Aprendizaje no supervisado

Lección 6: Exploración, segmentación e interpretación de los datos después de agruparlos

Introducción

Una compañía trasnacional de supermercados desea agrupar a sus establecimientos no solo por su ubicación geográfica, sino teniendo en cuenta múltiples variables como ingresos, gastos, número de clientes, tipo de cambio de la moneda, entre otras. Ya sabes agrupar los datos, pero es necesario también mostrarles a tus líderes los resultados obtenidos. ¿Cómo preparas una presentación con los resultados más relevantes para la toma de decisiones?

Recuerda que este proceso de agrupar datos es no supervisado, por lo tanto, hay muchos componentes subjetivos desde que se inicia el agrupamiento hasta la presentación de los resultados. Es por eso por lo que la forma en que muestres los resultados será fundamental para extraer información relevante de todo esto.

En este tema estudiarás cómo segmentar finalmente el conjunto de datos y, lo más importante, qué visualizaciones puedes mostrarles a los clientes, ya sea para que ayuden en un punto en particular del proceso o para la toma final de decisiones. Algunos de los contenidos que verás es posible que ya los domines, pero ahora aprenderás como usarlos en este contexto del agrupamiento de datos.

Tecnológico de Monterrey © 2023 | The Learning Gate

Segmentación de los datos basado en los resultados de agrupamiento

Volvamos al mercado inmobiliario, que es un buen ejemplo de cómo se pueden segmentar los datos de un mercado o modelo de negocio específico. Ten presente que estos datos de inmuebles incluyen información de distinta naturaleza, como la ubicación del inmueble o su tamaño. Pero ya sabes qué hacer con esas variables, conoces cómo agrupar los datos, incluso puedes seleccionar un número de grupos que te parezca interesante.

¡Comencemos!

Introducción

¿Cuál es entonces el reto que queda por resolver?

Debes asegurarte de que los resultados son correctos y que apoyarán la toma de decisiones. Este es tu objetivo principal, mientras que el proceso de agrupamiento es una herramienta para lograrlo. Por lo tanto, **es necesario visualizar los resultados, revisar cómo se comportan las variables en cada grupo, comparar los resultados con algunas opciones más**, por ejemplo, cambiando el número de grupos, entre otras estrategias.

Este paso final se puede dividir en dos partes: **una será el análisis que debes hacer para estar seguro de los resultados, y el otro es presentar esos resultados al usuario final**. De tu análisis de los resultados extraerás lo más relevante, por lo que usarás seguramente las mismas herramientas en ambos casos. Este estudio lo dividiremos en dos, primero verás cómo segmentar datos y analizar el comportamiento de las variables, y luego algunas herramientas más de visualización.

Segmentación de datos

En el tema anterior seguramente seleccionaste 6 grupos para la base de datos “housdata.csv”. Si haciendo tus propias pruebas notaste un número de grupos mejor, no te preocupes, puedes hacer el mismo análisis para

cualquier número de grupos. Recuerda que este proceso se puede repetir hasta estar seguros de encontrar información relevante.

- **Agrupación**

Agrupemos entonces los datos con *K-means* para *k=6*. Los datos que agruparemos ya tienen las transformaciones correspondientes, debido a que nos estamos centrando solamente en la etapa final. Un ejemplo de código para agrupar ya lo dominas, y puede ser el siguiente:

```
k = 6  # Debido principalmente a las gráficas K-Elbow
k_means = cluster.KMeans(n_clusters=k)
y_pred1 = k_means.fit_predict(X)
s1 = metrics.silhouette_score(X, y_pred1)
single_linkage = cluster.AgglomerativeClustering(linkage="single", n_clusters=k)
y_pred2 = single_linkage.fit_predict(X)
s2 = metrics.silhouette_score(X, y_pred2)
spectral = cluster.SpectralClustering(n_clusters=k, affinity="nearest_neighbors")
y_pred3 = spectral.fit_predict(X)
s3 = metrics.silhouette_score(X, y_pred3)
fig, (ax0, ax1, ax2) = plt.subplots(ncols=3, figsize=(15,4))
ax0.scatter(X_train[:, f], X_train[:, 1], c=y_pred1, cmap=plt.cm.Spectral)
ax0.set_ylim(0, 0.5e7)
ax0.set_title('K means: '+"{:.2f}".format(s1))
ax0.set_xlabel(t)
ax0.set_ylabel('price')
ax1.scatter(X_train[:, f], X_train[:, 1], c=y_pred2, cmap=plt.cm.Spectral)
ax1.set_title('Single Linkage: '+"{:.2f}".format(s2))
ax1.set_ylim(0, 0.5e7)
ax1.set_xlabel(t)
ax1.set_ylabel('price')
ax2.scatter(X_train[:, f], X_train[:, 1], c=y_pred3, cmap=plt.cm.Spectral)
ax2.set_ylim(0, 0.5e7)
ax2.set_title('Spectral Clustering: '+"{:.2f}".format(s3))
ax2.set_xlabel(t)
ax2.set_ylabel('price')
plt.show()
```

