Universidad Tecnológica Centroamericana
Facultad de Ingeniería

CC414 - Sistemas Inteligentes
Docente: Kenny Dávila, PhD

Mini-Proyecto #2: Métodos de Clasificación (5% Puntos Oro)

El objetivo de este mini proyecto es usar y evaluar múltiples algoritmos de clasificación que están disponibles en la librería Scikit-learn. Para esto, se le proveen **datos sintéticos** generados a partir de una distribución oculta sobre un tema de interés nacional.

Contexto

La enfermedad del dengue es una arbovirosis producida por el zancudo aedes aegypti. Cada año muchos compatriotas mueren en la lucha contra el dengue. Por esta razón es necesario concientizarnos y aprender a identificar esta patología para poder darle el manejo adecuado. En particular, ya conocemos el poder de las herramientas de aprendizaje de maquina e inteligencia artificial, y por ellos nos gustaría desarrollar sistemas inteligentes que sean capaces de clasificar correctamente los casos y signos de alarma en esta enfermedad.

Problema

Se le pide usar y evaluar la efectividad de diferentes algoritmos de aprendizaje de máquina para la detección de Dengue y sus niveles de gravedad. Para esta tarea se le proveen 3 versiones del mismo dataset de entrenamiento. La primera versión está basada en datos que normalmente se pueden obtener mediante exámenes de sangre de los pacientes ("laboratorio_train_synth_dengue.csv"). La segunda versión está basada en información que normalmente se puede obtener mediante un análisis clínico del paciente ("clinica_train_synth_dengue.csv"). La tercera versión combina toda la información disponible tanto de laboratorio como de clínica ("completo_train_synth_dengue.csv"). El objetivo es evaluar y comparar el rendimiento de los bosques aleatorios (Random Forests), maquinas de soporte de vectores (SVM) y bayesiano ingenuo (Naive Bayes) sobre cada versión de los datos de entrenamiento. Finalmente se requiere un análisis detallado de los resultados que deberá ser presentado en el reporte.

Atributos

El objetivo es distinguir entre 4 posibles clases: "no dengue", "dengue no grave y sin signos de alarma", "dengue no grave con signos de alarma" y "dengue grave". Para lograr ese objetivo, se consideran los siguientes atributos que se pueden obtener mediante análisis clínico:

1. **Sexo.** Masculino o Femenino. Se sabe que este atributo esta correlacionado con los rangos normales para otros atributos importantes en el diagnóstico.
2. **Días Fiebre.** Total de días que el paciente ha tenido fiebre en las ultimas 2 semanas. Todos los pacientes en el dataset han tenido fiebre al menos por un día.

3. **Días Ultima Fiebre.** Número de días desde que tuvo la ultima fiebre. Será 0 si el paciente tiene fiebre actualmente.
4. **Nauseas**. Indica si el paciente tiene o no nauseas al momento de hacer la evaluación clínica.
5. **Vómitos**. Indica si el paciente ha estado vomitando o no. En caso de tener vómitos, se indicará cuando estos hayan sido persistentes.
6. **Rash**. Indica si el paciente ha desarrollado "rash".
7. **Mialgias.** Indica si el paciente ha estado experimentando dolores musculares.
8. **Artralgias.** Se refiere a si el paciente tiene dolor en las articulaciones.
9. **Prueba Torniquete**. Se refiere al resultado de una prueba de torniquete en caso que esta haya sido aplicada. De lo contrario, se indicará que no se aplico esta prueba (valor NA).
10. **Dolor Abdominal**. Indica si el paciente tiene dolor abdominal constante o al tacto.
11. **Acumulación Fluidos**. Se ha detectado que el paciente está acumulando fluidos.
12. **Sangrado Mucosas.** El paciente tiene sangrado por las mucosas.
13. **Hemorragia.** Se ha detectado que el paciente esta perdiendo plasma.
14. **Shock.** El paciente se encuentra actualmente en estado de shock o está cercano a entrar en dicho estado.
15. **Letargia**. El paciente sufre de alteración de conciencia y no responde de la manera esperada a diferentes estímulos.
16. **Irritabilidad**. El paciente se encuentra en un estado muy sensible a ciertos estímulos.
17. **Hepatomegalia.** El hígado del paciente se encuentra aumentado con respecto a las proporciones normales.

