Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

Tarea 1

IIC2026 – Visualización de información
Entrega: 30 de agosto de 2019, 19:59

Durante las vacaciones, los ayudantes discutieron acerca de 3 diferentes modelos de *machine learning* capaces de clasificar la información de un cierto *dataset*. Querían saber cuál sería mejor para ocupar en esta ocasión. Dado que no lograron ponerse de acuerdo y ninguno tuvo el tiempo para demostrar con resultados convincentes cuál era mejor, se le ha solicitado a usted, estudiante de visualización, generar un reporte en el que se compare mediante gráficos cómo cada modelo clasifica los datos y qué tan bueno es.

Esta tarea pretende ser un primer acercamiento al manejo y limpieza de datos, uso de algoritmos de *machine learning* con la librería `sklearn` y a la creación de visualizaciones de información básicas. Esta tarea es **individual** y se entrega en su repositorio de Github creado a partir de la siguiente **invitación**.

1. Preprocesamiento de datos [1.5 puntos]

1.1. Objetivo

La meta de esta parte es llevar a cabo un proceso de limpieza, integración y transformación de datos mediante un *script* escrito en **Python**[1] utilizando la librería de procesamiento **pandas** y enfrentarse a los problemas que surgen al lidiar con un *dataset* real. Se trabajará con información de diversas plantas, pero que, lamentablemente, viene fragmentada en múltiples archivos con distinta estructura y formato. Su trabajo consistirá en integrar dichos *datasets* con el objetivo de tener de forma limpia y concisa la información que deberá visualizar en la siguiente parte de la tarea.

1.2. Datos

El *dataset* de interés corresponde a la información de ciertas plantas y a qué clase de plantas corresponden (qué tipo de plantas son). En particular, para cada planta se cuenta con 4 atributos numéricos y 1 dato categórico: `"sepal length (cm)"`, `"sepal width (cm)"`, `"petal length (cm)"`, `"petal width (cm)"`, `"class"`.

Este *dataset* está segmentado en 2 archivos con formato distinto: *training.csv* y *predict.json* los cuales deberá leer, filtrar y, luego, tendrá que generar nuevos archivos en

[1] Los textos escritos en magenta contienen enlaces de utilidad.

formato **CSV** que serán procesados por una librería entregada a ustedes por los ayudantes para entrenar modelos de machine learning y clasificar con ellos las plantas. Para lograr lo anterior, usted debe escribir un programa en **Python** y que utilice la librería **pandas** para procesar los archivos con los datos de plantas. Se espera que al final de esta etapa, transforme cada archivo a uno **CSV** con el siguiente formato:

1. La primera línea debe ser: `"sepal_length,sepal_width,petal_length,petal_width,class"`
2. Cada línea siguiente tiene que contener los datos de una planta en el mismo orden de la primera línea separados sólo por comas (sin espacios).

Un ejemplo de archivo resultante sería:

```
sepal_length,sepal_width,petal_length,petal_width,class
5.1,3.5,1.4,0.2,1
6.2,3.4,5.4,3.2,0
```

Es importante que no combinen la información de los archivos. Cada uno por sí solo debe ser convertido al formato pedido.

A continuación, se explicitan los detalles de cada archivo en consideración.

1.3. *training.csv*

Este archivo viene en formato **CSV**. En él, cada fila contiene el número identificador de una planta, el nombre de algún atributo y el valor que tal atributo toma para aquella planta. Es decir, el archivo cuenta con las siguientes columnas:

`ID_planta, atributo, valor`

- `ID_planta`: corresponde a un identificador único de cada planta muestreada.
- `atributo`: corresponde al nombre de un atributo de la planta, este puede ser `sepal_length`, `sepal_width`, `petal_length`, `petal_width` o `class`.
- `valor`: corresponde a un valor numérico asociado al atributo descrito en la columna anterior.

1.4. *predict.json*

Este archivo viene en formato **JSON**, donde cada entrada representa una planta. Cada objeto tiene la siguiente estructura:

```
ID_PLANTA: {
    "sepal_length": <número>,
    "sepal_width": <número>,
    "petal_length": <número>,
    "petal_width": <número>,
    "class": <número>,
}
```

`ID_PLANTA` es el identificador único de cada planta muestreada. Las llaves `sepal_length`, `sepal_width`, `petal_length`, `petal_width` son los atributos de la planta y `class` es el tipo real de la planta. Con este último valor será posible determinar si el modelo de *machine learning* a ocupar logra clasificar como corresponde la planta o no.

1.5. Problemas con datos

Actualmente, ambos archivos cuentan con el problema de poseer datos sin sentido. En particular, hay atributos negativos, plantas clasificadas con una clase distinta a `0`, `1` o `2` o plantas a las que les faltan uno o más atributos. Por lo tanto, será necesario un preprocesamiento en el que se eliminen todas las plantas que presenten estos problemas.

