C?ncer de seno

March 1, 2017

1 Cáncer de seno

Descripción:

Entrena distintos clasificadores de tumores de seno usando clasificadores bayesianos ingenuos con distintas distribuciones a partir de la base de datos de cáncer de seno de Wisconsin.Esta base de datos contiene 699 registros de tumores de seno, de los cuales 458 son benignos y 241 son malignos.

```
In [14]: %matplotlib notebook
         import matplotlib.pyplot as plt
         import numpy as np
         import pandas as pd
         from sklearn.naive_bayes import MultinomialNB,GaussianNB
         from pandas.tools.plotting import scatter_matrix
         from sklearn import preprocessing

         #Carga de los datos:
         data_set = pd.read_csv("breast_cancer_wisconsin.data",delimiter=",",header
```

1.0.1 Descripción de las variables

- 0 - Codigo de la muestra (ID)
- 1 - Grosor del tumor.
- 2 - Uniformidad del tamaño de la célula.
- 3 - Uniformidad de la forma de la célula.
- 4 - Adhesión marginal.
- 5 - Tamaño de célula epitelial.
- 6 - Núcleos desnudos.
- 7 - Cromatina blanda.
- 8 - Nucléodos normales.
- 9 - Mitosis de células.
- 10- Clase (2:Benigno,4:Maligno)

```
In [15]: # Mostramos los primeros 10 registros de nuestro conjunto
         # de datos

         print data_set.head(10)
```

```
         0   1   2   3   4   5   6   7   8   9   10
0  1000025   5   1   1   1   2   1   3   1   1   2
1  1002945   5   4   4   5   7  10   3   2   1   2
2  1015425   3   1   1   1   2   2   3   1   1   2
3  1016277   6   8   8   1   3   4   3   7   1   2
4  1017023   4   1   1   3   2   1   3   1   1   2
5  1017122   8  10  10   8   7  10   9   7   1   4
6  1018099   1   1   1   1   2  10   3   1   1   2
7  1018561   2   1   2   1   2   1   3   1   1   2
8  1033078   2   1   1   1   2   1   1   1   5   2
9  1033078   4   2   1   1   2   1   2   1   1   2
```

```
In [16]: print "El tamaño original de nuestro conjunto de datos es:"
         print data_set.shape

         #Primero lo que haremos sera remplazar el valor de '?' por el valor de NaN
         data_set_reemplazo = data_set.replace('?',np.nan)

         # Se eliminan los registros que tienen algun Valor NAn
         data_set_sin_nan = data_set_reemplazo.dropna()
         # Obtengo los registros que elimine para ver como son.
         only_na = data_set_reemplazo[~data_set_reemplazo.index.isin(data_set_sin_n

         print "La cantidad de registros con valores missing son 16:"
         print only_na.shape

         print "Al final nos quedamos con un conjunto de datos:"
         print data_set_sin_nan.shape
```

```
El tamaño original de nuestro conjunto de datos es:
(699, 11)
La cantidad de registros con valores missing son 16:
(16, 11)
Al final nos quedamos con un conjunto de datos:
(683, 11)
```

Ahora las variable que no vamos a considerar para nuestro clasificador sera la variable que tiene el identificador. Todas las variables restantes las vamos a utilizar con el hecho de haber quitado los registros que contenian valores missing.

```
In [17]: X = data_set_sin_nan.ix[:,1:9].as_matrix()
         y = data_set_sin_nan.ix[:,10].as_matrix()

         clf = MultinomialNB(alpha=1.0,class_prior = None,fit_prior = True)
         clf.fit(X, y)

         predicciones = clf.predict(X)
```

```
        predicciones = predicciones.astype(int)

        aciertos=0.0
        total_datos= float(data_set_sin_nan.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)+"%"
```

```
El porcentaje de aciertos es el siguiente:
90.4831625183%
```

```
In [18]: X = data_set_sin_nan.ix[:,1:9].as_matrix()
         y = data_set_sin_nan.ix[:,10].as_matrix()

         clf = GaussianNB()
         clf.fit(X, y)

         predicciones = clf.predict(X)

         predicciones = predicciones.astype(int)

         aciertos=0.0
         total_datos= float(data_set_sin_nan.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)+"%"
```

```
El porcentaje de aciertos es el siguiente:
96.3396778917%
```