También se proveen los siguientes atributos que normalmente se pueden obtener mediante pruebas de laboratorio:

18. **Plaquetas.** Valores séricos de las plaquetas.
19. **Linfocitos.** Valores séricos de los linfocitos.
20. **Hematocritos.** Valores séricos de los hematocritos.
21. **Leucocitos.** Valores séricos de los leucocitos.

Se asume que todos los datos presentados pertenecen a adultos y son consistentes con rangos observables en adultos con y sin Dengue.

Parte 1. Análisis de Datos

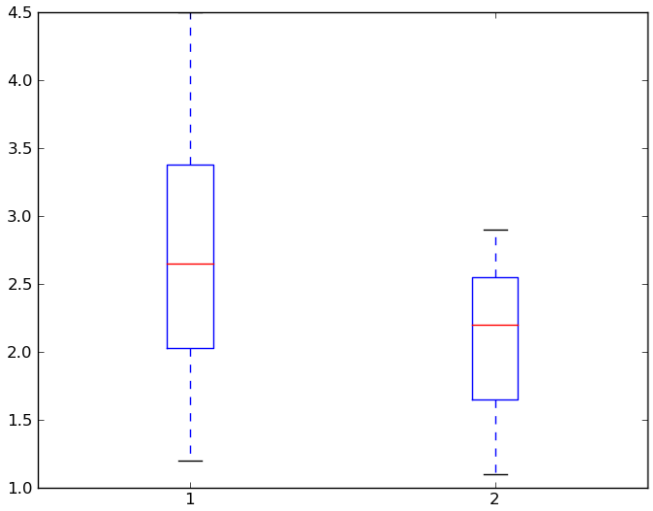
En cualquier dominio nuevo, siempre es necesario conocer los datos con los que trabajamos. Es importante estudiar los rangos de atributos desconocidos. Es importante observar sus distribuciones estadísticas, y de ser posible, es bueno analizar la correlación entre ciertos valores y las clases que queremos identificar.

En este trabajo se le pide analizar cada uno de los 21 atributos que se han provisto. Lo primero que debe hacer es distinguir entre atributos continuos y discretos. Luego, se debe aplicar un tipo de análisis diferente según el tipo del atributo. Para los discretos, podemos hacer una tabla (o matriz) donde

podemos contar cuantas veces se observa cada uno de los valores del atributo por cada cierto tipo de clase:

[No. Atributo]	Clase 1	Clase 2	…	Clase N
Valor 1	[Conteo]	[Conteo]	…	[Conteo]
Valor 2	[Conteo]	[Conteo]	…	[Conteo]
….	…	…	…	…
Valor N	[Conteo]	[Conteo]	…	[Conteo]

A partir de dicha tabla es fácil obtener estadísticas adicionales que nos podrían ayudar a predecir la importancia de ciertos atributos. En el caso de los atributos continuos, es útil poder apreciar la distribución de los valores existentes y contrastarla por cada una de las clases en el dataset. Una manera de hacerlo es mediante el uso de gráficos de caja (Box Plots):



https://stackoverflow.com/questions/4842805/boxplot-with-variable-length-data-in-matplotlib

Para cada atributo, debe identificar el tipo del atributo e incluir el correspondiente grafico de cajas o matriz según en el reporte del mini proyecto. No se darán puntos si solamente se incluye el código para calcularlos, deben agregar de manera ordenada en el reporte. También se le pide analizar los datos generados y predecir que atributos podrían llegar a ser los mas importantes y cuales los menos importantes. Dicha predicción debe estar apoyada directamente por los datos generados y no basado en fuentes externas, aunque también es valido usar dichas fuentes para reforzar o contrastar lo que se ha observado en el dataset.

Parte 2. Random Forests

Como primer experimento, se le pide utilizar bosques aleatorios para clasificar los pacientes en cada uno de los datasets provistos. Los Random Forests cuentan con una serie de parámetros que controlan su efectividad. Su objetivo es tratar de maximizar el promedio del score F-1 por clase, y deberá experimentar con diferentes configuraciones de los parámetros de los árboles de decisión. Debe experimentar con diferentes valores para cada uno de los siguientes parámetros: "criterio", "número de árboles", "profundidad máxima" y "número de atributos". Se le pide probar al menos 15 configuraciones validas distintas de los parámetros del clasificador Random Forests.

Por cada configuración, debe probar el clasificador en cada uno de los 3 datasets. En cada una de estas pruebas deberá aplicar la técnica de validación cruzada (cross-validation) dividiendo los datos de entrenamiento disponibles en 5 partes. Se entrenarán 5 clasificadores distintos alternando cada una de las 5 partes como datos de validación y usando las demás como datos de entrenamiento.

