Jovenes_Adultos

March 1, 2017

1 Jovenes/Adultos:

Una estación de radio busca clasificar a su audiencia en jovenes y adultos a partir de sus gustos musicales.

Para esto la estación realizo una encuesta la cual consistía en indicar si les gustaba cierta agrupación (valor igual a 1) o no (valor igual a 0) y se obtuvieron los siguiente resultado:

0 = Jovenes 1 = Adultos

2 Usando EMV tenemos lo siguiente:

In [46]:
```
import pandas as pd

data_set = pd.read_csv("data_gustos.csv",sep=',')

print (data_set)
print (list(data_set.columns.values))
```

```
    Pink_Floyd  The_Beatles  R.E.M  Nirvana  Queen  Oasis  Tipo_Audiencia
0            1            0      0        1      1      1               0
1            1            1      0        1      1      0               0
2            1            1      1        0      0      1               0
3            1            0      1        0      0      1               0
4            1            0      0        0      1      0               0
5            1            1      1        0      0      0               0
6            1            1      0        0      1      1               1
7            1            1      1        0      0      1               1
8            1            1      1        1      1      0               1
9            1            1      1        0      1      0               1
10           1            1      1        0      1      1               1
11           1            1      0        1      1      0               1
12           1            1      0        1      0      0               1
['Pink_Floyd', 'The_Beatles', 'R.E.M', 'Nirvana', 'Queen', 'Oasis', 'Tipo_Audiencia
```

In [47]:
```
"""
Separamos el conjunto de datos primero para Jovenes cuyo valor en el tipo_
igual a cero y los adultos es igual a 1
```

```
        """
        data_set_j=data_set[data_set['Tipo_Audiencia']==0]
        data_set_a=data_set[data_set['Tipo_Audiencia']==1]

        data_set_j = data_set_j.ix[:,0:6]
        data_set_a = data_set_a.ix[:,0:6]

        print "========Jovenes========"
        print (data_set_j)
        print "========Adultos========"
        print (data_set_a)
```

```
========Jovenes========
   Pink_Floyd  The_Beatles  R.E.M  Nirvana  Queen  Oasis
0           1            0      0        1      1      1
1           1            1      0        1      1      0
2           1            1      1        0      0      1
3           1            0      1        0      0      1
4           1            0      0        0      1      0
5           1            1      1        0      0      0
========Adultos========
    Pink_Floyd  The_Beatles  R.E.M  Nirvana  Queen  Oasis
6            1            1      0        0      1      1
7            1            1      1        0      0      1
8            1            1      1        1      1      0
9            1            1      1        0      1      0
10           1            1      1        0      1      1
11           1            1      0        1      1      0
12           1            1      0        1      0      0
```