Hasta el momento hemos considerado al conjunto de datos quitando los registros que tienen valores faltantes los cuales los podemos visualziar a continuacion:

```
In [19]: print only_na
```

```
           0   1   2   3   4   5    6   7   8   9   10
23   1057013   8   4   5   1   2  NaN   7   3   1   4
40   1096800   6   6   6   9   6  NaN   7   8   1   2
139  1183246   1   1   1   1   1  NaN   2   1   1   2
145  1184840   1   1   3   1   2  NaN   2   1   1   2
158  1193683   1   1   2   1   3  NaN   1   1   1   2
164  1197510   5   1   1   1   2  NaN   3   1   1   2
235  1241232   3   1   4   1   2  NaN   3   1   1   2
249   169356   3   1   1   1   2  NaN   3   1   1   2
275   432809   3   1   3   1   2  NaN   2   1   1   2
292   563649   8   8   8   1   2  NaN   6  10   1   4
294   606140   1   1   1   1   2  NaN   2   1   1   2
297    61634   5   4   3   1   2  NaN   2   3   1   2
315   704168   4   6   5   6   7  NaN   4   9   1   2
321   733639   3   1   1   1   2  NaN   3   1   1   2
411  1238464   1   1   1   1   1  NaN   2   1   1   2
617  1057067   1   1   1   1   1  NaN   1   1   1   2
```

Como podemos observar los valores missing solo se encuentran en la columna que representa la cantidad de núcleos desnudos, una primer aproximación que podriamos proponer es el utilizar el promedio para llenar los valores faltantes considerando que ahora la cantidad registros con valores faltantes representa un 2.2% el cual es medianamente bajo podriamos usar el promedio, pero en caso de que la cantidad de registros con valores faltante sea un porcentaje alto del total podriamos elegir la mediana ya que si en nuetro conjunto de datos tenemos valores outlier en cada una de las variables donde existan valoes missing lo que podria pasar es que nuestro estimador en este caso el promedio tenga un sesgo muy grande por lo tanto se recomendaria usar la mediana.

```
In [20]: print "El calculo del promedio esta dada por lo siguiente"
         promedio_nucleos_desnudos = np.asarray(data_set_sin_nan.iloc[:,6], dtype=n
```

```
El calculo del promedio esta dada por lo siguiente
```

Ahora lo que haremos sera reemplazar el promedio por los valores faltantes y veremos como se comporta nuestro clasificador.

```
In [21]: #
         data_set_reemplazo_promedio = data_set.replace('?',promedio_nucleos_desnuc

         X = data_set_reemplazo_promedio.ix[:,1:9].as_matrix()
         y = data_set_reemplazo_promedio.ix[:,10].as_matrix()


         clf = GaussianNB()
         clf.fit(X, y)
```

```
        predicciones = clf.predict(X)

        predicciones = predicciones.astype(int)

        aciertos=0.0
        total_datos= float(data_set_reemplazo_promedio.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)+"%"
```

```
El porcentaje de aciertos es el siguiente:
95.8512160229%
```

Como nos damos cuenta el rendimiento de nuestro clasificador baja, lo que vamos hacer ahora es usar la mediana para llenar los valores faltantes.

```
In [22]: print "El calculo del promedio esta dada por lo siguiente"
         mediana_nucleos_desnudos = np.median(np.asarray(data_set_sin_nan.iloc[:,6]
         print mediana_nucleos_desnudos
```

```
El calculo del promedio esta dada por lo siguiente
1.0
```

```
In [23]: data_set_reemplazo_mediana = data_set.replace('?',int(mediana_nucleos_desn

         X = data_set_reemplazo_mediana.ix[:,1:9].as_matrix()
         y = data_set_reemplazo_mediana.ix[:,10].as_matrix()


         clf = GaussianNB()
         clf.fit(X, y)

         predicciones = clf.predict(X)

         predicciones = predicciones.astype(int)

         aciertos=0.0
         total_datos= float(data_set_reemplazo_mediana.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)+"%"
```

```
El porcentaje de aciertos es el siguiente:
95.9942775393%
```

2 Conclusiones:

En este caso vemos que la distribución que mejor se adapta a nuestro conjunto de datos es las distribución normal ya que nuestro conjunto se adapta a este tipo de distribución por otra parte al intentar llenar los valores missing nos damos cuenta de que podemos generar ruido en nuestro clasificador si no conocemos bien la distribución de la variable ya que agregando la mediana como el promedio para llenar estos valores el rendimiento de nuestro clasificador bajo.