Tarea 1 EL4106 - Semestre Otoño 2021

Profesor: Javier Ruiz del Solar
Auxiliar: Patricio Loncomilla

Fecha enunciado: 22 de Marzo de 2021
Plazo entrega tarea: 5 de Abril de 2021

El objetivo de esta tarea es analizar, visualizar y clasificar datos usando las bibliotecas de software pandas, numpy, matplotlib y seaborn, las cuales serán utilizadas en el curso. Se usará el conjunto de datos MAGIC Gamma Telescope Data Set, el cual corresponde a un conjunto de simulaciones de *air showers* generados por rayos gamma primarios v/s hadrones. El conjunto de datos forma parte del *UC Irvine Machine Learning Repository*. El conjunto de datos contiene 10 características, además de una clase, la cual puede ser h (hadrón) o g (no-hadrón). Hay 12332 ejemplos de rayos gamma y 6688 ejemplos de hadrones. El conjunto de datos contiene 11 columnas: las primeras 10 son las características y la última es la clase. Los datos están disponibles tanto en u-cursos como en la página del repositorio.

Se pide realizar lo siguiente:

Parte 1: Visualización y análisis de datos

1) Leer el conjunto de datos magic04.data usando la función pd.read_csv(). Se debe usar los siguientes nombres para las columnas: `['fLength', 'fWidth', 'fSize', 'fConc', 'fConc1', 'fAsym', 'fM3Long', 'fM3Trans', 'fAlpha', 'fDist', 'class']`
2) Separar los datos en un subconjunto con los datos con clase positiva (hadrón) y otro con clase negativa (no-hadrón)
3) Para cada característica, generar histogramas para los datos de la clase negativa y de la clase positiva. Ambos histogramas deben dibujarse superpuestos, es decir, usando un mismo gráfico. Para esto, se recomienda usar plt.hist() con alpha=0.8.
4) Calcular la matriz de correlación de los datos (incluyendo la clase como 0 para no-hadron o 1 para hadrón), usando el valor absoluto de ~~éstas~~las componentes de dicha matriz. Se debe usar como base la función de correlación existente en pandas.
5) Indicar por orden cuáles son las características más correlacionadas con la clase
6) Indicar por orden cuáles son los 5 pares de características más correlacionadas entre sí.
7) Graficar las dos características distintas más correlacionadas en un *scatter plot* (gráfico de dispersión)
8) Indicar por orden cuáles son los 5 pares de características distintas menos correlacionadas entre sí.
9) Graficar las dos características distintas menos correlacionadas en un *scatter plot*
10) Graficar la matriz de correlación de los datos, usando el valor absoluto de éstas. Se debe usar la función sns.heatmap( ) de seaborn

Parte 2: Clasificación usando naive Bayes e histogramas

Se pide programar, entrenar y calibrar un clasificador naive Bayes para determinar en forma probabilística si un candidato corresponde a un pulsar. La regla de decisión del mencionado clasificador está dada por:

$$\frac{P(características|hadrón)}{P(características|no\_hadrón)} \geq \Theta$$

El umbral Θ depende de los costos asociados a cada decisión ($c_{nohadrón}$ y $c_{hadrón}$) y de las probabilidades a priori:

$$\Theta = \frac{c_{\text{nohadrón}} P(\text{hadrón})}{c_{\text{hadrón}} P(\text{no\_hadrón})},$$

En este caso concreto, las distribuciones de probabilidad condicional, i.e. las verosimilitudes, de ambos conjuntos no se tienen, por lo que deberán estimarlas de 2 maneras: calculando histogramas normalizados y calculando un modelo Gaussiano a partir de un conjunto de entrenamiento de cada clase (en Parte 3).

