Trabajo Semana 6 - Naive Bayes Algorithm en Medicina
Asignatura: Machine Learning
Especialización en Inteligencia Artificial
Realizado por: Michael Andrés Mora Poveda

0. Objeto de estudio:

El objetivo de este trabajo es aplicar el algoritmo de **Naive Bayes** para el dataset **Breast Cancer Wisconsin**, con la finalidad de determinar si un paciente tiene un tumor maligno o no (M = malignant, B = benign).

Los siguientes links serán útiles para conocer la estructura de los datasets:

- https://www.kaggle.com/datasets/uciml/breast-cancer-wisconsin-data?resource=download
- https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Diagnostic%29
- https://www.cc.gatech.edu/projects/disl/VISTA/breast.html

Este ejercicio también quiere mostrar la relación y utilidad que pueden proporcionar los algoritmos de machine learning en el sector médico, con la idea de apoyar a los especialistas médicos en temas relacionados con diagnósticos e imágenes relacionadas con enfermedades.

1. Actores y contextos:

El contexto se encuentra en el campo médico de imágenes y diagnósticos médicos relacionados con Oncología, cuya finalidad es estudiar todo lo relacionado con tumores. Los actores involucrados son los médicos que se pueden apoyar en el sistema que soporta el algoritmo ya mencionado para pronósticos más acertados y reduciendo al mismo tiempo los falsos positivos (pacientes que no tienen ningún tumor).

2. Objetivos:

- Realizar análisis exploratorio para ver distribuciones, estadísticas y estado del dataset seleccionado.
- Proceso de data-cleaning y feature engineering para generar una data de calidad para el modelo de entrenamiento.
- Aplicación del Naive Bayes algorithm para clasificación basado en features, los cuales están relacionados con conceptos técnicos de medicina asociados a esta enfermedad.
- Medir a través de la matriz de confusión, precision, recall y accuracy el performance del modelo para determinar su factibilidad

3. Informe final:

Este ejercicio fue realizado en Jupyter Notebook, y se puede encontrar encontrar en el siguiente repositorio de Github:

- Trabajo semana 6 - Repo Github

***Nota**: El análisis exploratorio fue realizado con el framework de Pandas Profiling, el cual es un reporte interactivo con distribuciones, correlaciones y medidas estadísticas por cada uno de los features. Se puede ejecutar en el Jupyter directamente ó se puede generar un archivo html interactivo.*

3.1 Descripción del dataset y análisis exploratorio:

A continuación veremos varios segmentos del código implementado:

Para empezar, vamos a importar las librerías clásicas y el dataset correspondiente:

In [1]:
```
#Se importan las librerías clásicas:
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
import plotly as pty
import plotly.graph_objs as go
from sklearn.preprocessing import StandardScaler
from pandas_profiling import ProfileReport
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
from sklearn.metrics import (accuracy_score, confusion_matrix,
ConfusionMatrixDisplay,f1_score, classification_report)
from mlxtend.plotting import plot_confusion_matrix
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score
```

```
/var/folders/xh/l9ggclrd1fx2brvxlnc_9hv40000gn/T/ipykernel_1294/2318952864.py:10: DeprecationWarning: `import pandas_profiling` is going to be deprecated by April 1st. Please use `import ydata_profiling` instead.
  from pandas_profiling import ProfileReport
```

In [2]:
```
# Establecemos los headers de los features, dado que la data viene sin ellos:
list_columns = ['id number', 'Clump Thickness', 'Uniformity of Cell Size', 'Uniformity of Cell Shape',
                'Marginal Adhesion' , 'Single Epithelial Cell Size', 'Bare Nuclei', 'Bland Chromatin',
                'Normal Nucleoli', 'Mitoses', 'Class']
```

In [3]:
```
#Importamos los archivos contenidos en el folder y chequeamos los datasets:
dfBreastCancer = pd.read_csv('breast-cancer-wisconsin/breast-cancer-wisconsin.data', names = list_columns)
dfBreastCancer.head()
```

Out[3]:

	id number	Clump Thickness	Uniformity of Cell Size	Uniformity of Cell Shape	Marginal Adhesion	Single Epithelial Cell Size	Bare Nuclei	Bland Chromatin	Normal Nucleoli	Mitoses	Class
0	1000025	5	1	1	1	2	1	3	1	1	2
1	1002945	5	4	4	5	7	10	3	2	1	2

Imagen propia

El siguiente ciclo for nos ayuda a visualizar metadata asociada al dataset para mayor entendimiento del ejercicio:

