

np.save('Cells' , cells)
np.save('Labels' , labels)
```

```
print('Cells : {} | labels : {}'.format(cells.shape , labels.shape))
```

```
Cells : (96453, 50, 50, 3) | labels : (96453,)
```

Data Visualization



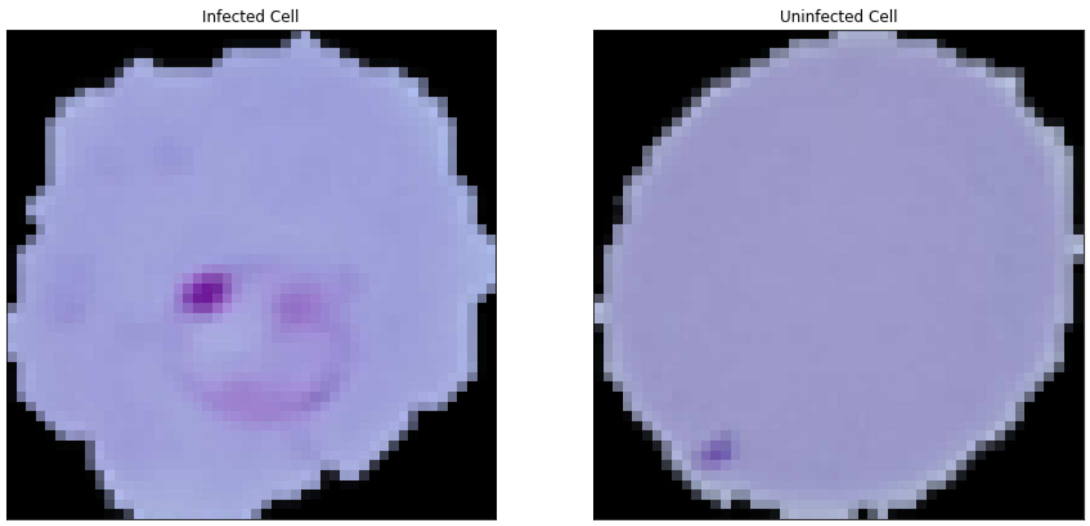
Data Visualization

```
In [6]: plt.figure(1 , figsize = (15 , 9))
        n = 0
        for i in range(49):
            n += 1
            r = np.random.randint(0 , cells.shape[0] , 1)
            plt.subplot(7 , 7 , n)
            plt.subplots_adjust(hspace = 0.5 , wspace = 0.5)
            plt.imshow(cells[r[0]])
            plt.title('{} : {}'.format('Infected' if labels[r[0]] == 1 else 'Unifected' ,
                                       labels[r[0]]) )
            plt.xticks([]) , plt.yticks([])

        plt.show()
plt.figure(1, figsize = (15 , 7))
plt.subplot(1 , 2 , 1)
plt.imshow(cells[0])
plt.title('Infected Cell')
plt.xticks([]) , plt.yticks([])

plt.subplot(1 , 2 , 2)
plt.imshow(cells[60000])
plt.title('Uninfected Cell')
plt.xticks([]) , plt.yticks([])

plt.show()
```



Data Shuffling

```
n = np.arange(cells.shape[0])
np.random.shuffle(n)
cells = cells[n]
labels = labels[n]
```

```
cells = cells.astype(np.float32)
labels = labels.astype(np.int32)
cells = cells/255
```

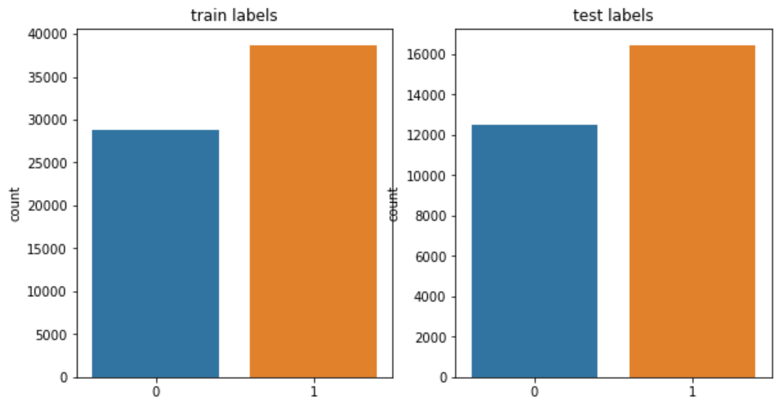
Data Splitting

Spliting into training and testing