En total si prueban 15 configuraciones, en 3 datasets, usando la validación cruzada con 5 partes, entonces usted terminará entrenando un total de 15x3x5=225 clasificadores distintos. Este es un trabajo a completarse en parejas y se recomienda dividir las pruebas para que les ajuste el tiempo.

Como resultado de este ejercicio, se le pide reportar una tabla como sigue:

						F1-Promedio por Clase					
Conf. Id	**Dataset**	**Param 1**	**Param 2**	**...**	**Param N**	**P1**	**P2**	**P3**	**P4**	**P5**	**Promedio**
1	Lab	val P1 1	val P2 1	...	val PN 1						
2	Lab	val P1 2	val P2 2	...	val PN 2						
...		...	...	...	...	...	...	...	...	...	...
N	Lab	val P1 N	val P2 N	...	val PN N						
1	Clinica	val P1 1	val P2 1	...	val PN 1						
2	Clinica	val P1 2	val P2 2	...	val PN 2						
...		...	...	...	...	...	...	...	...	...	...
N	Clinica	val P1 N	val P2 N	...	val PN N						
1	Completo	val P1 1	val P2 1	...	val PN 1						
2	Completo	val P1 2	val P2 2	...	val PN 2						
...		...	...	...	...	...	...	...	...	...	...
N	Completo	val P1 N	val P2 N	...	val PN N						

Debe seleccionar la configuración que obtuvo el **mejor promedio** del **F-1 promedio por clase** sobre los datos de validación y reportar estadísticas mas detalladas sobre esta configuración. Esto incluye la matriz de confusión para cada uno de los datasets sobre los datos de validación, y el accuracy total en los datos de validación. Note que la validación se ejecuta en 5 partes, por lo que para obtener la matriz de confusión de los datos de validación usted deberá sumar las matrices individuales de cada una de las 5 partes. Luego, es valido calcular el accuracy total en base a esta matriz de confusión combinada.

Posteriormente se liberarán los datasets de prueba. Consistentes con los datasets de entrenamiento, se le proveerán 3 datasets de prueba: uno basado en datos clínicos, uno basado en datos de laboratorio y

uno basado en el conjunto completo de atributos. Debe entrenar tres clasificadores Random Forest, uno por cada dataset de entrenamiento, usando el 100% de los datos disponibles por dataset y mejor la configuración de parámetros encontrada en la prueba anterior. Luego, se le pide evaluar cada clasificador con el dataset de pruebas correspondiente. **Por cada dataset de pruebas**, debe reportar la matriz de confusión, el promedio de F-1 por clase, y la efectividad (accuracy) total.

Parte 3. Support Vector Machines

Como segundo experimento, se le pide utilizar máquinas de soporte de vectores para clasificar los pacientes en cada uno de los datasets provistos. Las maquinas de soporte de vectores cuentan con una serie de parámetros que controlan su efectividad. Su objetivo es tratar de maximizar el promedio del score F-1 por clase, y deberá experimentar con diferentes configuraciones de los parámetros de las SVM. Debe experimentar con diferentes valores para cada uno de los siguientes parámetros: "kernel", "C", y "Gamma". Se recomienda concentrarse en experimentos con kernels gausianos (Kernel="rbf"), para los cuales solo aplican los parámetros "C" y "Gamma". Se le pide probar al menos 15 configuraciones validas distintas de los parámetros del clasificador basado en SMV.

SVC: https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

Es importante tener presente que las SVM son sensibles a los rangos de los atributos y que también es requerido que estos sean numéricos. Por lo tanto, debe convertir los datos categóricos a números y debe usar codificación en aquellos atributos discretos no binarios (un atributo binario por cada valor posible del atributo). Posteriormente debe usar Normalización los datos para queden en rangos similares.

Debe repetir el mismo procedimiento experimental descrito en la parte 2. Básicamente, se le pide probar al menos 15 configuraciones sobre cada uno de los 3 datasets usando cross-validation en 5 partes. Debe reportar una tabla resumen como se pide en la Parte 2 por lo que también se recomienda una división del trabajo.

La mejor configuración debe ser usada para reportar estadísticas más detalladas sobre los datos de validación. Finalmente, también debe entrenar 3 clasificadores, uno por dataset de entrenamiento, usando el 100% de los datos y la mejor configuración y luego debe evaluar cada uno de ellos usando los datos de pruebas cuando estos se vuelvan disponibles.