In [48]:
```
        #Se obtiene las cabeceras para despues iterar sobre el data_set
        cabeceras=list(data_set_j.columns.values)
        #print cabeceras


        #Se calcula el promedio para cada una de nuestras variables con respecto a
        q_i_j=[]
        q_i_a=[]
        for cabecera in cabeceras:
            promedio_j = data_set_j[cabecera].mean()
            promedio_a = data_set_a[cabecera].mean()
            q_i_j.append(promedio_j)
            q_i_a.append(promedio_a)

        #print q_i_j
        #print q_i_a
```

```
        total_audiencia = total_audiencia=data_set.shape[0]
        total_j = data_set_j.shape[0]
        total_a = data_set_a.shape[0]

        print ("Las cantidades por cada clase son las siguientes:")
        print total_j
        print total_a
        print total_audiencia

        #Se calcula el parametro para la clase.
        q_c_j = float(total_j)/float(total_audiencia)
        q_c_a = float(total_a)/float(total_audiencia)
```

```
Las cantidades por cada clase son las siguientes:
6
7
13
```

```
In [49]: def clasificador_bayes(x_i):
             b_q_j_l=[]
             b_q_a_l=[]
             for ind in range(len(x_i)):
                 b_q_j=(q_i_j[ind]**x_i[ind])*((1-q_i_j[ind])**(1-x_i[ind]))
                 b_q_a=(q_i_a[ind]**x_i[ind])*((1-q_i_a[ind])**(1-x_i[ind]))
                 b_q_j_l.append(b_q_j)
                 b_q_a_l.append(b_q_a)
             #print b_q_j_l,b_q_a_l
             #print (reduce(lambda x,y: x * y,b_q_j_l,1))
             #print (reduce(lambda x,y: x * y,b_q_a_l,1))
             pr_j=q_c_j*(reduce(lambda x,y: x * y,b_q_j_l,1))
             pr_a=q_c_a*(reduce(lambda x,y: x * y,b_q_a_l,1))
             #print "Las proabilidades son"
             print "Pr_J_0: ",pr_j
             print "Pr_A_1: ",pr_a
             if pr_j>pr_a:
                 print pr_j,"J-0"
             elif pr_j<pr_a:
                 print pr_a,"A-1"
             else:
                 print "Las probabilidades son iguales, no se puede determinar la c
             print

         def main():
             x_1 = [1, 1, 0, 1, 1, 0]#1-1
             x_2 = [1, 0, 1, 1, 1, 1]#0-0
             x_3 = [1, 1, 0, 0, 0, 0]#1-1
             x_4 = [1, 1, 1, 1, 1, 1]#1-1
```

```
            x_5 = [0, 1, 1, 1, 1, 1]#1 La primer entrada en este vector
            clasificador_bayes(x_1)
            clasificador_bayes(x_2)
            clasificador_bayes(x_3)
            clasificador_bayes(x_4)
            clasificador_bayes(x_5)

        main()
```

```
Pr_J_0:  0.00961538461538
Pr_A_1:  0.0403677954698
0.0403677954698 A-1

Pr_J_0:  0.00961538461538
Pr_A_1:  0.0
0.00961538461538 J-0

Pr_J_0:  0.0192307692308
Pr_A_1:  0.0215294909172
0.0215294909172 A-1

Pr_J_0:  0.00961538461538
Pr_A_1:  0.0403677954698
0.0403677954698 A-1

Pr_J_0:  0.0
Pr_A_1:  0.0
Las probabilidades son iguales, no se puede determinar la clase
```

Lo que podemos ver utilizando que utilizando EMV es que para el ultimo vector no se puede determinar la clase a la que pertenece y esto se debe a la primer entrada de nuestro vector a clasificar la cual es 0 pero en la variable correspondiente que tiene que ver con la banda de Pink_Floyd solo existen valores de 1 entonces cuando se quiere calcular la proabilidad condicional esta es cero para cualquiera de las dos clases. Este caso cae en lo comentado en clase con la Teoría del cisne negro por lo tanto nuestro clasificador no esta generalizando. Esto tambien se debe en gran medida a que el conjunto de datos que tenemos es muy pequeño.

3 Usando MAP tenemos lo siguiente:

Ahora vamos a implementar utilizando el estimador de Máximo a Posteriori.

```
In [50]: """
         Separamos el conjunto de datos primero para Jovenes cuyo valor en el tipo_
         igual a cero y los adultos es igual a 1
         """
```

```
        data_set_j=data_set[data_set['Tipo_Audiencia']==0]
        data_set_a=data_set[data_set['Tipo_Audiencia']==1]

        data_set_j = data_set_j.ix[:,0:6]
        data_set_a = data_set_a.ix[:,0:6]

        print "========Jovenes========"
        print (data_set_j)
        print "========Adultos========"
        print (data_set_a)
```

```
========Jovenes========
   Pink_Floyd  The_Beatles  R.E.M  Nirvana  Queen  Oasis
0           1            0      0        1      1      1
1           1            1      0        1      1      0
2           1            1      1        0      0      1
3           1            0      1        0      0      1
4           1            0      0        0      1      0
5           1            1      1        0      0      0
========Adultos========
    Pink_Floyd  The_Beatles  R.E.M  Nirvana  Queen  Oasis
6            1            1      0        0      1      1
7            1            1      1        0      0      1
8            1            1      1        1      1      0
9            1            1      1        0      1      0
10           1            1      1        0      1      1
11           1            1      0        1      1      0
12           1            1      0        1      0      0
```