```
from sklearn.model_selection import train_test_split

train_x , x , train_y , y = train_test_split(cells , labels ,test_size = 0.3 , random_state = 111)
```

```
plt.figure(1 , figsize = (15 ,5))
n = 0
for z , j in zip([train_y , y] , ['train labels','test labels']):
    n += 1
    plt.subplot(1 , 3  , n)
    sns.countplot(x = z )
    plt.title(j)
plt.show()
```



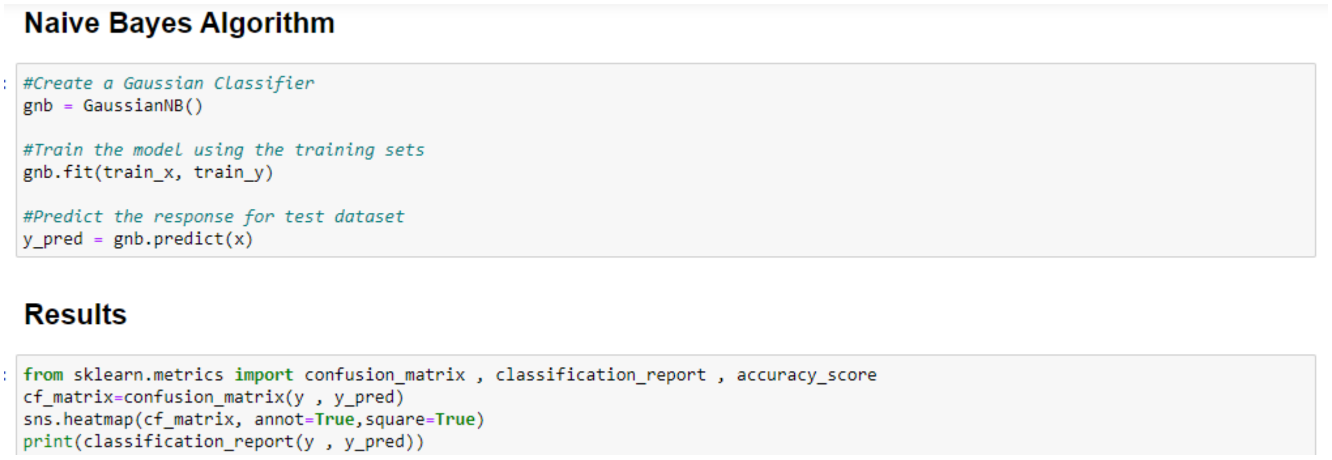
```
print('train data shape {}, test data shape {}'.format(train_x.shape,x.shape))
```

```
train data shape (67517, 50, 50, 3), test data shape (28936, 50, 50, 3)
```

```
print(train_x.shape,x.shape)
train_x = train_x.reshape(67517,50*50*3)
x= x.reshape(28936,50*50*3)
```

```
(67517, 50, 50, 3) (28936, 50, 50, 3)
```

Naïve Bayes Algorithm



Naive Bayes Algorithm

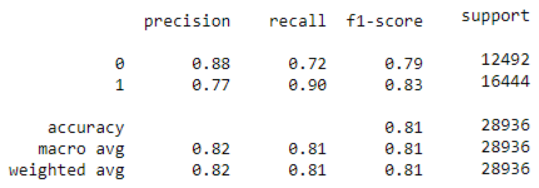
```
#Create a Gaussian Classifier
gnb = GaussianNB()

#Train the model using the training sets
gnb.fit(train_x, train_y)

