Basile Álvarez Andrés José

No. Cuenta: 316617187

Email: andresbasile123@gmail.com

Fecha: 07/10/21

Inteligencia Artificial

Grupo III

Semestre 2022-1

Reporte Práctica 3: Métricas de Distancia

Objetivo: Obtener las matrices de distancia (Euclidiana, Chebyshev, Manhattan, Minkowski) en Google Colab a partir de una matriz de datos.

Características Generales: Una medida de distancia es una puntuación objetiva que resume la diferencia entre dos elementos. Algunos algoritmos que utilizan métricas de distancia son: K vecinos más cercanos, Máquinas de Soporte Vectorial, Clustering (K means y Jerárquico), entre otros. Las medidas de distancia se utilizan en algoritmos para el *aprendizaje basado en instancias*, como, por ejemplo, para clasificar imágenes, reconocimiento facial, sistemas de recomendación, agrupamiento de elementos, entre otros.

La *distancia euclidiana* implica la aplicación del Teorema de Pitágoras para obtener la distancia entre dos puntos en el espacio euclidiano. La *distancia de Chebyshev* implica obtener el valor máximo absoluto de las diferencias entre las coordenadas de un par de elementos. La *distancia de Manhattan* se utiliza cuando se necesita calcular la distancia entre dos puntos n una ruta similar a una cuadrícula (con obstáculos). La *distancia de Minkowski* es una generalización de las anteriores y se utiliza para obtener la distancia entre dos puntos en un espacio *n-dimensional*.

En esta práctica, los datos a utilizar serán 202 registros de ingresos, gastos, ahorros de un grupo de personas. A continuación, mostramos a mayor detalle cada una de las columnas que integran nuestros datos:

- ingresos: son ingresos mensuales de 1 o 2 personas, si están casados.
- gastos_comunes: son gastos mensuales de 1 o 2 personas, si están casados.
- pago_coche
- gastos_otros
- ahorros
- vivienda: valor de la vivienda.
- estado_civil: 0-soltero, 1-casado, 2-divorciado
- hijos: cantidad de hijos menores (no trabajan).

- trabajo: 0-sin trabajo, 1-autonomo, 2-asalariado, 3-empresario, 4-autonomos, 5-asalariados, 6-autonomo y asalariado, 7-empresario y autonomo, 8-empresarios o empresario y autónomo
- comprar: 0-alquilar, 1-comprar casa a través de crédito hipotecario con tasa fija a 30 años.

Desarrollo

Primeramente, tenemos que definir aquellas bibliotecas de Python que nos serán útiles para importar, limpiar y analizar los datos contenidos en el archivo separado por comas *Hipoteca.csv*. Estas serán: *pandas* (manipulación y análisis de datos), *matplotlib* (para la creación de gráficas y visualización de los datos), *numpy* para utilizar vectores y matrices de *n* dimensiones y *scipy*, de donde utilizaremos *cdist* para el cálculo de las distancias.

Más tarde, importamos el archivo *Hipoteca.csv* y lo primero que hacemos es guardarlo en un DataFrame. La importación del archivo se realizó a partir de *Google Drive*, lo cual facilitará el poder almacenar los datos en la nube y utilizarlos sin la necesidad de descargarlos. Una vez importados los datos, mostramos el DataFrame que los contiene:

	ingresos	gastos_comunes	pago_coche	gastos_otros	ahorros	vivienda	estado_civil	hijos	trabajo	comprar
0	6000	1000	0	600	50000	400000	0	2	2	1
1	6745	944	123	429	43240	636897	1	3	6	0
2	6455	1033	98	795	57463	321779	2	1	8	1
3	7098	1278	15	254	54506	660933	0	0	3	0
4	6167	863	223	520	41512	348932	0	0	3	1
...	...	...	...	...	...	...	...	...	...	...
197	3831	690	352	488	10723	363120	0	0	2	0
198	3961	1030	270	475	21880	280421	2	3	8	0
199	3184	955	276	684	35565	388025	1	3	8	0
200	3334	867	369	652	19985	376892	1	2	5	0
201	3988	1157	105	382	11980	257580	0	0	4	0

202 rows × 10 columns

Figura 1: DataFrame *Hipoteca*.

