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

Tarea 3: Minería de Datos IIC2433

Profesor: Vicente Domínguez

Fecha de entrega : 04 de Diciembre de 2020, 23:59 hrs.

Indicaciones

- Se debe entregar la tarea en el repositorio asignado a cada uno por *Github Classroom*. Deben ir al siguiente link: `https://classroom.github.com/a/y8eBLfcH`
- Cada hora de atraso descuenta 1 punto de la nota que obtengas.
- La tarea es *individual*. La copia será sancionada con una nota 1.1 en la tarea, además de las sanciones disciplinarias correspondientes.
- **No se permite** el uso de librerías diferentes a las mencionadas en este enunciado.
- **El código entregado se comparará con otras implementaciones** que estén disponibles en la *web*. La copia directa será sancionada con un 1.1.

Introducción

Clustering es una de las técnicas de análisis de datos exploratorios más comunes utilizadas para obtener una intuición sobre la estructura de los datos. *Clustering* es un método de aprendizaje no supervisado, que consiste en ir identificando subgrupos (*clusters*) de puntos "similares" dentro de un conjunto de datos.
Cuando nos enfrentamos a problemas de segmentación *Clustering* resulta ser una técnica muy usada, tenemos la ventaja que al ser una técnica de aprendizaje no supervisado no tenemos la necesidad de tener datos etiquetados.

En esta tarea, ustedes deberán implementar el algoritmo de *clustering jerárquico* [Patel et al., 2015] y aplicarlo a un *dataset* realizando el correspondiente análisis. Este algoritmo tiene dos versiones: **clustering aglomerativo**, donde se parte de n clusters según la cantidad de datos y se van incorporando datos creando

un nuevo *cluster* más grande en cada paso (de aquí tenemos la idea de aglomerativo), o **clustering divisivo**, donde se realiza el trabajo opuesto a la técnica anterior, partiendo de un gran *cluster* y dividiéndolo en otros más pequeños.

En esta tarea deberán implementar desde cero el algoritmo de *clustering jerárquico* en su versión de *clustering aglomerativo*, incorporando los conceptos de **matriz de distancias**, **dendrograma**, y usar la métrica de evaluación *silhouette score*, que penaliza las clases por estar "muy juntas".

Base de Datos

Link al Dataset: Diabetes 130-US hospitals for years 1999-2008 Data Set

Para esta tarea utilizaremos un *dataset* de pacientes de 130 hospitales de USA entre los años 1999 y 2008, *diabetic_data.csv*. Los pacientes fueron seleccionados acorde a los siguientes parámetros:

1. Fue ingresado al hospital.
2. Tiene algún tipo de diabetes dentro del diagnóstico ingresado.
3. El paciente tuvo una estadía en el hospital desde 1 día hasta 14 días.
4. Al paciente se le realizaron exámenes de laboratorio durante la hospitalización.
5. Al paciente le fue administrado medicamentos durante la estadía en el hospital.

La cantidad de pacientes que cumplieron los criterios fue de 101,766.
Este *dataset* fue creado con el objetivo de verificar la incidencia en la readmisión de pacientes diabéticos hospitalizados previamente, analizando el uso de ciertos tratamientos.
En la siguiente tabla 1 se entrega información sobre los atributos. [1]