11) Dividir la base de datos en 2 conjuntos representativos: entrenamiento (80%) y prueba (20%). Compruebe la representatividad de estos, verificando si la proporción de ambas clases se mantiene cercana a la proporción del conjunto completo. Se recomienda hacer una permutación al azar de los datos de cada clase antes de definir los conjuntos de entrenamiento y prueba.
12) Utilice Naive Bayes y encuentre los histogramas de cada clase a partir de las muestras del conjunto de entrenamiento. Elija un número de bins que le parezca apropiado a priori. Se recomienda normalizar los histogramas.
13) Encontrar las verosimilitudes en ambas clases para cada muestra del conjunto de prueba, usando los histogramas.
14) Mover Θ, clasificar el conjunto de prueba y calcular la Tasa de Verdaderos Positivos y Tasa de Falsos Positivos cada vez para luego generar la curva ROC (TVP vs TFP). Recuerde que Θ puede tomar valores entre 0 e ∞, pero no todo el rango entrega información importante.
15) Genere la curva precisión-recall

Parte 3: Clasificación usando gaussianas multivariantes

16) Entrenar un modelo gaussiano multidimensional para cada clase, encontrando la media y covarianza, a partir del conjunto de entrenamiento. Puede basarse en las funciones np.mean( ) y np.cov( ) de numpy.
17) Encontrar las verosimilitudes en ambas clases para cada muestra del conjunto de prueba, usando las gaussianas.
18) Mover Θ, clasificar el conjunto de prueba y calcular la Tasa de Verdaderos Positivos y Tasa de Falsos Positivos cada vez para luego generar la curva ROC (TVP vs TFP).
19) Genere la curva precisión-recall

Parte 4: Comparación de resultados

20) Grafique ambas curvas ROC en un único gráfico
21) Grafique ambas curvas precisión-recall en un único gráfico
22) Compare los tiempos requeridos para entrenar ambos clasificadores

El código debe implementado en Python, usando *google colaboratory* (una herramienta que permite ejecutar notebooks de Python en la web: https://colab.research.google.com).

Field Code Changed

En la parte 1 se puede usar las librerías pandas, matplotlib y seaborn

En la parte 2 se puede usar las librerías scikit-learn (sólo para dividir los datos en entrenamiento y prueba), numpy para generar los histogramas y matplotlib para graficar los histogramas y curvas ROC.

En la parte 3 se puede usar sólo numpy para calcular las medias y covarianzas, y matplotlib para graficar las curvas ROC.

En la parte 4 se puede usar matplotlib para graficar las curvas ROC en un mismo gráfico, además de time.time() para medir los tiempos de entrenamiento (se puede agregar al código de las parte 2 y 3)

No se puede usar funciones que generen curvas ROC ni curvas precisión-recall, esto debe ser programado por los alumnos. Además, el alumno debe evitar divisiones por cero en los cálculos.

Los informes y códigos deben ser subidos a u-cursos a más tardar a las 23:59 del día Lunes 5 de abril. En el notebook entregado, incluir un breve comentario indicando cómo ejecutar su programa. El archivo de conjunto de datos a usar no puede ser renombrado ni modificado de ninguna forma. **Las Tareas atrasadas serán penalizadas con un punto base más un punto de descuento adicional por cada día extra de atraso.**

Importante: La evaluación de la tarea considerará el correcto funcionamiento del programa, la inclusión de los resultados de los pasos pedidos en el informe, la calidad de los experimentos realizados y de su análisis, la inclusión de las partes importantes del código en el informe, así como la forma, prolijidad y calidad del mismo.

Nota: Se subirá una guía de ayuda breve que indica cómo usar colaboratory, y cómo poder manipular los datos.

Nota extra: Para detectar copias/plagios, se usará un software llamado turnitin: https://www.turnitin.com/. Los alumnos deben inscribirse en la plataforma para poder subir los informes de las tareas. Los datos del curso son estos:

EL4106 Inteligencia computacional 2021

Class ID: 28719436

Enrollment key: EL41062021

Para evitar problemas, los alumnos deben programar todo lo que se pide por sí mismos, y escribir el informe en su totalidad. No se puede reusar códigos de otros cursos. Las copias/plagios serán castigados.

Field Code Changed