- **Interpretación**

En el código anterior, en la variable *y_pred1* aparecen las nuevas etiquetas que el algoritmo *K-means* le asignó a cada dato. Es decir, con esas

etiquetas es que debemos particionar o segmentar los datos. Esos son nuestros grupos. Si nuestros datos están almacenados en un *dataframe* de nombre *house* como en el tema anterior, entonces podemos adicionarle una columna con estas nuevas etiquetas, de la siguiente manera:

```
house['groups'] = y_pred1
```

- **Conclusión**

Ahora tu *dataframe* ya tiene la información de los grupos, y es lo que necesitamos para empezar a analizar el resultado. Ten en cuenta que este análisis ya es después de analizar cuál podía ser un número de grupos adecuados. Por lo que ahora revisarás que precisamente ese resultado tenga sentido para la toma de decisiones.

Análisis final

Hay varias maneras de hacer este análisis final, lo que verás a continuación es solo una de ellas. Tú mismo puedes crear tu propia metodología o secuencia de pasos en esta etapa, dependiendo de tu objetivo final. Lo que haremos de momento es lo siguiente:

1. Construir un nuevo *dataframe* con las variables como filas y los grupos como columnas.
2. Por cada uno de los grupos calcularás algún estadístico de las variables, por ejemplo, el promedio de cada variable.
3. Terminado el paso anterior puedes mostrar este nuevo *dataframe* como guía del comportamiento de las variables en cada grupo, y como resultado de la segmentación en sí.

a) Por lo tanto, una celda de este nuevo *dataframe* se va a corresponder con un grupo y una variable, y ahí copiarás el estadístico.
b) Si la variable no es numérica, puedes incluir algún otro estadístico como la moda.

c) Tomaremos las variables originales, sin transformar, creemos que es mejor en este punto

Representar en código

¿Cómo puedes hacer todo eso en el código? No es tan complejo, primero puedes crear una lista de *dataframes* donde filtres el *dataframe* original según las etiquetas de los grupos. Luego, realiza un código que replique el algoritmo o estrategia anterior. Todo junto puede quedar así:

```
# crear una lista de dataframes
houses = [house[house['groups']==g] for g in range(k)]
```

```
grouped = pd.DataFrame() #Inicializar un dataframe vacío
# Añadir una columna para poner el nombre de las variables en cada fila
grouped['Features'] = house.columns[:-1]
# Algoritmo para llenar el dataframe con el promedio de cada variable dentro de cada grupo
for g in range(k):
    row = []
    for col in grouped['Features']:
        if houses[g][col].dtype != 'object':
            row.append(np.round(houses[g][col].mean(),2))
        else:
            row.append(houses[g][col].value_counts().keys()[0])
    grouped['Group'+str(g)] = row
# Visualizar el dataframe
grouped
```

Ese código debe generar un *dataframe* como se muestra a continuación:

	Features	Group0	Group1	Group2	Group3	Group4	Group5
0	date	2014-07-09 00:00:00	2014-06-26 00:00:00	2014-06-24 00:00:00	2014-07-08 00:00:00	2014-05-12 00:00:00	2014-06-23 00:00:00
1	price	1.23294e+06	409728	624259	534945	1.45162e+06	378412
2	bedrooms	4.39	3.36	3.88	3.48	3.36	2.67
3	bathrooms	3.47	1.89	2.29	2.53	2.86	1.35
4	sqft_living	4128.61	1811.55	2354.59	2320.01	3471.94	1306.26
5	sqft_lot	47407.6	19668.6	10115.3	10798.6	22130	8185.09
6	floors	1.88	1.11	1.26	2.06	1.65	1.18
7	waterfront	0	0	0	0	1	0
8	view	0.99	0.06	0.5	0.06	3.55	0.08
9	condition	3.34	3.7	4	3.09	3.45	3.39
10	sqft_above	3420.68	1484.73	1550.15	2259.05	2627.55	1168.64
11	sqft_basement	707.94	326.83	804.43	60.96	844.39	137.62
12	yr_built	1985.83	1970.28	1947.13	1998.25	1962.55	1947.37
13	yr_renovated	601.28	740.99	881.15	433.74	907.97	1356.55
14	street	8434 W Mercer Way	24345 35th Pl S	2739 31st Ave S	2520 Mulberry Walk NE	5044 Butterworth Rd	6520-6588 8th Ave NW
15	city	Bellevue	Bellevue	Seattle	Seattle	Vashon	Seattle
16	statezip	WA 98040	WA 98034	WA 98115	WA 98029	WA 98070	WA 98117
17	country	USA	USA	USA	USA	USA	USA

Figura 2. *Dataframe* con los grupos en las columnas y el promedio de cada variable dentro de cada grupo por las filas. Generado en Colab.

¿Qué conclusiones puedes obtener de este nuevo *dataframe*?

Diríamos que varias. Si ves con detenimiento, y recuerdas que la variable más importante en esta base de datos es el precio, puedes notar que hay tres grandes grupos de precio que luego cada uno se divide en dos grupos más pequeños. Nota que hay dos grupos donde el promedio del precio de los inmuebles sobrepasa el millón de dólares, podríamos decir que ese es un gran grupo donde están los inmuebles más caros. Además, hay otros dos grupos de casas que cuestan más de medio millón y un tercero con casas por debajo del medio millón. Estas últimas serían las casas más baratas.

¿Y qué puedes decir de esos grupos más pequeños? Veamos otro detalle:

- **Variable *waterfront***

¿Y qué puedes decir de esos grupos más pequeños? Veamos otro detalle, en particular presta atención a la variable *waterfront*. Seguramente notaste que en solo un grupo su valor es 1 y en el resto de los grupos 0. Esta es una variable booleana, de manera que tener promedio de 1 significa que todos los inmuebles en ese grupo tienen *"waterfront"*. Además, ¡este es el grupo de las casas más caras! Puede parecer poco sorprendente que las casas más caras tengan vista al lago, o al mar, pero que el algoritmo lo detecte automáticamente sí es relevante.

Este es un ejemplo donde el algoritmo de agrupamiento encuentra en los datos un comportamiento que posiblemente era esperado, de manera que valida lo adecuado de su salida, o sea, de esta partición de los datos.

- **Variable *view***

Hay más detalles que llaman la atención, por ejemplo, la variable *view* también tiene más promedio en las casas más caras. Al parecer, la vista es un factor fundamental para determinar el precio de estos inmuebles.

¿Cómo podría aprovechar el cliente final estas conclusiones? Por ejemplo, puede asegurarse de que es una buena idea invertir en inmuebles con buena vista. Si quizás no tenía muchas dudas sobre esta conclusión como se comentó, ahora puede estar seguro de ello porque los datos lo confirman.

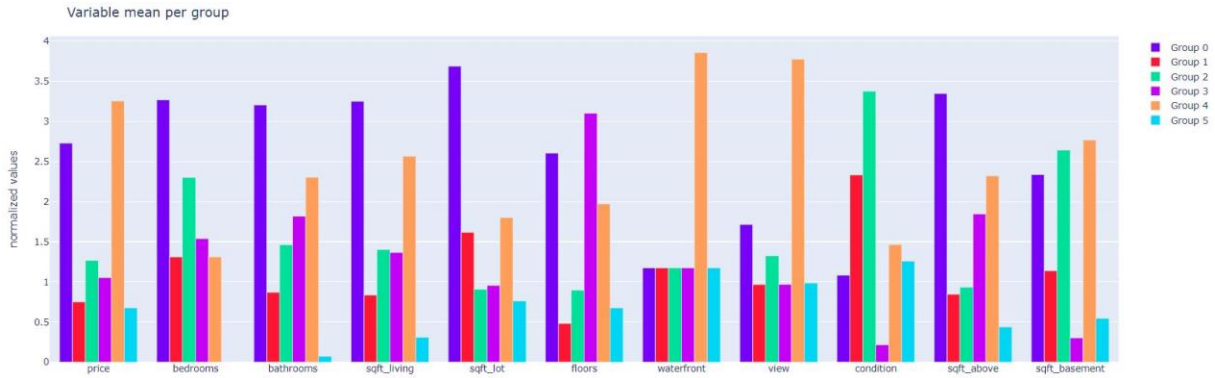
Finalmente, a modo de ejercicio, revisa las variables que se relacionan con el tamaño de la propiedad. Hay varias de esas variables. El objetivo principal sería encontrar alguna relación entre ellas y el precio de las viviendas en cada grupo. Algo que puede esperarse es que casas más grandes sean más caras, sin embargo, esta conclusión no es tan evidente en la variable *sqft_basement*, de modo que el usuario final sabrá si esta información puede ser de utilidad para la toma de decisiones.