[1] https://www.hindawi.com/journals/bmri/2014/781670/

Nombre del Atributo	Tipo	Descripción y valores
Encounter ID	Numérico	Identificador único de registro
Nº de paciente	Numérico	Identificador único de paciente
Raza	Nominal	Valores: Caucasico, Asiatico, Afro Americano, Hispanico, y otros
Género	Nominal	Valores: male, female, y unknown/invalid.
Edad	Nominal	Agrupado en intervalos de 10 años
Peso	Numérico	Peso en libras
Tipo de admisión	Nominal	Identificador entero de 9 valores diferentes
Tipo de alta médica	Nominal	Identificador entero de 29 valores diferentes
Fuente de admisión	Nominal	Identificador entero de 21 valores diferentes
Estadía en el hospital	Numérico	Identificador entero de días entre la admisión y el alta médica
Código de pago	Nominal	Identificador entero de 23 valores diferentes
Especialidad médica	Nominal	Identificador entero de la especialidad médica de admisión de 84 valores diferentes
Nº de procedimientos de laboratorio	Numérico	Número de test de laboratorio realizados durante la estadía
Nº de procedimientos	Numérico	Número de procedimientos (distintos a test de lab.) realizados durante la estadía
Nº of medicamentos	Numérico	Número de diferentes nombres genéricos administrado durante la estadía.
Nº de visitas del paciente	Numérico	Número de visitas posteriores al alta durante el año siguiente
Nº de visitas de emergencia	Numérico	Número de visitas de emergencia posteriores al alta durante el año siguiente
Nº de visitas paciente ingresado	Numérico	Número de visitas como paciente ingresado posteriores al alta durante el año siguiente
Diagnostico 1	Nominal	Diagnóstico primario (código de 3 dígitos iniciales ICD9); 848 valores distintos.
Diagnostico 2	Nominal	Diagnóstico secundario (código de 3 dígitos iniciales ICD9); 923 valores distintos.
Diagnostico 3	Nominal	Diagnóstico secundario adicional (código de 3 dígitos iniciales ICD9); 954 valores distintos.
Nº de diagnósticos	Numérico	Número de diagnósticos entrados en el sistema
Glucose serum test result	Nominal	Indica rango del resultado del test o si no fue tomado △
A1c test result	Nominal	Indica rango del resultado del test o si no fue tomado ⋆
Cambio de medicamentos	Nominal	Indica si hubo cambio en los medicamentos para la diabetes (dósis o nombre genérico) •
Medicamentos diabetes	Nominal	Indica si hubo medicamento rectado para diabetes, Valores: "si" y "no"
24 atributos para medicamentos	Nominal	Indica si el medicamento fue recetado ∘, ∗
Readmitido	Nominal	Días de la re-admisión del paciente ♢

Table 1:
△ Valores: "> 200," "> 300," "normal," y "none" no fue medido
⋆ Valores: "> 8" resultado mayor a 8%, "> 7" resultado mayor a 7% pero menor a 8%, "normal" resultado menor a 7%, y "none" no fue medido.
• Valores: "cambio" y "no cambio".
∘ Nombres medicamentos genéricos: metformin, repaglinide, nateglinide, chlorpropamide, glimepiride, acetohexamide, glipizide, glyburide, tolbutamide, pioglitazone, rosiglitazone, acarbose, miglitol, troglitazone, tolazamide, examide, sitagliptin, insulin, glyburide-metformin, glipizide-metformin, glimepiride-pioglitazone, metformin-rosiglitazone, and metformin-pioglitazone.
∗ Valores: "up" dosis aumento durante estadía, "down" dosis disminuyó durante estadía, "steady" la dosis no cambio, y "no" si no fue prescrito el medicamento
♢ Valores: "< 30" paciente fue readmitido en menos de 30 días, "> 30" paciente fue readmitido en más de 30 días, y "No" si no hay reporte de readmisión

Para cargar el *dataset*, deberás crear una función llamada `preprocesamiento` que se encargue de abrir el *dataset* con pandas y limpiar los datos según la descripción de la tabla 1, por ejemplo, tratar los datos defectuosos, los `NaN`, etc. Por otro lado, en esta función puedes excluir una o más columnas para no utilizarlas en el estudio posterior (deberás explicar estas decisiones en la parte 5 de la tarea). Finalmente, la función debe retornar un `DataFrame` con los datos limpios y listos para entregarlos al algoritmo de *clustering*.

NOTA: Para este paso, y en la definición de la función, se podrá hacer uso de la librería `sklearn`. Esto solo será permitido cuando deban realizar reducción de dimensionalidad (requerimiento explicado más abajo), es decir, **el proceso de normalización, limpieza y *encoding* de las columnas se debe realizar sin hacer uso de la librería**.

Actividades

A continuación se detallan las tareas que deberán realizar:

1. Implementación del algoritmo [2 pts]

Deberán implementar el algoritmo de *Clustering Aglomerativo*, partirán tomando cada dato como un *cluster* aislado, para luego computar y almacenar las distancias entre *clusters* en una **matriz de distancias**. Luego, en cada iteración, deberán unir los *clusters* más cercanos que encuentren y computar nuevamente esta información en la matriz de *distancias*. Este proceso se repite hasta que, finalmente, queda un solo gran *cluster*. A continuación se explicará en detalle el paso a paso:

1.1 Distancias:

El algoritmo de *agglomerative clustering*, se basa en fuertemente en **calcular distancias** entre los distintos puntos o *clusters*. En tu implementación del algoritmo, tendrás que definir dos métricas en relación a esto:

1. **Métrica de distancia:** Se refiere a cómo se calculará la distancia entre dos puntos, utilizarán la distancia `Euclidiana` y la de `Manhattan`. Su implementación **debe soportar como parámetro cualquiera de las dos métricas** y calcular las distancia de acuerdo a ella (en otras palabras, es obligación implementar las dos distancias).
2. **Linkage:** Hace referencia a cómo (o desde dónde) medir la distancia entre dos *clusters*, deberán implementar los siguientes dos métodos:
   - `Complete:` Calcula la **distancia máxima** entre todos los pares de puntos de ambos *clusters*.

     $$D(C_a, C_b) = \max d(i, j),\ i \in C_a, j \in C_b$$

   - `Centroid:` Calcula la **distancia entre dos puntos representativos** de cada *cluster*, estos son calculados como el promedio de todos los puntos que hay en el *cluster* (centroides).

     $$D(C_a, C_b) = D(\mu_a, \mu b)$$

     Al igual que en el caso anterior, el **algoritmo debe recibir como parámetro cualquiera de estos criterios** y utilizarlo correctamente al momento de formar los *clusters* (deben implementar los 2 criterios).

Desde estas definiciones, en cada iteración de tu algoritmo deberás aplicar la métrica escogida sobre los *clusters* que existan en dicha iteración y, según la menor distancia recogida, unir los *clusters* asociados. Esta información de distancias la debes guardar en un objeto asociado llamado, *matriz de distancias*. Para

la tarea, no hay una restricción de la implementación del objeto (pueden ocupar lista ligadas, un árbol, diccionarios, etc.)

1.2 Clustering:

Tu algoritmo debe recibir los parámetros `n_clusters` y `max_dist`. El primero de estos determina cuántos *clusters* generará tu algoritmo, la idea básica es retornar las asociaciones de datos que tu algoritmo tenía cuando había esa cantidad de *clusters*. Por otro lado, el segundo parámetro tiene uso solo si el primero es `None`, en este caso no se deberá buscar por un específico número de *clusters*, sino que se deberán retornar las asociaciones hechas cuando el siguiente *merge* de *clusters* tenga una distancia mayor a la del parámetro.

En resumen, la clase `ClusteringAglomerativo` deberá recibir los siguientes parámetros:

- `distance_metric`: euclideana, manhattan.
- `linkage`: complete, mean point.
- `n_clusters`: número de clusters a conseguir (puede ser `None`).
- `max_dist`: si `n_clusters` es `None` determina un *threshold* para la distancia entre *clusters* tomada para predecir datos.

Para esta parte de la tarea sólo podrán hacer uso de las librerías de `python` **numpy** y **pandas**.

2. Aplicación del algoritmo [1.5 pts]

En esta parte se espera la aplicación y ejecución de su algoritmo utilizando distintas técnicas de preprocesamiento y parámetros. En general, deben realizar los siguientes pasos:

1. **Preprocesamiento de datos**: limpieza de datos y normalización de datos, se deben tomar decisiones respecto a columnas significativas.
2. **Reducción de dimensionalidad**: se genera una segunda versión del *dataset*, obtenida luego de realizar una reducción de dimensionalidad utilizando `PCA` para tener 2 dimensiones por dato. En este paso podrán hacer uso de la librería `sklearn`.
3. **Entrenamiento**: Para cada *dataset* (el reducido y el original), deberás probar 3 configuraciones distintas del algoritmo implementado (con distintos parámetros de *linkage*, *distance_metric* y *n_clusters*). De este modo, tendrás 6 modelos de *clustering* distintos: (1) *dataset* sin reducir con *set* de parámetros 1, (2) *dataset* sin reducir con *set* de parámetros 2, (3) *dataset* sin reducir con *set* de parámetros 3. Luego lo mismo, pero con el *dataset reducido* (ocupando los mismos tres *sets paramétricos* que con los datos sin reducir).

Tengan en mente que la elección de sus modelos de *clustering* tiene como objetivo que luego puedan determinar qué técnica estudiada funciona mejor con el *dataset* entregado.

3. Comparación de resultados [0.5 pto]

En esta parte de la tarea deberán, siguiendo los pasos de la parte 2, comparar los resultados obtenidos por los 6 modelos del algoritmo que ustedes entrenaron.

Para comparar calcularán la métrica *Silhouette Score* de la librería `sklearn`, luego tendrán que comentar y dar una intuición de los resultados obtenidos desde una mirada *dataset reducido* vs el *dataset original*. Finalmente, deberán identificar la mejor configuración para cada uno de los *datasets*.

NOTA: En esta sección pueden utilizar la librería `sklearn` para la implementación de la métrica.