In [51]:
```
        #Se obtiene las cabeceras para despues iterar sobre el data_set
        cabeceras=list(data_set_j.columns.values)
        #print cabeceras

        alpha = 1
        beta = 2
        #Se calcula el promedio para cada una de nuestras variables con respecto a
        q_i_j=[]
        q_i_a=[]


        total_audiencia = total_audiencia=data_set.shape[0]
        total_j = data_set_j.shape[0]
        total_a = data_set_a.shape[0]

        for cabecera in cabeceras:
            sum_j = data_set_j[cabecera].sum()+(alpha-1)
            sum_a = data_set_a[cabecera].sum()+(alpha-1)
```

```
    q_i_j.append(float(sum_j)/float(total_j+beta+alpha-2))
    q_i_a.append(float(sum_a)/float(total_a+beta+alpha-2))

#print q_i_j
#print q_i_a


#Se calcula el parametro para la clase.
q_c_j = float(total_j+alpha-1)/float(total_audiencia+beta+alpha-2)
q_c_a = float(total_a+alpha-1)/float(total_audiencia+beta+alpha-2)


def clasificador_bayes_map(x_i):
    b_q_j_l=[]
    b_q_a_l=[]
    for ind in range(len(x_i)):
        b_q_j=(q_i_j[ind]**x_i[ind])*((1-q_i_j[ind])**(1-x_i[ind]))
        b_q_a=(q_i_a[ind]**x_i[ind])*((1-q_i_a[ind])**(1-x_i[ind]))
        b_q_j_l.append(b_q_j)
        b_q_a_l.append(b_q_a)
    #print b_q_j_l,b_q_a_l
    #print (reduce(lambda x,y: x * y,b_q_j_l,1))
    #print (reduce(lambda x,y: x * y,b_q_a_l,1))
    pr_j=q_c_j*(reduce(lambda x,y: x * y,b_q_j_l,1))
    pr_a=q_c_a*(reduce(lambda x,y: x * y,b_q_a_l,1))
    print "Pr_J_0: ",pr_j
    print "Pr_A_1: ",pr_a
    if pr_j>pr_a:
        print pr_j,"J-0"
    elif pr_j<pr_a:
        print pr_a,"A-1"
    else:
        print "Las probabilidades son iguales, no se puede determinar la c
    print

def main():
    x_1 = [1, 1, 0, 1, 1, 0]#1-1 - 1
    x_2 = [1, 0, 1, 1, 1, 1]#0-0 - 0
    x_3 = [1, 1, 0, 0, 0, 0]#1-1 - 1
    x_4 = [1, 1, 1, 1, 1, 1]#1-1  - 1
    x_5 = [0, 1, 1, 1, 1, 1]#1- No se puede determinar - 1
    clasificador_bayes_map(x_1)
    clasificador_bayes_map(x_2)
    clasificador_bayes_map(x_3)
    clasificador_bayes_map(x_4)
    clasificador_bayes_map(x_5)

main()
```

```
Pr_J_0:  0.00629475327943
Pr_A_1:  0.0280380249023
0.0280380249023 A-1

Pr_J_0:  0.00472106495957
Pr_A_1:  0.00240325927734
0.00472106495957 J-0

Pr_J_0:  0.0209825109314
Pr_A_1:  0.0280380249023
0.0280380249023 A-1

Pr_J_0:  0.00354079871968
Pr_A_1:  0.0168228149414
0.0168228149414 A-1

Pr_J_0:  0.000590133119946
Pr_A_1:  0.00240325927734
0.00240325927734 A-1
```

Lo que nos podemos dar cuenta que utilizando MAP nuestro clasificador puede asignar a cada uno de los nuevos vectores su clase correspondiente con respeco a EMV.