2. Visualizar clasificaciones [2.0 puntos]

2.1. Objetivo

Esta sección será el primer acercamiento a generar datos que luego podrán ser visualizados. A partir de un programa escrito en **Python**, usted deberá entrenar los 3 modelos de *machine learning* con los datos preprocesados de *training.csv* y luego usar dichos modelos para clasificar las plantas contenidas en el archivo *predict.json*. Para el proceso de entrenar y predecir, deberá utilizar los modelos `DecisionTreeClassifier`, `KNeighborsClassifier` y `GaussianNB`

Luego de tener la clasificación de cada planta según los 3 modelos. Deberá generar un gráfico de puntos 2D por modelo en el que el color de cada punto corresponda a la clase indicada por el modelo al momento de clasificar. Además deberá generar un gráfico de puntos con las verdaderas clases indicadas en el archivo *predict.json*, de modo que se pueda comparar cada respuesta de los modelos con los valores reales de las clases. Un ejemplo de los gráficos solicitados es el siguiente:

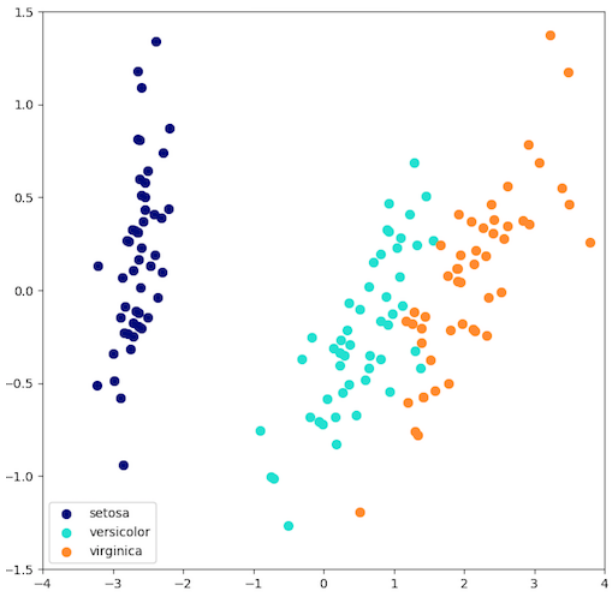

Figura 1: Ejemplo gráfico de puntos donde cada color es una clase

No olvide que debe realizar uno de estos gráficos para cada modelo y otro con las clases reales.

Recuerde que el *dataset* indica que cada planta cuenta con 4 atributos, es decir 4 columnas con datos por fila, pero para visualizar solo se ocupan 2 (coordenada x, y). Para esto deberá utilizar una técnica de reducción de dimensionalidad llamada `PCA`[2] que es capaz de reducir las columnas de atributos a 2 columnas, comprimiendo la información de todos los atributos. Esto permitirá graficar los puntos en 2D.

Puede utilizar el programa o librería que estime conveniente para generar el gráfico. Algunas opciones son **matplotlib**, **altair**, **plotly**, **RawGraphs**, **Tableau**, **Seaborn** etc. No tiene que replicar exactamente el ejemplo de la Figura 1, pero sí debe producir el mismo tipo de gráfico que muestre la clase por cada punto. Además, los gráficos deben ser reproducibles por los ayudantes, esto implica que debe agregar el código y/o las instrucciones necesarias para utilizar los datos preprocesados y generar la misma imagen que mostrarán en el reporte final.

3. Visualizar métricas [1.0 punto]

3.1. Objetivo

Esta sección busca que experimente una de las tareas típicas a realizar cuando se trabaja con modelos de *machine learning*, que es comparar mediante diversas métricas cuál modelo clasifica mejor el *dataset*. Para esta ocasión se va a comparar el *accuracy*, *precision* y *recall* de cada modelo.

Para calcular las métricas, deberá obtener la clasificación según cada modelo y dependiendo de la métrica, deberá considerar cierta información:

- *accuracy*: para esta métrica, se debe contar la cantidad de veces que la predicción realizada por el modelo haya sido igual a la clase indicada en la columna `class` (clasificación correcta) y luego deberá dividir ese resultado en la cantidad total de predicciones realizadas. También puede ocupar la función `accuracy_score` que tiene `sklearn`.
- *precision*: esta métrica se calcula **por clase** y corresponde a la cantidad de veces que la predicción realizada de dicha clase fue correcta, dividida por la cantidad de veces que el modelo predijo dicha clase, sea correcta o incorrecta.
- *recall*: esta métrica se calcula **por clase** y corresponde a la cantidad de veces que la predicción realizada de dicha clase fue correcta, dividida por la cantidad de veces que estaba presente dicha clase en los datos originales.