4. Visualización [1.0 pto]

Se realizarán dos visualizaciones a partir de los resultado obtenidos para el mejor modelo para el *dataset reducido* y para el *dataset original*:

4.1 Dendrograma:

Deberás generar los respectivos ***Dendrograma*** (ver la Figura 1). En esta representación, el eje Y corresponde a la distancias de unión en el momento que se creo cada *cluster* y el eje X son los distintos *datapoints* iniciales (también llamados hojas). Podemos ver que el corte que se hizo generó 3 *clusters* (uno de cada color).

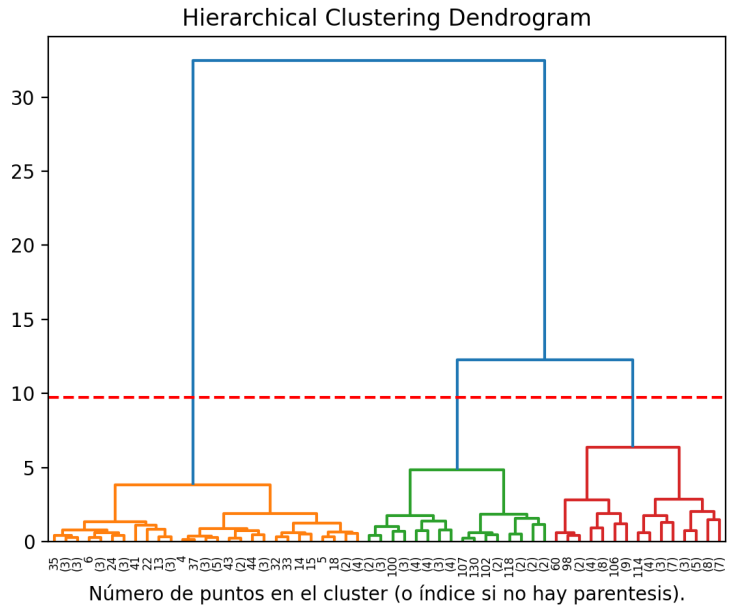

Figure 1: Dendrograma

Para hacer esta visualización podrán hacer uso de las funciones de dos funciones del módulo `scipy.cluster`: `hierarchy.linkage` y `hierarchy.dendrogram`. Estas funciones se encargarán de plotear tu dendrograma desde la matriz de distancias condensadas, una idea de su funcionamiento es el siguiente:

```
linkage_matrix = linkage(condensed_dist_matrix, method='complete')

dendrogram(linkage_matrix)

plt.show()
```

Notar que en este, el método `linkage` recibe como parámetro `method` el tipo de *linkage* que se utiliza para calcular distancia entre *clusters* (los nombre calzan con los definidos en la sección de *Distancias*).

Finalmente, cabe destacar que este *plot* puede ser algo caótico dependiendo del número de nodos y uniones que se quieran visualizar, para hacer una visualización más limpia del gráfico pueden verificar los parámetros `truncate_mode` y `p` de la función `dendrogram` o pueden pasar una matriz de distancia de una iteración más avanzada de su algoritmo (una en la cual exista una menor cantidad de nodos).

4.2 Clusters:

Finalmente, deberán hacer la visualización de los *clusters* obtenidos en los dos modelos. El gráfico debe señalar los *clusters* retornados por tu algoritmo, esta debe ser en 2D y debe mostrar claramente cada *datapoint* con el color representativo para el *cluster* al que pertenece, como se muestra en la figura 2. La visualización debe incluir la leyenda con los datos de qué color corresponde a qué cluster. Para la visualización del modelo entrenado desde *dataset original* deben usar `PCA` para reducir el *dataset* a 2D después de haber obtenido los *clusters* en alta dimensionalidad. Esto es simplemente para poder visualizarlos.

NOTA: En esta sección pueden utilizar la librería `sklearn` para la implementación de `PCA`.