Como ves, un análisis exhaustivo del comportamiento de las variables por grupos puede ser muy relevante, no solo para comprobar que el agrupamiento tiene algún sentido, sino también para presentarle resultados al cliente. Aunque en el próximo subtema veremos qué visualizaciones pueden ser útiles, vamos a incluir en este una de esas visualizaciones relacionadas con las variables para reafirmar lo aprendido.

Gráfica de barras
Gráfica de barras para el promedio de las variables en los grupos

Ver los datos en la tabla sin dudas te ha ayudado a encontrar varios aspectos importantes en los datos. Sin embargo, si fueran muchas más variables sería difícil ir analizando cada valor de promedio e ir haciendo esa comparación un tanto manual. ¿Qué puedes hacer entonces? Puedes aplicar lo que ya conoces sobre métodos de visualización de resultados. Un gráfico un tanto simple que puede ser muy útil es representar la

información de la tabla en una gráfica de barras, como se muestra a continuación:



Nota algunos detalles en la gráfica. Lo primero es que los valores están normalizados, eso es para que todas las barras tengan más o menos la misma escala. Otro detalle es que las variables están en el eje x y agrupan la información de cada grupo. Finalmente, los colores representan a los grupos.

Como seguramente notaste, los grupos cero y cuatro corresponden a los inmuebles más caros, por lo que las barras naranja y azul son mayores en la mayoría de las variables. ¿Por qué?, pues porque estas son las casas más caras, de mayor espacio, con más habitaciones y con mejor vista en la mayoría de los casos.

- **Generación de gráfica en código**

¿Cómo puedes generar esta gráfica en el código? Lo primero es que se está usando una librería llamada *plotly*, muy útil en estos casos, ya que es más cómoda que *matplotlib* para gráficas más complejas. Veamos:

```
dfg = grouped.transpose()
dfg.columns = dfg.loc['Features']
dfg = dfg.drop('Features')
dfg = dfg[dfg.columns[1:12]]
dfg

Xt = StandardScaler().fit_transform(dfg.to_numpy()
Xt = Xt + abs(np.min(Xt))
```

Preparación de los datos

Antes de analizar el código que dibuja la gráfica vamos a preparar los datos. Las cuatro primeras líneas transponen el *dataframe* para que sea más cómo el procesamiento. Mientras las últimas dos garantizan que se normalicen los datos y que no haya barras con valores negativos para poder compararlas mejor. Nota que en este proceso solo se incluyen las variables numéricas.

Análisis del código

Ahora analiza este código. La primera línea es para importar la librería que gráfica. A continuación, aparecen las líneas de código que nos ayudan a construir la gráfica. Fíjate cómo se agrupan las barras por variable y cómo se repite este proceso por cada variable. Las últimas líneas son para configurar los textos y mostrar la figura. Desde luego, en internet encontrarás mucha información acerca de esta librería.

```
import plotly.graph_objects as go
fig = go.Figure()
for j in range(len(dfg)):
  fig.add_trace(go.Bar(
      y=Xt[j,:],
      x=dfg.columns,
      name='Group '+str(j)
  ))fig.update_layout(
    title='Variable mean per group',
    yaxis_title='normalized values',
    margin=dict(l=50, r=50, t=50, b=50))fig.show()
```

Visualizaciones asociadas a los grupos obtenidos

Una agencia de compraventa de inmuebles te contrata junto a tu equipo de científicos de datos para que los apoyes segmentando el mercado. El jefe de la sección de ventas debe vender cuanto antes los inmuebles y necesita poner un precio justo. Tu labor es revisar los precios de inmuebles parecidos y ayudar con una propuesta. Para que la propuesta sea adecuada debes apoyarla con un riguroso análisis de los resultados y mejor si algunas visualizaciones ayudan a comprender las conclusiones que se obtienen.