Parte 4. Naive Bayes

Como tercer experimento, se le pide utilizar Bayesianos Ingenuos (Naive Bayes) para clasificar los pacientes en cada uno de los datasets provistos. A diferencia de los clasificadores anteriores, hay muy pocos parámetros que podamos ajustar en un clasificador Naive Bayes. Lo más importante es escoger el tipo de distribución que se utilizara para modelar las probabilidades de los valores para cada uno de los atributos. Los datasets provistos contienen combinaciones de datos numéricos y categóricos y es probable que cada tipo de distribución funcione mejor sobre algún conjunto de atributos específicos.

Para cada dataset, deberá realizar pruebas con los siguientes tipos de Naive Bayes: Gausiano (GaussianNB), Bernoulli (BernoulliNB), y Categórico (CategoricalNB). Para cada una de estas variaciones, se le pide usar cross-validation (5 partes) para estimar el promedio del score F-1 por clase. Usando los datos de validación, se le pide calcular la matriz de confusión combinada, el promedio del score F-1 por

clase y el accuracy. En total son 3 tipos de clasificadores Naive Bayes y 3 datasets, por lo que deberá reportar un total de 9 matrices de confusión con sus respectivas estadísticas.

El tipo de distribución Gaussiana es el más genérico y se puede usar con cualquier tipo de datos siempre y cuando estos se conviertan a números. Igual que con el SVM, será necesario que transforme los datos de entrenamiento (y prueba) antes de pasárselos al clasificador Naive Bayes. Sin embargo, no se esperan buenos resultados sobre los atributos categóricos que tienen una distribución distinta a la normal.

El tipo de distribución Bernoulli funciona bien sobre atributos binarios. Muchos de los atributos en los datasets son binarios y funcionaran bien con este tipo de clasificador. Sin embargo, todos los atributos deben ser convertidos a números primero igual que en el caso anterior. Adicionalmente, atributos continuos deben ser binarizados aplicando un umbral. Se recomienda usar los resultados de la parte 1 para escoger valores de umbral adecuados para cada atributo continuo y de esa manera convertirlos a valores binarios (valor attributo > valor umbral).

El tipo de distribución categórica funciona bien sobre atributos categóricos, sean binarios o no. Siempre se debe convertir los valores no numéricos a números, pero no es necesario que atributos categóricos con de dos atributos sean codificados en múltiples atributos binarios, sino que se puede usar números continuos ya que el clasificador entiende que se trata de categorías independientes. Puede usar el Codificador Ordinal para esta tarea. Siempre será necesario convertir los atributos continuos a categorías ya sea usando binarización o rangos predefinidos.

https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OrdinalEncoder.html

Finalmente, por cada tipo de distribución (Normal, Bernoulli y Categórica) también debe entrenar 3 clasificadores, uno por dataset de entrenamiento (9 clasificadores en total), usando el 100% de los datos y luego debe evaluar cada uno de ellos usando los datos de pruebas correspondientes cuando estos se vuelvan disponibles.

Requisitos

1. (mp_2_parte_1.py) Implementación de un script basado en Python que generé las estadísticas de la parte 1.

2. (mp_2_parte_2_cross.py) Implementación de un script basado en Python que recibe como entrada el nombre del archivo del dataset de entrenamiento, y genera los resultados de cross-validacion para distintas configuraciones de **Random Forests**.

3. (mp_2_parte_2_train.py) Implementación de un script basado en Python que recibe como entrada el nombre del archivo del dataset de entrenamiento y el nombre del archivo de salida, y entrene un clasificador **Random Forest** usando 100% de los datos de entrenamiento y lo salve en archivo indicado.

4. (mp_2_parte_2_test.py) Implementación de un script basado en Python que recibe como entrada el nombre del archivo del dataset de prueba y el nombre del archivo donde se guardó el clasificador **Random Forest** y genere las estadísticas de evaluación requeridas.

5. (mp_2_parte_3_cross.py) Implementación de un script basado en Python que recibe como entrada el nombre del archivo del dataset de entrenamiento, y genera los resultados de cross-validacion para distintas configuraciones de **SVM**.

6. (mp_2_parte_3_train.py) Implementación de un script basado en Python que recibe como entrada el nombre del archivo del dataset de entrenamiento y el nombre del archivo de salida, y entrene un clasificador **SVM** usando 100% de los datos de entrenamiento y lo salve en archivo indicado.