In [4]:
```
#Importamos los archivos contenidos en el folder y chequeamos la metadata:
with open('breast-cancer-wisconsin/breast-cancer-wisconsin.names', 'r') as f:
    lines = f.readlines()
    for line in lines:
        print(line)
        #line
```

```
4. Relevant Information:


   Samples arrive periodically as Dr. Wolberg reports his clinical cases.
   The database therefore reflects this chronological grouping of the data.
   This grouping information appears immediately below, having been removed
   from the data itself:


     Group 1: 367 instances (January 1989)
     Group 2:  70 instances (October 1989)
     Group 3:  31 instances (February 1990)
```

Imagen propia

Las variables técnicas que forman parte del estudio y del traning son las siguientes:

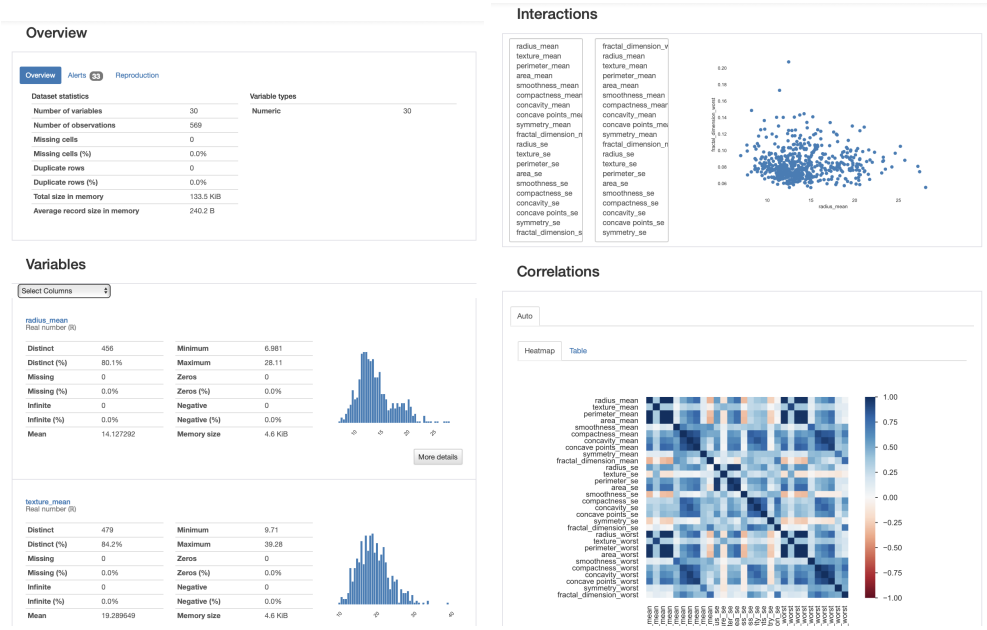
1. Sample code number id
2. Clump Thickness
3. Uniformity of Cell Size
4. Uniformity of Cell Shape
5. Marginal Adhesion
6. Single Epithelial Cell Size
7. Bare Nuclei
8. Bland Chromatin
9. Normal Nucleoli
10. Mitoses
11. Class: (2 for benign, 4 for malignant).

Luego, realizamos algunas validaciones y limpieza del dataset:

```
In [5]: # Ajustamos la columna 'Bare Nuclei' a float dado que su tipo actual es object:
        dfBreastCancer['Bare Nuclei'] = pd.to_numeric(dfBreastCancer['Bare Nuclei'], errors='coerce')

In [6]: # Revisamos el tipo de variable de cada feature:
        dfBreastCancer.info()

        <class 'pandas.core.frame.DataFrame'>
        RangeIndex: 699 entries, 0 to 698
        Data columns (total 11 columns):
         #   Column                       Non-Null Count  Dtype
        ---  ------                       --------------  -----
         0   id number                    699 non-null    int64
         1   Clump Thickness              699 non-null    int64
         2   Uniformity of Cell Size      699 non-null    int64
         3   Uniformity of Cell Shape     699 non-null    int64
         4   Marginal Adhesion            699 non-null    int64
         5   Single Epithelial Cell Size  699 non-null    int64
         6   Bare Nuclei                  683 non-null    float64
         7   Bland Chromatin              699 non-null    int64
         8   Normal Nucleoli              699 non-null    int64
         9   Mitoses                      699 non-null    int64
         10  Class                        699 non-null    int64
        dtypes: float64(1), int64(10)
        memory usage: 60.2 KB

In [7]: # Detectamos valores nulos dentro de los features:
        dfBreastCancer.isnull().sum()

Out[7]: id number                      0
        Clump Thickness                0
        Uniformity of Cell Size        0
        Uniformity of Cell Shape       0
        Marginal Adhesion              0
        Single Epithelial Cell Size    0
        Bare Nuclei                   16
        Bland Chromatin                0
        Normal Nucleoli                0
        Mitoses                        0
        Class                          0
        dtype: int64
```