¡Comencemos!

Introducción
En el subtema anterior ya viste cómo te puede ayudar una visualización para comprender o mostrar los resultados obtenidos de una manera adecuada. En este subtema estudiarás varias visualizaciones más, algunas más avanzadas, pero todas son importantes, no es simple distinguir a priori cuál te será más útil. No pierdas de vista en ningún momento que este es un proceso no supervisado y eso hace que todo el análisis que se haga será bueno para la interpretación de los resultados.

Importancia de las variables
Cuando realizaste la gráfica de barras para analizar cómo se comporta el promedio de cada variable en cada grupo, consideraste a cada variable por separado. Sin embargo, viste la influencia de cada una de ellas en todos los grupos, ya que agrupaste por grupo.

¿Qué se puede obtener si el centro del estudio es ahora los grupos en lugar de las variables?

Esto nos mostraría la importancia que tiene una variable dentro de un grupo en relación con otras variables de ese mismo grupo. Aunque parezca un tanto contradictorio, este análisis nos ayuda a medir la importancia de las variables, en particular de ellas dentro un mismo.

- **Gráfico de cajas y botones**

El gráfico que se mostrará es de cajas y bigotes. La idea es mostrar no solo el promedio de las variables por grupo, sino comparar además las distintas variables. Adicionalmente, este gráfico permite analizar la desviación estándar de los valores, de manera que se pueda analizar la varianza dentro de cada grupo. Para hacerlo, otra vez comenzaremos con la preparación de los datos para hacer el gráfico. Analiza el código siguiente:

```
house1 = house.sort_values('groups')
groups = house1['groups']
house1 = house1[house1.columns[1:12]]

Xt = StandardScaler().fit_transform(house1.to_numpy())

x = ['Group '+str(g) for g in groups]
```

En la primera línea se ordenan las etiquetas de los grupos. Esto servirá para que salgan ordenados los valores del eje *x*, que por cierto son los nombres de cada grupo.

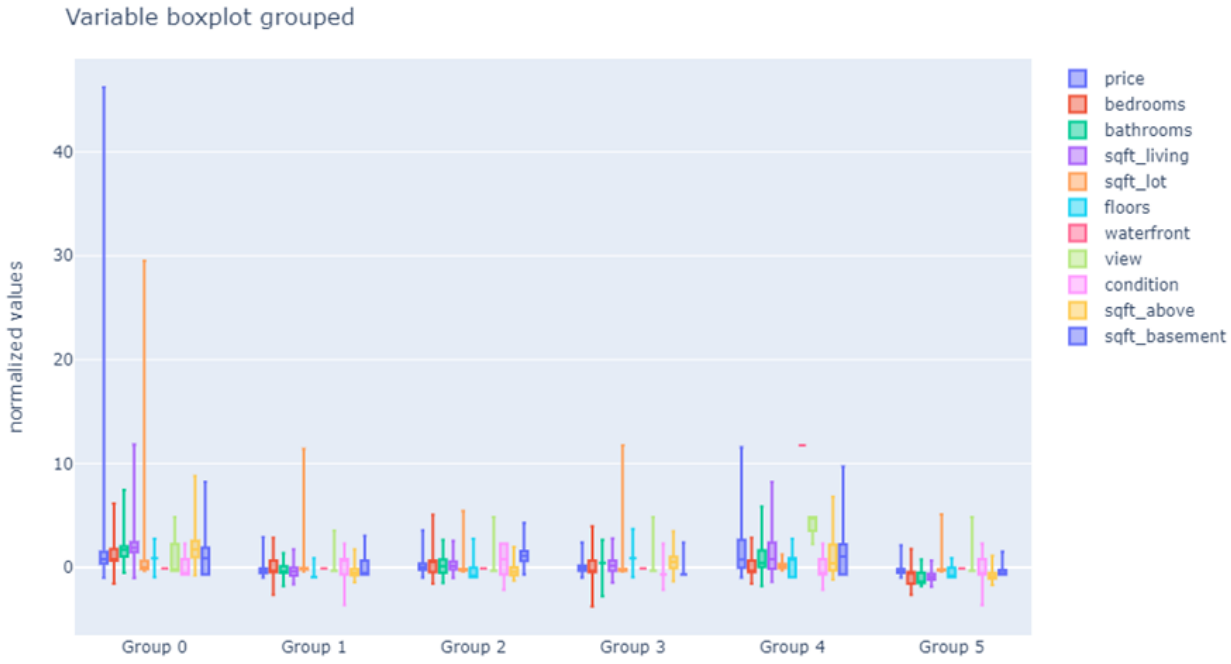
La tercera línea se queda con las variables numéricas solamente.