Como los datos ya se encuentran preparados, comenzaremos aplicando la primera métrica de distancia: la distancia euclidiana. Para esta métrica, se pasa al método *cdist* el DataFrame *Hipoteca* dos veces, para calcular la métrica de distancia euclidiana de cada uno de los puntos con los demás puntos. Después, convertimos el resultado a un DataFrame y visualizamos los resultados redondeando a tres números después del punto decimal.

```
DstEuclideana = cdist(Hipoteca, Hipoteca, metric='euclidean')
MEuclideana = pd.DataFrame(DstEuclideana)
```

```
MEuclideana.round(3)
```

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	
0	0.000	236994.702	78577.840	260974.591	51769.581	39149.061	30003.798	206425.706	108991.941	76488.543	178296.795	27412.227	126257.750	68523.558	242085.849	94879
1	236994.702	0.000	315439.177	26550.528	287970.808	276141.622	207115.405	33742.472	345963.774	312810.380	59059.207	209746.028	114631.091	304468.897	5197.516	331850
2	78577.840	315439.177	0.000	339168.030	31494.808	39645.761	108564.128	284512.008	31548.759	17030.195	256835.625	105714.635	203466.010	13972.245	320537.561	18167
3	260974.591	26550.528	339168.030	0.000	312273.311	300095.494	231251.132	54727.356	369945.815	337121.576	84336.723	233623.418	136635.567	327877.979	22478.885	355839
4	51769.581	287970.808	31494.808	312273.311	0.000	15176.477	81054.393	257851.795	58617.026	24868.540	229053.970	79086.890	178017.315	28613.734	293036.029	44676
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
197	53923.596	275716.907	62456.927	301032.758	33981.853	40260.802	75868.017	247632.364	82392.341	49258.107	216778.301	76258.794	171687.417	62386.024	280668.664	70217
198	122858.124	357126.266	54616.720	381921.267	71302.673	85526.169	151476.306	327814.188	30605.694	47621.346	298081.659	149940.101	248861.649	68496.769	362145.363	37666
199	18967.999	249015.958	69848.439	273593.155	39655.592	31283.565	43417.279	219384.112	98206.746	64240.730	190029.780	42585.769	140750.709	62471.815	254069.717	84234
200	37975.571	261065.406	66722.600	286156.617	35401.101	34884.354	59413.858	232457.936	91190.756	56800.660	202049.132	59682.473	155678.258	63083.485	266056.205	77930
201	147421.532	380612.957	78717.768	405600.560	96032.257	110377.422	175732.011	351670.615	51056.467	72480.501	321575.116	174373.435	273195.721	92664.641	385606.431	61064

202 rows × 202 columns

Figura 2: DataFrame obtenido después de aplicar la distancia euclidiana al DataFrame *Hipoteca*.

En la imagen anterior, nos percatamos que el cálculo de la distancia euclidiana devuelve una matriz donde todos los elementos de la diagonal principal son iguales a cero. Esto se debe a que la distancia de un punto consigo mismo es igual a cero. También vemos que la matriz tiene más columnas que el DataFrame original y esto se debe a que justamente cada registro es comparado con todos los demás registros para obtener la distancia entre todos los puntos.

Haciendo una visualización de los primeros diez datos con *iloc*[0:10], nos percatamos de mejor manera como todos los elementos de la diagonal principal son cero.

```
MEuclideana.iloc[0:10]
```