Imagen propia

Después de estas depuraciones, realizamos la validación de escalas, distribuciones y medidas de tendencia central asociadas a los features:



Imágenes propias

Con este reporte exploratorio generado, confirmamos que sólo 1 columnas necesitó ajustes de valores nulos y que las escalas entre ellas eran bastante similares. Además, normalizamos los features dadas las condiciones del modelo Gaussian Naive Bayes para tener así un mejor performance.

Nota: El reporte interactivo es bastante completo y extenso. Estas imágenes son sólo una muestra.

3.2 Entrenamiento del algoritmo Naive Bayes

A continuación veremos la normalización de los feature y traning del modelo:

2.1 Normalización y training de la data.

Dado que varias de las distribuciones vistas anteriormente no tienen una distribución gaussiana a pesar de tener escalas numéricas similares, realizaremos la normalización de la data para cumplir con este criterio conceptual del Gaussian Naive Bayes Algorithm:

```
In [11]: # Creamos un objeto tipo scaler:
         scaler = StandardScaler()

In [12]: # Generamos la partición de los datos del target y los features:
         X = dfBreastCancer.drop(['Class'], axis = 1)
         y = dfBreastCancer.Class

In [13]: # Normalizamos la data:
         scaler.fit(X)
         X_normalized_data = scaler.transform(X)

In [14]: #Visualizamos la data ya normalizada:
         X_normalized_data

Out[14]: array([[ 0.20693572, -0.69999505, -0.74329904, ..., -0.17966213,
                  -0.61182504, -0.34391178],
                [ 0.20693572,  0.28384518,  0.2668747 , ..., -0.17966213,
                  -0.28411186, -0.34391178],
                [-0.50386559, -0.69999505, -0.74329904, ..., -0.17966213,
                  -0.61182504, -0.34391178],
                ...,
                [ 0.20693572,  2.25152563,  2.28722218, ...,  1.87236122,
                   2.33759359,  0.23956962],
                [-0.14846494,  1.59563215,  0.94032386, ...,  2.69317056,
                   1.02674087, -0.34391178],
                [-0.14846494,  1.59563215,  1.61377302, ...,  2.69317056,
                   0.37131451, -0.34391178]])

In [15]: # Reemplazamos los números de las categorías para los resultados Benigno 2 por 0 y Maligno 4 por 1:
         y = y.replace({2: 0, 4: 1})
         y = y.values
         y

Out[15]: array([0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1,
```

Imagen propia

```
In [16]: # Realizamos el split para el training y testing sets:
         X_train, X_test, y_train, y_test = train_test_split(X_normalized_data, y, test_size = 0.3, random_state = 42)

In [17]: # Creamos una instancia u objeto del algoritmo seleccionado y realizamos el training:
         from sklearn.naive_bayes import GaussianNB
         nb_classifier = GaussianNB()
         nb_classifier.fit(X_train, y_train)

Out[17]: GaussianNB()
         In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
         On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.
```

Imagen propia

En general, este entrenamiento es relativamente rápido dado que la fase inicial de exploración y feature engineering facilita instanciar y entrenar el modelo gaussiano.

3.3 Métricas y performance del modelo:

Al ser un modelo binario de clasificación (si el paciente tiene un tumor maligno o no) calculamos la matriz de confusión, precision, recall y score:

2.2 Revisión de métricas y performance del modelo:

A continuación, revisaremos varias métricas asociadas a modelos de clasificación y corroborar si el performance es bueno o no:

```
In [18]: # Revisamos el score de rendimiento de los data tests:
         print("Naive Bayes score: ", round(nb_classifier.score(X_test, y_test), 3))

         Naive Bayes score:  0.962

In [19]: # Aplicamos el test set de predicciones y vemos el accuracy:
         y_pred = nb_classifier.predict(X_test)
         accuracy = accuracy_score(y_pred, y_test)

In [20]: # Revisamos las métricas asociadas a la matriz de confusión:
         print(classification_report(y_test, y_pred))

                       precision    recall  f1-score   support

                    0       0.99      0.96      0.97       143
                    1       0.92      0.97      0.94        67

             accuracy                           0.96       210
            macro avg       0.95      0.96      0.96       210
         weighted avg       0.96      0.96      0.96       210
```