La última construye una lista con el nombre que se le asignará a cada grupo. Estos nombres irán en el eje *x* como ya se mencionó.

A continuación, puedes revisar el código que genera la figura. Lo diferente es el tipo de gráfico y algunos detalles de configuración, como *boxmode='group'*, que es fundamental para lograr lo que se quiere. Ten en cuenta que en esta librería cada gráfico tiene sus detalles diferentes.

```
fig = go.Figure()
for j in range(len(house1.columns)):
  fig.add_trace(go.Box(
      y=Xt[:,j],
      x=x,
      boxpoints=False,
      name=house1.columns[j]
  ))
fig.update_layout(
    title='Variable boxplot grouped',
    yaxis_title='normalized values',
    boxmode='group',
)
fig.show()
```

La figura generada es la siguiente:



Lo primero que debes conocer antes de analizar el gráfico, es que esta librería produce gráficos interactivos, es decir, puedes hacer zoom sobre el gráfico y ver toda la información a detalle. Eso es importante porque

como ves la figura parece ser demasiado pequeña para sacar conclusiones.

Aun así, saltan a la vista algunos detalles. Por ejemplo, el precio en el "Grupo 0" tienen una gran desviación, así como la variable ***sqft_lot***. Aunque este es uno de los grupos de casas caras, es posible que algunas no lo sean tanto. En el otro grupo de casas caras resalta otra vez la variable ***waterfront***, que tiene color rojo en el gráfico. También, a modo de ejercicio, sigue revisando el gráfico y piensa qué otras conclusiones serían de interés.

Método PCA

¿Ya conoces el método PCA (Salo et al., 2019)?

No te preocupes si no lo dominas, como resumen diremos que las siglas responden a Análisis de Componentes Principales y su **objetivo** es reducir la dimensionalidad de un conjunto de datos dado un número de dimensiones finales. ¿Y por qué es interesante en este tema de agrupamientos? Bueno, la verdad este método es bastante importante en este contexto porque también es un algoritmo no supervisado. Sin embargo, no es un método de agrupamiento como tal, sino de selección de variables.

Entonces, ¿cómo lo puedes aprovechar? Hay dos aplicaciones fundamentales. Una es reducir la dimensionalidad y agrupar los datos en un conjunto reducido de variables. ¿Qué tanto se deben reducir? Es una pregunta subjetiva y puede que la respuesta la encuentres luego de analizar los resultados como ya hiciste antes. Luego verás esta aplicación, pero primero hablemos de la segunda forma en que te puede ayudar.

- **Ejemplo**

Cuando en temas anteriores seleccionaste dos variables para ver el resultado del agrupamiento a partir de plotear esas variables como puntos en el plano, también coloreaste los puntos con colores diferentes para

cada grupo. En ese momento, esas dos variables se seleccionaron manualmente.

PCA precisamente nos puede ayudar a seleccionar las dos variables, o componentes, más importantes de los datos, y a partir de ahí hacer el ploteo con lo que se consideran las dos variables principales de los datos. Por lo tanto, algo que haremos es transformar los datos para reducir su dimensión y luego ver cómo queda ese ploteo. Para ello analiza el código:

```
from sklearn.decomposition import PCA

X2 = PCA(n_components=2).fit_transform(X)



plt.scatter(X2[:, 0], X2[:, 1], c=y_pred1, cmap=plt.cm.Spectral)
plt.title('Visualized with PCA')
plt.show()
```

La primera línea importa el método y la segunda transforma los datos creando un nuevo *dataset* ahora en la variable *X2*. Los nuevos datos solo tienen dos columnas, e incluiremos las etiquetas generadas por *K-means* para plotearlas y analizar si se obtuvo un buen resultado.