4 scikit-learn:

Ahora lo que vamos a ver es el paquete de scikit-learn y utilizar el clasificador bayesiano que esta implementado en este paquete y compararlo contra nuestros clasificadores implementados.

```
In [55]: import pandas as pd

         data_set = pd.read_csv("data_gustos.csv",sep=',')
         X = data_set.ix[:,0:6].as_matrix()

         Y= [0,0,0,0,0,0,1,1,1,1,1,1,1]

         from sklearn.naive_bayes import BernoulliNB
         clf = BernoulliNB()
         clf.fit(X, Y)

         print clf.predict([[1, 1, 0, 1, 1, 0]])
         print clf.predict([[1, 0, 1, 1, 1, 1]])
         print clf.predict([[1, 1, 0, 0, 0, 0]])
         print clf.predict([[1, 1, 1, 1, 1, 1]])
         print clf.predict([[0, 1, 1, 1, 1, 1]])

         predicciones = clf.predict(X)
```

```
        predicciones = predicciones.astype(int)

        aciertos=0.0
        total_datos= float(data_set.shape[0])
        print total_datos

        for ind in range(len(predicciones)):
            if int(predicciones[ind])==int(Y[ind]):
                aciertos+=1


        #El porcentaje de aciertos es el siguiente:
        print "El porcentaje de aciertos es el siguiente:"
        print str((aciertos/total_datos)*100)+"%"
```

```
[1]
[0]
[1]
[1]
[1]
13.0
El porcentaje de aciertos es el siguiente:
76.9230769231%
```

Ahora vamos a usar un clasificador donde el tipo de la distribución es normal para de igual forma poder comparar con nuestros clasificadores.

```
In [56]: import pandas as pd

        data_set = pd.read_csv("data_gustos.csv",sep=',')
        X = data_set.ix[:,0:6].as_matrix()

        Y= [0,0,0,0,0,0,1,1,1,1,1,1,1]

        from sklearn.naive_bayes import GaussianNB
        clf = GaussianNB()
        clf.fit(X, Y)

        print "Las clasificaciones de los nuevos vectores son las siguientes:"
        print clf.predict([[1, 1, 0, 1, 1, 0]])
        print clf.predict([[1, 0, 1, 1, 1, 1]])
        print clf.predict([[1, 1, 0, 0, 0, 0]])
        print clf.predict([[1, 1, 1, 1, 1, 1]])
        print clf.predict([[0, 1, 1, 1, 1, 1]])

        predicciones = clf.predict(X)
```

```
        predicciones = predicciones.astype(int)

        aciertos=0.0
        total_datos= float(data_set.shape[0])
        print total_datos

        for ind in range(len(predicciones)):
            if int(predicciones[ind])==int(Y[ind]):
                aciertos+=1


        #El porcentaje de aciertos es el siguiente:
        print "El porcentaje de aciertos es el siguiente:"
        print str((aciertos/total_datos)*100)+"%"
```

```
Las clasificaciones de los nuevos vectores son las siguientes:
[1]
[0]
[1]
[1]
[1]
13.0
El porcentaje de aciertos es el siguiente:
76.9230769231%
```

Notamos que el porcentaje de aciertos de nuestro clasificadores usando scikit-learn es el mismo esto en gran medida se debe a que no tenemos una gran cantidad de datos.

5 Conclusiones:

Para nuestro conjunto de vectores a clasificar obtuvimos los siguientes resultados: * Mi_Clasificador_Bernoulli_EMV= [1,0,1,1,"No se determino"] * Mi_Clasificador_Bernoulli_MAP= [1,0,1,1,1] * scikit-learn_Bernoulli= [1,0,1,1,1] * scikit-learn_Gaussian= [1,0,1,1,1]

En general podemos concluir que los resultados que obtenemos por parte de nuestras implementaciones es similar a los genderdos por el paquete de scikit-learn. Pero con respecto al rendimiento de nuestro clasificador nos muy alto en gran medida por la poca cantidad de datos que tenemos es por lo que nuestro clasificador no puede generalizar.

```
In [ ]:
```