#Predict the response for test dataset
y_pred = gnb.predict(x)
```

Results

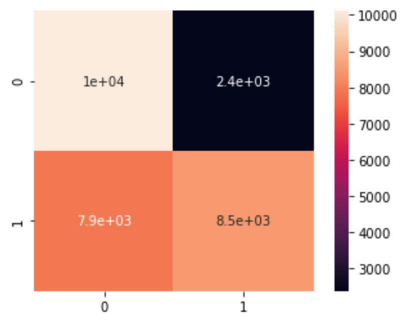
```
from sklearn.metrics import confusion_matrix , classification_report , accuracy_score
cf_matrix=confusion_matrix(y , y_pred)
sns.heatmap(cf_matrix, annot=True,square=True)
print(classification_report(y , y_pred))
```



```
              precision    recall  f1-score   support

           0       0.88      0.72      0.79     12492
           1       0.77      0.90      0.83     16444

    accuracy                           0.81     28936
   macro avg       0.82      0.81      0.81     28936
weighted avg       0.82      0.81      0.81     28936
```



The accuracy of Naïve Bayes Algorithm is 83%.

Support Vector Machine Algorithm

Support Vector Machines

```
#Create a svm Classifier
clf = svm.SVC(kernel='linear') # Linear Kernel

#Train the model using the training sets
clf.fit(train_x, train_y)

#Predict the response for test dataset
y_pred = clf.predict(x)
```

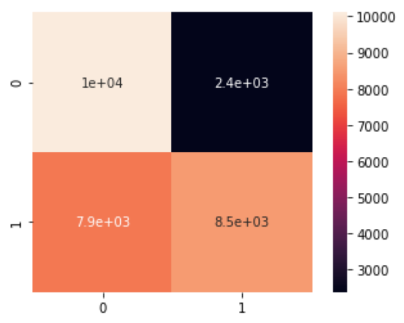
Results

```
from sklearn.metrics import confusion_matrix , classification_report , accuracy_score
cf_matrix=confusion_matrix(y , y_pred)
sns.heatmap(cf_matrix, annot=True,square=True)
print(classification_report(y , y_pred))
```

```
              precision    recall  f1-score   support

           0       0.92      0.74      0.82     12492
           1       0.78      0.93      0.85     16444

    accuracy                           0.84     28936
   macro avg       0.85      0.84      0.83     28936
weighted avg       0.85      0.84      0.83     28936
```

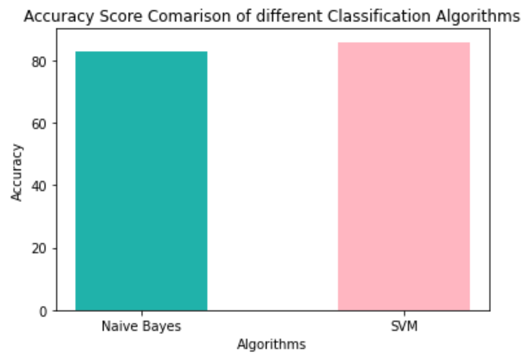


The accuracy of SVM Algorithm is 86%.

Algorithm Comparison

```
algos = ["Naive Bayes", "SVM"]
scores = [score1,score2]
```

```
plt.bar(algos, scores, width=0.5,color = ['lightseagreen', 'lightpink'])
plt.title("Accuracy Score Comarison of different Classification Algorithms")
plt.xlabel('Algorithms')
plt.ylabel('Accuracy')
plt.show()
```



Conclusion

Proposed project is user-friendly, scalable, reliable and an expandable analysis which can also help in reducing treatment costs by providing Initial diagnostics in time. The model can also serve the purpose of training tool for medical students and will be a soft diagnostic tool.

There are many possible improvements that could be explored to improve the scalability and accuracy of this prediction system. As we have developed a generalized system, in future we can use this system for the analysis of different data sets.

The performance of the diagnosis can be improved significantly by handling numerous class labels in the prediction process, and it can be another positive direction of research.