- **Figura 1**

Analiza la figura. Es necesario repetir que los datos se agruparon con todas las variables y *PCA* solo se utilizó para plotear el resultado con dos de ellas, ya que con *PCA* se pueden obtener las componentes principales de mayor varianza en los datos. En la imagen, puedes notar seis grupos de colores y, lo que es sorprendente, como casi no se solapan o se juntan puntos de distintos colores. Un detalle interesante es que los puntos verdes son pocos y están encima de los demás, por lo que estos puntos pueden corresponderse con las casas que tienen *waterfront*.

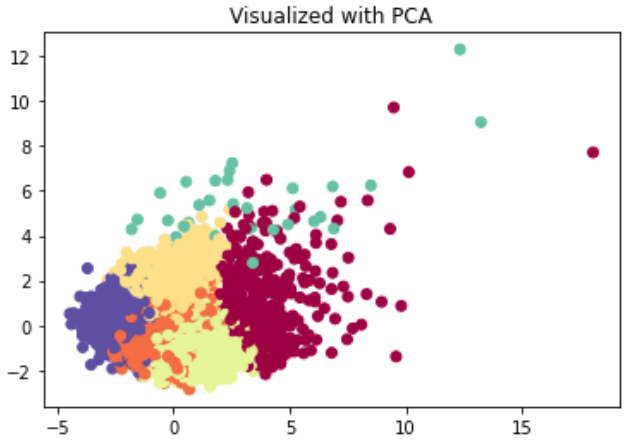


Figura 7. Datos agrupados con todas las variables y ploteados solo con las dos componentes principales. Generada en Colab.

Es probable que te preguntes: ¿qué sucede si agrupamos solamente las componentes principales de los datos? Tiene sentido porque esto ayuda a reducir el número de variables antes de agrupar.

Sin embargo, tiene el inconveniente de simplificar mucho los datos y hace que pierdas información valiosa en el proceso. Lo aconsejable es que haya un equilibrio entre el desempeño del algoritmo y la reducción de los datos.

- **Figura 2**

En la imagen, se ven los grupos más definidos, con menos solapamiento porque no hay puntos de colores distintos mezclados. ¿Es este resultado mejor que el anterior? No lo sabemos, habría que completar el análisis. Por lo pronto, puedes decir que no detecta el grupo de las casas caras con *waterfront*, así que seguramente no es mejor resultado. ¿Cómo harías el código para esto? Simple, agrupa los datos que quedaron transformados en la variable *X2*.

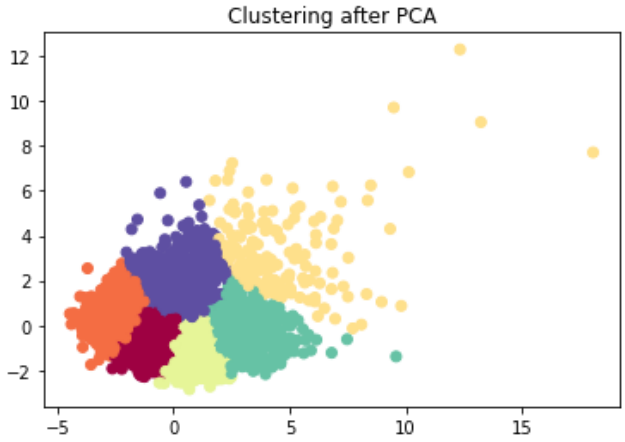


Figura 8. Datos agrupados después de reducir la dimensionalidad con PCA. Generada en Colab.

Método t-SNE

Finalmente, analizarás otra manera de reducir la dimensionalidad de los datos, pero más compleja, por lo que no se recomienda cuando hay muchas variables. Se trata del método *t-SNE* (Linderman et al., 2019), que son las siglas en inglés de "Incrustamiento de Vecino Estocástico t-distribuido".

