```
import pandas as pd
import numpy as np
import random


def main():

        df1=pd.read_csv("SPECT.csv")
        df1=df1.values
        #shuffle the training examples
        np.random.shuffle(df1)
        X=df1[:,1:]
        Y=df1[:,0]
        nx=X.shape[1]
        m=X.shape[0]
        Y=Y.reshape((m,1))
        test_size=m//10
        #ex_part=(size//10)+1;

        print("Number Of features : ",nx,"\nNumber of training examples : ",m)

        print("Using plain Naive Bayes")
        select_all_features = [1]*nx
        fitness = NB(X,Y,select_all_features)
        print("The fitness is ",fitness,"\n\n")


        population=[]
        for i in range(30):
                chromosome = [0]*nx
                for k in range(nx):
                        t = random.randint(0,1)
                        chromosome[k]=t
                population.append(chromosome)

        #print(population)
        print("Using Naive Bayes with Genetic Algorithm")
        absolute_best_chromosome=[0]*nx
        absolute_best_fitness=0

        for iteration in range(50):
                fitnesses  = [None]*30
                fitness_sum = 0
                for i in range(30):
                        fitness = NB(X,Y,population[i])
                        if fitness>absolute_best_fitness:
                                absolute_best_fitness=fitness
                                absolute_best_chromosome=population[i].copy()
                        #print("chromosome is : ",population[i],"\nfitness is : ",fitness,"
\n\n")
                        fitnesses[i]=fitness
                        fitness_sum += fitness

                #print("Average fitness after iteration ",iteration," is : ",fitness_sum/30
)
                print("Running iteration : ",iteration)

                #making prob and cumulative prob arrays
                prob = [None]*30
                cum_prob = [None]*30
                for i in range(30):
                        prob[i] = fitnesses[i]/fitness_sum
                        if i == 0:
                                cum_prob[0]=prob[0]
                        else:
                                cum_prob[i]=cum_prob[i-1]+prob[i]

                #selection step
```

```
                new_population=[]
                for i in range(30):
                        random_num = random.uniform(0, 1)#generate random number between 0
and 1
                        for j in range(len(cum_prob)):
                                if cum_prob[j]>random_num:
                                        new_population.append(population[j])
                                        break
                population = new_population.copy()

                #crossover
                used_indices = [0]*30
                for i in range(40):
                        chromosome1_index = random.randint(0,29)
                        chromosome2_index = random.randint(0,29)

                        if(used_indices[chromosome1_index]==1 or used_indices[chromosome2_i
ndex]==1):
                                continue
                        else:
                                used_indices[chromosome1_index]=1
                                used_indices[chromosome2_index]=1

                        chromosome1 = population[chromosome1_index]
                        chromosome2 = population[chromosome2_index]

                        cross_index_start = nx - nx//4

                        for j in range(cross_index_start,nx):
                                temp = chromosome1[j]
                                chromosome1[j] = chromosome2[j]
                                chromosome2[j] = temp

                #Mutation - assuming all chromosomes are mutated and a mutation rate of 10%
                for i in range(30):
                        for k in range(nx//10):
                                gene_num = random.randint(0,nx-1)
                                population[i][gene_num] = (population[i][gene_num] + 1)%2
                                #switching 0's and 1's

        print("The Best chromosome is ",absolute_best_chromosome)
        print("The Best fitness is ",absolute_best_fitness)



                #print(chromosome)


def NB(X,Y,chromosome):
        nx=X.shape[1]
        m=X.shape[0]
        Y=Y.reshape((m,1))
        test_size=m//10
        #ex_part=(size//10)+1;

        X_new = np.zeros((m,1))

        for k in range(len(chromosome)):
                if(chromosome[k]==1):
                        X_new = np.concatenate((X_new,X[:,k].reshape((m,1))),axis=1)

        X = X_new[:,1:]
        nx=X.shape[1]

        #print("Number of features chosen is : ",nx)

        accuracy=0
```

```
        for fold in range(10):
                tp,tn,fp,fn=0,0,0,0
                X_test = X[(fold*test_size):((fold+1)*test_size),:]
                X_train = np.concatenate((X[0:(fold*test_size),:],X[(fold+1)*test_size:m,:]
),axis=0)
                Y_test = Y[(fold*test_size):((fold+1)*test_size),:]
                Y_train = np.concatenate((Y[0:(fold*test_size),:],Y[(fold+1)*test_size:m,:]
),axis=0)


                m_test,m_train = X_test.shape[0],X_train.shape[0]

                #print("Testing and training sizes are : ",m_train," ",m_test)

                total_yes=0
                total_no=0
                for i in range(m_train):
                        if(Y_train[i][0]=='Yes'):
                                total_yes+=1
                        else:
                                total_no+=1

                prob_yes = total_yes/m_train
                prob_no = total_no/m_train

                #print("Total yes and no : ",total_yes," ",total_no)

                prob_yes_1,prob_no_1,prob_yes_0,prob_no_0 = [0]*nx,[0]*nx,[0]*nx,[0]*nx

                for i in range(m_train):

                        if Y_train[i]=='Yes':
                                for k in range(nx):
                                        if(X_train[i][k]==1):
                                                prob_yes_1[k]+=1
                                        else:
                                                prob_yes_0[k]+=1

                        else:
                                for k in range(nx):
                                        if(X_train[i][k]==1):
                                                prob_no_1[k]+=1
                                        else:
                                                prob_no_0[k]+=1


                '''if fold == 0:
                        print(prob_yes_1,"\n",prob_yes_0,"\n",prob_no_1,"\n",prob_no_0)'''

                for k in range(nx):
                        prob_yes_1[k] = prob_yes_1[k]/prob_yes
                        prob_yes_0[k] = prob_yes_0[k]/prob_yes
                        prob_no_1[k] = prob_no_1[k]/prob_no
                        prob_no_0[k] = prob_no_0[k]/prob_no

                '''if fold == 0:
                        print(prob_yes_1,"\n",prob_yes_0,"\n",prob_no_1,"\n",prob_no_0)'''



                for i in range(m_test):
                        prob_yes_given_features=prob_yes
                        prob_no_given_features=prob_no

                        for k in range(nx):
                                if(X_test[i][k]==1):
                                        prob_yes_given_features *= prob_yes_1[k]
```

```
                                        prob_no_given_features *= prob_no_1[k]
                                else:
                                        prob_yes_given_features *= prob_yes_0[k]
                                        prob_no_given_features *= prob_no_0[k]

                        '''if fold == 0 and i==0:
                                print(prob_yes_given_features,"\n",prob_no_given_features)'
''

                        pred_class = 'Yes'
                        if(prob_yes_given_features<prob_no_given_features):
                                pred_class='No'

                        if(Y_test[i]=='Yes' and pred_class=='Yes'):
                                tp+=1
                        elif(Y_test[i]=='Yes' and pred_class=='No'):
                                fn+=1
                        elif(Y_test[i]=='No' and pred_class=='No'):
                                tn+=1
                        else:
                                fp+=1

                accuracy += (tp+tn)/(tp+tn+fp+fn)
        '''precision = tp/(tp + fp)
        recall = tp/(tp + fn)'''

        #print("Accuracy : ",accuracy)
        return (accuracy/10)

if __name__=='__main__':
        main()
```