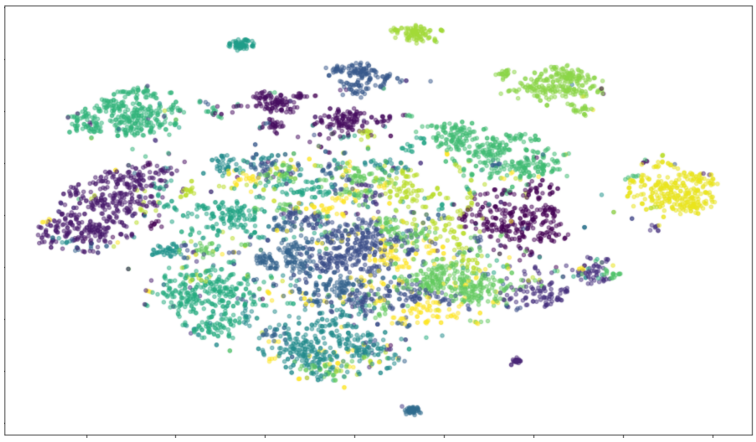

Figure 2: Ejemplo visualización de clusters

5. Análisis [1 pto]

En esta parte deberá responder las siguientes preguntas sobre las distintas toma de decisiones realizadas, haciendo uso de un máximo de 5 líneas (se realizará descuento si se sobrepasa el máximo permitido):

1. ¿Cómo trabajo los datos tipo `Nan` y datos defectuosos?
2. ¿Utilizó todos los atributos o dejó algunos afuera? ¿Por qué?
3. ¿Qué diferencias observa al usar los diferentes parámetros? ¿Cuál resultó ser el más apropiado? ¿Por qué? Comente en función de la métrica calculada y la visualización.
4. ¿Qué observó al usar la métrica de evaluación *Silhoutte Score*? Comente.
5. ¿Qué puede decir sobre la segmentación generada? ¿Existe algún tipo de paciente en particular que resulta ser readmitido en el hospital? Comente y justifique gráficamente.

Formato

El archivo `.ipynb` **que subas debe seguir, estrictamente, el siguiente formato:**

1. **Preprocesamiento de datos**: Se deben explicar los pasos realizados y las decisiones tomadas.
2. **Declaración de clase**: en esta sección se define la clase `ClusterAglomerativo`. En particular, esta clase debe contar con los siguientes métodos:
   - `init`: inicializa la clase del algoritmo, esta debe recibir los parámetros `distance_metric`, `linkage`, `n_clusters` y `max_dist`.
   - `fit_predict`: parte con una matriz de distancias vacía y, según los atributos de *linkage* y métrica de distancia, ir llenando esta matriz. Finalmente, debe retornar los *labels* o *clusters* asociados a cada dato. Tiene como *input* solo a la matriz X.
   - `condensed_distance_matrix`: desde la matriz de distancia computada deben retorna una versión condensada de esta. Una matriz condensada de distancias es el triangulo superior derecho de la matriz de distancia agrupado como un vector de una dimensión. Se van concatenando fila por fila los elementos de la matriz, en orden, sin los elementos de la diagonal.
   - `plot_dendrogram`: se encarga de la visualización del dendrograma.
3. **Comparación de resultado**s: En esta sección deberán realizar la ejecución del algoritmo usando distintos parámetros y calculando la métrica correspondiente sobre los resultados obtenidos. Finalmente, deberán identificar la mejor combinación para cada *dataset*.
4. **Visualización**: Se hace el *plot* del *dendrograma* y los *clusters* generados.

5. **Análisis**: Deberán responder las preguntas señaladas usando celdas en estilo MARKDOWN de JUPYTER NOTEBOOK. puede que este link les sea de ayuda.

Nota: Para la implementación del algoritmo en si sólo se podrán utilizar las librerías **numpy** y **pandas**. Para la visualización pueden ocupar librerías gráficas como **matplotlib**, **seaborn** o **altair**. **Solo en las secciones estrictamente señaladas** podrán hacer uso de **sklearn** y **scipy**. El uso de cualquier otra librería no está permitido y será sancionado.

Entrega

A la hora de subir tu tarea, el **único archivo** de código que debes subir es **el archivo .ipynb**, este debe seguir el formato comentado en la sección anterior. Además, en el mismo debe ir el informe con todo el análisis esperado en la tarea.

Finalmente, **todo el código debe venir ya ejecutado**, es decir, cada celda con su respectivo *output*.

Descuentos

Se aplicarán los siguientes descuentos:

- Hasta 0.5 puntos por no documentar clara y ordenadamente las funciones que implementen.
- 0.5 por no respetar el formato mencionado para el `.ipynb`.
- 0.5 por subir archivos que nos sean `.ipynb`, `.gitignore` o `readme.md`.
- 0.5 por exceder el número de líneas permitidas en la sección análisis.

References

Sakshi Patel, Shivani Sihmar, and A. Jatain. A study of hierarchical clustering algorithms. *2015 2nd International Conference on Computing for Sustainable Global Development (INDIACom)*, pages 537–541, 2015.