Como ejemplo: suponga que se tienen las siguientes 3 clases: alumno, profesor y ayudante; y que se dispone de una lista de personas con la siguientes clases y la predicción realizada por un modelo de *machine learning*:

[2]No es necesario entender cómo funciona PCA, sino solo saber cómo ocupar la librería sklearn que permite utilizar PCA.

Clase verdadera	Clase predicha
profesor	profesor
profesor	alumno
ayudante	alumno
ayudante	alumno
ayudante	ayudante
ayudante	ayudante
alumno	ayudante
alumno	alumno
alumno	profesor
alumno	alumno
alumno	profesor
alumno	alumno

En forma de matriz de confusión, esta se vería del siguiente modo:

	Predicción Profesor	Predicción Ayudante	Predicción Alumno	Total clases
Clase Profesor	1	0	1	2
Clase Ayudante	0	2	2	4
Clase Alumno	2	1	3	6
Total Predicción	3	3	6	12

- *accuracy*: el modelo predijo lo mismo que la clase verdadera en 6 ocasiones de 12. Por lo tanto, el *accuracy* es 6/12.
- *precision*:
  - Profesor: el modelo predijo lo mismo que la clase verdadera 1 de las 3 veces que indicó que los datos correspondían a la clase "Profesor". Por lo tanto, la métrica *precision* de esta clase es 1/3.
  - Ayudante: el modelo predijo lo mismo que la clase verdadera 2 de las 3 veces que indicó que los datos correspondían a la clase "Ayudante". Por lo tanto, la métrica *precision* de esta clase es 2/3.
  - Alumno: el modelo predijo lo mismo que la clase verdadera 3 de las 6 veces que indicó que los datos correspondían a la clase "Alumno". Por lo tanto, la métrica *precision* de esta clase es 3/6.
- *recall*:
  - Profesor: el modelo predijo lo mismo que la clase verdadera 1 de las 2 veces que estaba presente esta clase. Por lo tanto, la métrica *recall* de esta clase es 1/2.
  - Ayudante: el modelo predijo lo mismo que la clase verdadera 2 de las 4 veces que estaba presente esta clase. Por lo tanto, la métrica *recall* de esta clase es 2/4.
  - Alumno: el modelo predijo lo mismo que la clase verdadera 3 de las 6 veces que estaba presente esta clase. Por lo tanto, la métrica *recall* de esta clase es 3/6.

Luego de calcular cada métrica, deberá generar un gráfico de barras para poder comparar cada una de ellas según los diferentes modelos. Es posible apreciar que para las métricas *precision* y *recall* se tienen 3 datos por cada modelo (uno por clase). **Queda a decisión de usted** cuál de las siguientes 2 opciones hacer:

- Un **gráfico de barra agrupado** donde cada grupo corresponda a un modelo y cada barra del grupo a la métrica (*precision* o *recall*) de cada clase. Esta opción implica generar 3 gráficos distintos. No olvide incluir las leyendas y nombres necesarios para identificar cada clase, modelo y tipo de métrica. En caso de que los gráficos solicitados no contengan las leyendas y nombres mencionados, se aplicará un descuento en el puntaje de este ítem de la tarea.
- Un gráfico de barra por métrica de la clase, es decir, un gráfico donde, para los 3 modelos, se grafique la métrica *precision* o *recall* de una clase en específico. Esta opción implica generar 9 gráficos distintos. Al igual que en la primera opción, se aplicará un descuento en el puntaje de este ítem si los gráficos no cuentan con las leyendas y nombres necesarios.

Importante: para cada modelo deberá imprimir en consola o escribir en un archivo el nombre del modelo utilizado y las 3 métricas calculadas para dicho modelo.

4. Presentación de resultados [1.5 puntos]

Debe crear un informe en el que presente los resultados obtenidos en esta tarea. Este debe ser un documento **HTML** que contenga la siguiente información:

1. Explicación paso a paso de cómo se genera cada gráfico. Esto es, desde que se abre el archivo de datos hasta que se obtiene el resultado final. Considere que debe haber una explicación distinta por tipo de gráfico, puesto que uno es un gráfico de puntos y el otro es de barra. Basta con explicaciones de alto nivel, no es necesario explicar cada línea de código.
2. Incluir los gráficos de puntos de la sección 2 para cada modelo y el gráfico con la respuesta correcta (4 gráficos en total).
3. Incluir los gráfico de la sección 3 con la comparación del *accuracy*, *precisión* y *recall*.
4. Conclusión obtenida a partir de los resultados.

Recuerde hacer uso de los diferentes *tags* que posee **HTML** para mostrar la información de forma ordenada y amigable para el lector. Se espera que al menos el documento contenga lo siguiente:

- Contenido centrado (textos e imágenes) y texto justificado.
- Cambios estéticos en un archivo **CSS**, que enriquezcan la presentación de su informe.