Imagen propia

In [21]:
```
# Generamos la matriz:
conf_matrix = confusion_matrix(y_test, y_pred)
```

In [22]:
```
# Graficamos la matriz de confusión para una mejor visualización:

%matplotlib inline
#disp = ConfusionMatrixDisplay(confusion_matrix=conf_matrix, display_labels=nb_classifier.classes_)
#disp.plot()
#plt.show()
fig, ax = plot_confusion_matrix(conf_mat=conf_matrix, figsize=(3, 3), cmap=plt.cm.Greens)
plt.xlabel('Predictions', fontsize=18)
plt.ylabel('Actuals', fontsize=18)
plt.title('Confusion Matrix', fontsize=18)
plt.show()
```

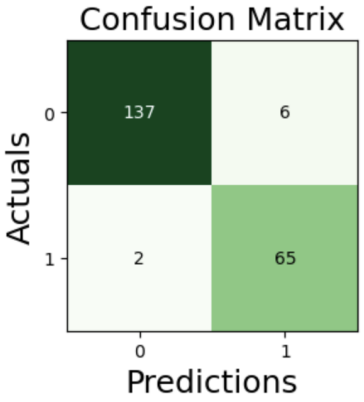


Imagen propia

Conclusiones

- A nivel general este es uno de los algoritmos supervisados más prácticos de aplicar en problemas de clasificación.
- La practicidad que nos da el framework de Pandas profiling es bastante útil para analizar más a fondo las distribuciones y medidas estadísticas de todos los features del dataset, es decir, nos da más tiempo para centrarnos en la solución del problema.
- Las escalas numéricas de los features en general se encontraban bastante similares, sin embargo se aplicó la normalización de las características dado que uno de los principales criterios del algoritmo de Naive Bayes para variables continuas es que estas están distribuidas normalmente.
- Las métricas de precision y recall en general nos indican que el modelo tiene un buen performance para predicción de cáncer de seno. Es esencial tener en cuenta si queremos centrarnos más en el recall (True Positives / (True Positives + False Negatives)) que tiene como meta ser lo más preciso posible tanto para pacientes con y sin cáncer o si en el precision (True Positives / (True Positives + False Positives)) que tiene como meta predecir los mayores casos positivos de cáncer posibles reduciendo al mismo tiempo los falsos positivos.

Referencias:

[1]. Pedregosa, F., Varoquaux, Ga"el, Gramfort, A., Michel, V., Thirion, B., Grisel, O., … others. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12(Oct), 2825–2830. Taken from: https://scikit-learn.org/stable/modules/naive_bayes.html

[2]. Saini, A. Naive Bayes Algorithm: A Complete guide for Data Science Enthusiasts. (2021). Analytics Vidhya .Taken from:https://www.analyticsvidhya.com/blog/2021/09/naive-bayes-algorithm-a-complete-guide-for-data-science-enthusiasts/

[3]. Geeks for Geeks. (2023). Naive Bayes Classifiers. Taken from: https://www.geeksforgeeks.org/naive-bayes-classifiers/

[4]. Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. Computing in Science & Engineering, 9(3), 90–95.

[5]. Sharma, P. Implementation of Gaussian Naive Bayes in Python Sklearn. (2021). Analytics Vidhya .Taken from: https://www.analyticsvidhya.com/blog/2021/11/implementation-of-gaussian-naive-bayes-in-python-sklearn/

[6]. Pedregosa, F., Varoquaux, Ga"el, Gramfort, A., Michel, V., Thirion, B., Grisel, O., … others. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12(Oct), 2825–2830. Taken from: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.ConfusionMatrixDisplay.html

[7]. Kumar, A. (2023). SPython – Draw Confusion Matrix using Matplotlib. Taken from: https://vitalflux.com/python-draw-confusion-matrix-matplotlib/

[8]. Brownlee, J. (2021). How to Use ROC Curves and Precision-Recall Curves for Classification in Python. Taken from: https://machinelearningmastery.com/roc-curves-and-precision-recall-curves-for-classification-in-python/

[9]. Wolberg, W. Wisconsin Breast Cancer dataset. University of Wisconsin Hospitals, Madison. Taken from: https://www.cc.gatech.edu/projects/disl/VISTA/breast.html