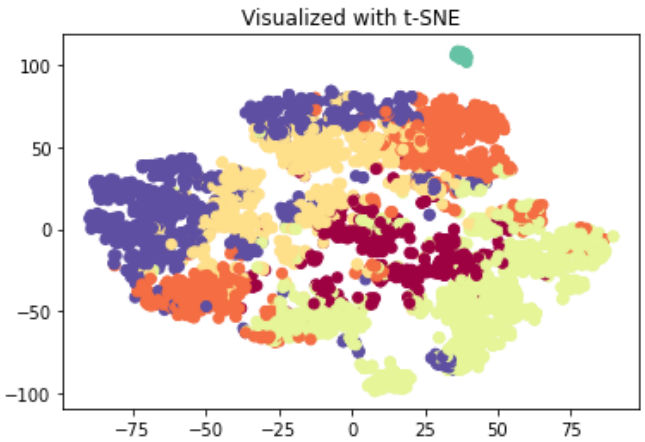
Este método reduce al mínimo la divergencia entre las distribuciones de pares de puntos. Aunque al igual que *PCA* este método se ha utilizado como parte del proceso de agrupamiento, nosotros solo lo usaremos como apoyo a las visualizaciones. ¿Dónde puedes encontrar una implementación de este método? Efectivamente, en *Scikit-learn*. Analiza el código para que veas cómo usarlo.

```
from sklearn.manifold import TSNE

tsne = TSNE(n_components=2, init='pca', random_state=0, method='exact', verbose=1).fit_transform(X)

plt.scatter(tsne[:, 0], tsne[:, 1], c=y_pred1, cmap=plt.cm.Spectral)
plt.title('Visualized with t-SNE')
plt.show()
```

Realmente, es muy parecido a la manera en que usas *PCA*. La mayor diferencia es en el tiempo de ejecución, ¡notarás que *t-SNE* se tarda demasiado! Usarás este método del mismo modo, es decir, primero debes

reducir dimensionalidad con *t-SNE* para luego agrupar los datos sobre las dimensiones que quedaron.

La imagen que se genera es bastante particular, veamos:



En esta imagen los grupos no están tan compactos como la anterior, pero se puede decir que hay una región para cada grupo, o cada color en este caso. Lo que sí llama mucho la atención es el pequeño grupo verde que está arriba y a la derecha. Seguramente se trata de ese grupo de casas caras con *waterfront*, ¿recuerdas?

Si dominas varias herramientas de visualización, te exhortamos a que analices cuál o cuáles pueden ser útiles en este contexto y asegúrate que puedas ejecutarlas.

Hasta aquí el curso de agrupamiento de datos. Después de tu estudio estás listo para transformar variables, seleccionar el número de grupos, segmentar los datos, y mostrar algunas visualizaciones para la toma de decisiones.

Recuerda que agrupar datos te ayuda a encontrar aspectos de interés en los datos que sean completamente nuevos, aspectos que ya se sospecha que existen, o aspectos que no se espera que existan. En tu rol como científico de datos seguramente necesitarás aplicar, en muchas ocasiones,

esto que has aprendido. Esperamos que puedas ir practicando el uso de estas herramientas en tu labor diaria.

Ideas para llevar

El proceso de agrupamiento es no supervisado y, por lo tanto, saber si los resultados que se obtienen son buenos o no es un criterio subjetivo. Es por eso, por lo que distintas estrategias de visualización y de interpretación de los resultados son necesarias. Revisa estas recomendaciones, algunas ya las viste durante el tema, pero te serán de mucha utilidad en tu rol:

- Mostrar los resultados del agrupamiento usando algún estadístico de las variables en los grupos es una buena opción, pero no la única para este propósito. Investiga otras maneras y, de ser posible, diseña tus propias estrategias.
- Recuerda que el método t-SNE se puede utilizar para reducir la dimensionalidad de los datos, pero se recomienda que no se utilice en grandes volúmenes de datos. PCA puede ser una alternativa en esos casos.
- Intenta crear una manera de realizar las visualizaciones automáticamente para comparar resultados diferentes, por ejemplo, agrupamientos con distinto número de grupos. Esto te será muy provechoso cuando necesites mostrar varias alternativas de resultado al usuario final. Incluso, puede ayudarte en tu propia interpretación de los resultados.

Material adicional

- https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html?highlight=pca#sklearn.decomposition.PCA
- https://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html

Bibliografía

Los contenidos de esta lección están basados en la siguiente bibliografía:

- Linderman, G. C., & Steinerberger, S. (2019). Clustering with t-SNE, probably. *SIAM Journal on Mathematics of Data Science*, *1*(2), 313-332.
- Salo, F., Nassif, A. B., & Essex, A. (2019). Dimensionality reduction with IG-PCA and ensemble classifier for network intrusion detection. *Computer Networks*, *148*, 164-175.