7. (mp_2_parte_3_test.py) Implementación de un script basado en Python que recibe como entrada el nombre del archivo del dataset de prueba y el nombre del archivo donde se guardó el clasificador **SVM** y genere las estadísticas de evaluación requeridas.

8. (mp_2_parte_4_cross.py) Implementación de un script basado en Python que recibe como entrada el nombre del archivo del dataset de entrenamiento, y genera los resultados de cross-validacion para distintas configuraciones de **Naive Bayes**.

9. (mp_2_parte_4_train.py) Implementación de un script basado en Python que recibe como entrada el nombre del archivo del dataset de entrenamiento, el tipo de distribución del Naive Bayes y el nombre del archivo de salida, y entrene un clasificador **Naive Bayes** basado en el tipo de distribución indicado usando 100% de los datos de entrenamiento y lo salve en archivo indicado.

10. (mp_2_parte_4_test.py) Implementación de un script basado en Python que recibe como entrada el nombre del archivo del dataset de prueba, el tipo de distribución del Naive Bayes y el nombre del archivo donde se guardó el clasificador **Naive Bayes** y genere las estadísticas de evaluación requeridas.

11. Un reporte que detalle lo siguiente:
    a. Introducción incluyendo una breve motivación del problema en sus propias palabras
    b. Descripción de su implementación incluyendo comandos específicos para ejecutar su código fuente y librerías específicas de Python que son requeridas por sus programas.
    c. Resultados y Estadísticas conforme a todo lo solicitado en la Parte 1.
    d. Resultados y Estadísticas conforme a todo lo solicitado en la Parte 2.
    e. Resultados y Estadísticas conforme a todo lo solicitado en la Parte 3.
    f. Resultados y Estadísticas conforme a todo lo solicitado en la Parte 4.
    g. Análisis Resumido.
        i. ¿Qué clasificador le dio los mejores resultados en validación?
        ii. ¿Qué clasificador le dio los mejores resultados en los datos de prueba?
        iii. ¿Qué clasificador prefiere y por qué razones?
    h. Dificultades encontradas. Describa en sus propias palabras cuales fueron los retos más difíciles durante la realización de este mini proyecto.
    i. Conclusiones.

Evaluación

1. Parte 1: 10%
2. Parte 2: 30%
3. Parte 3: 30%
4. Parte 4: 20%
5. Reporte: 10%

Fechas Importantes

1. Mini-Proyecto asignado: 28 de noviembre del 2020
2. Datos de entrenamiento disponibles: 28 de noviembre del 2020
3. Datos de prueba disponibles: 9 de diciembre del 2020
4. Entrega del mini-proyecto: 12 de diciembre del 2020 (antes de 23:59:59 GMT-5)

Otras políticas

1. Este mini-proyecto deberá trabajarse y entregarse **individualmente o en parejas.**
2. Las expectativas del proyecto serán idénticas ya sea que se entregue individualmente o con pareja. Por ende se recomienda trabajar en parejas.
3. El **plagio** será penalizado de manera severa.
4. Los estudiantes que entreguen proyectos 100% originales recibirán una nota parcial a pesar de errores existentes en la funcionalidad y otras imperfecciones en el reporte. En cambio, los estudiantes que presenten proyectos que cumplan a la perfección todos los requisitos pero que contenga material plagiado (código o texto) recibirán 0% automáticamente.
5. Es mandatorio el uso de Python (versión 3.7+) para este proyecto. La librería Scikit-learn esta disponible para Python solamente.
6. Python es un lenguaje de scripting pero también tiene soporte para programación orientada a objetos y como mínimo el uso de funciones. Se tomará en cuenta el estilo de programación y **se substraerán puntos por código desordenado**. Todo el código ejecutable debería estar encapsulado en funciones y es recomendable que exista una función principal que llame todas las demás rutinas (una función main()).
7. **Se sustraerán puntos por usar nombres de archivos "hard-coded"**. Los nombres de los archivos deben especificarse desde la consola de comandos y deben ser leídos por el script usando los valores de sys.argv.
8. Esta permitido el uso de librerías externas para la implementación de funciones de este proyecto. Sin embargo, trate de utilizar solamente las librerías que sean altamente necesarios para intentar **mantener el numero de dependencias tan bajo como sea posible**.
9. Proyectos entregados después de la fecha de entrega solamente podrán recibir la mitad de la calificación final. Por esta razón, es posible que **un trabajo incompleto pero entregado a tiempo termine recibiendo mejor calificación que uno completo entregado un minuto tarde**.