No olvide que este documento será su informe de la tarea, por lo que adicionalmente se evaluará que cumpla con una correcta redacción y ortografía.

5. *Bonus* [1 punto]

Se otorgará un bono de 0.5 puntos de nota a aquellos informes con dinamismo en los gráficos de clasificación (solo los gráficos de puntos). Este debe ser tal que el informe solo muestre 1 gráfico a la vez y ser capaz de cambiar a elección el gráfico a mostrar. Puede utilizar *tabs*, *dropdowns* o cualquier otro mecanismo implementado en **JavaScript** para lograr dicho objetivo.

Se otorgará otro bono de 0.5 puntos de nota a aquellos que hagan una página *responsive*. Esta debe adaptarse al tamaño de diferentes pantallas, en particular, a la de un celular. Puede utilizar los comandos que usted estime conveniente mientras sea mediante **JavaScript** o **CSS**. Un ejemplo de página *responsive* es: `https://pudding.cool/2018/08/pockets/`

6. Entregables

Usted debe entregar su tarea en el repositorio que se le crea automáticamente al ingresar al **enlace** señalado anteriormente. Los entregables que se esperan de esta tarea son:

- **Preprocesamiento de datos**: Un *script* escrito en **Python 3.4** (o superior) que procese los archivos entregados y genere nuevos archivos con el formato pedido para entrenar los diferentes modelos y clasificar.
- **Visualizar clasificaciones**: Un *script* escrito en **Python 3.4** (o superior) que procese los datos y clasifique las plantas del archivo *predict.json*. Este debe ser un archivo distinto al de preprocesamiento. Puede ser un archivo *.py* o *.ipynb*
- **Visualizar métricas**: Un *script* escrito en **Python 3.4** (o superior) que procese los datos y genere las métricas de comparación entre los modelos. Puede ser el mismo archivo que el usado para clasificar las plantas. Puede ser un archivo *.py* o *.ipynb*
- **Presentación resultados**: Un informe en formato **HTML** que detalle lo realizado y logrado en las anteriores secciones y un archivo **CSS** que enriquezca su presentación.
- ***Bonus* 1**: Un *script* escrito en **JavaScript** que agregue dinamismo al **HTML**.
- ***Bonus* 2**: Un *script* escrito en **JavaScript** o código escrito en **JavaScript** que haga *responsive* la página.

7. Política de atraso

En la eventualidad de entregar pasada la fecha de entrega, se aplicará un **descuento** a la nota final obtenida en su tarea.

De haber atraso, el descuento comienza desde las 5 décimas. El descuento aumenta linealmente hasta las 24 horas posteriores del plazo inicial hasta un total de 20 décimas. Pasado un día del plazo inicial, el descuento es de 70 décimas, es decir, nota final **1**. La figura 2 muestra la función de descuento en función a las horas de atraso.

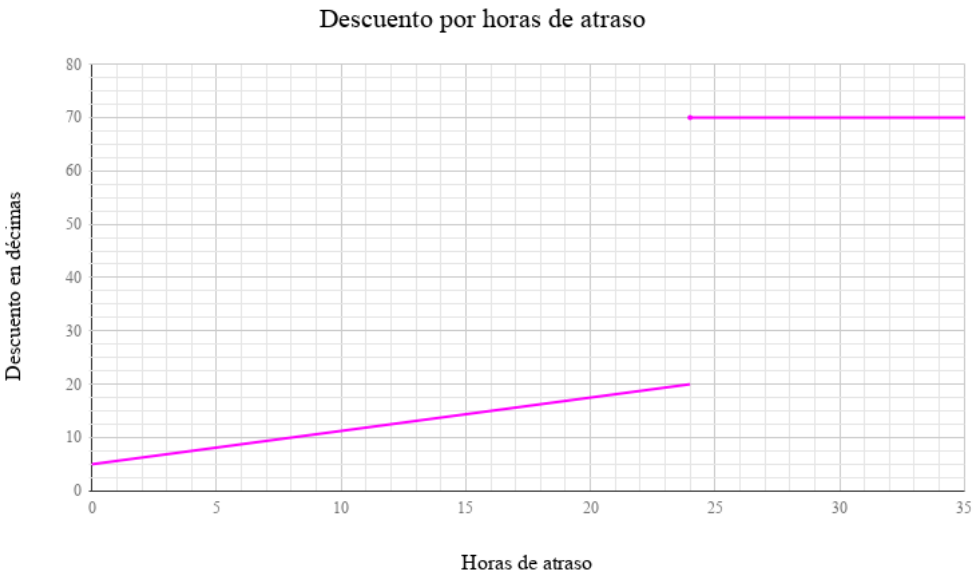

Figura 2: Descuento en nota según horas de atraso.