	0	1	2	3	4	5	6	7	8	9	10	11	12	
0	0.000000	236994.701964	78577.840350	260974.591407	51769.581416	39149.060512	30003.797860	206425.706195	108991.940697	76488.543044	178296.794579	27412.227454	126257.749810	6852
1	236994.701964	0.000000	315439.176808	26550.527773	287970.807817	276141.622437	207115.404780	33742.472390	345963.774390	312810.379793	59059.207242	209746.028237	114631.090826	30446
2	78577.840350	315439.176808	0.000000	339168.030097	31494.808048	39645.760694	108564.128321	284512.007926	31548.758977	17030.194685	256835.625483	105714.634947	203466.010009	1397
3	260974.591407	26550.527773	339168.030097	0.000000	312273.311450	300095.494243	231251.131504	54727.355591	369945.815299	337121.576353	84336.722684	233623.417542	136635.567181	32787
4	51769.581416	287970.807817	31494.808048	312273.311450	0.000000	15176.476831	81054.392885	257851.794851	58617.026426	24868.539744	229053.969743	79086.889672	178017.315363	2861
5	39149.060512	276141.622437	39645.760694	300095.494243	15176.476831	0.000000	69082.893910	245517.452754	69857.763606	38195.246432	217421.871683	66481.219596	164996.199193	3123
6	30003.797860	207115.404780	108564.128321	231251.131504	81054.392885	69082.893910	0.000000	176803.204943	138853.960905	105892.923725	148346.524041	5934.227667	97502.684748	9836
7	206425.706195	33742.472390	284512.007926	54727.355591	257851.794851	245517.452754	176803.204943	0.000000	315357.550518	282695.457394	33626.894430	179050.547952	82005.440362	27316
8	108991.940697	345963.774390	31548.758977	369945.815299	58617.026426	69857.763606	138853.960905	315357.550518	0.000000	34544.425223	287181.187176	136336.054120	234613.645426	4479
9	76488.543044	312810.379793	17030.194685	337121.576353	24868.539744	38195.246432	105892.923725	282695.457394	34544.425223	0.000000	253877.428130	103863.049796	202732.628099	2647

10 rows × 202 columns

Figura 3: Primeras 10 filas del DataFrame obtenido después de aplicar la distancia euclidiana al DataFrame *Hipoteca*.

La métrica de distancia euclidiana puede aplicarse a un conjunto de datos menor. En la siguiente imagen, se muestra la utilización de *iloc* para limitar a qué subconjunto de los datos originales aplicaremos la métrica de distancia euclidiana:

```
DstEuclideana = cdist(Hipoteca.iloc[0:10], Hipoteca.iloc[0:10], metric='euclidean')
MEuclideana = pd.DataFrame(DstEuclideana)
MEuclideana
```

	0	1	2	3	4	5	6	7	8	9
0	0.000000	236994.701964	78577.840350	260974.591407	51769.581416	39149.060512	30003.797860	206425.706195	108991.940697	76488.543044
1	236994.701964	0.000000	315439.176808	26550.527773	287970.807817	276141.622437	207115.404780	33742.472390	345963.774390	312810.379793
2	78577.840350	315439.176808	0.000000	339168.030097	31494.808048	39645.760694	108564.128321	284512.007926	31548.758977	17030.194685
3	260974.591407	26550.527773	339168.030097	0.000000	312273.311450	300095.494243	231251.131504	54727.355591	369945.815299	337121.576353
4	51769.581416	287970.807817	31494.808048	312273.311450	0.000000	15176.476831	81054.392885	257851.794851	58617.026426	24868.539744
5	39149.060512	276141.622437	39645.760694	300095.494243	15176.476831	0.000000	69082.893910	245517.452754	69857.763606	38195.246432
6	30003.797860	207115.404780	108564.128321	231251.131504	81054.392885	69082.893910	0.000000	176803.204943	138853.960905	105892.923725
7	206425.706195	33742.472390	284512.007926	54727.355591	257851.794851	245517.452754	176803.204943	0.000000	315357.550518	282695.457394
8	108991.940697	345963.774390	31548.758977	369945.815299	58617.026426	69857.763606	138853.960905	315357.550518	0.000000	34544.425223
9	76488.543044	312810.379793	17030.194685	337121.576353	24868.539744	38195.246432	105892.923725	282695.457394	34544.425223	0.000000

Figura 4: Aplicación de la métrica de distancia euclidiana a un subconjunto del conjunto original de datos

Posteriormente, importamos *distance* de *scipy.spatial* para calcular la distancia euclidiana entre el primer y segundo registros del DataFrame *Hipoteca*.

```
from scipy.spatial import distance
Objeto1 = Hipoteca.iloc[0]
Objeto2 = Hipoteca.iloc[1]
dstEuclideana = distance.euclidean(Objeto1, Objeto2)
dstEuclideana
```

```
236994.70196398906
```

Figura 5: Cálculo de la distancia euclidiana entre el primer y segundo registros del DataFrame *Hipoteca*

Nos percatamos que este cálculo es igual a lo que veíamos en las primeras dos columnas de los primeros dos registros de la figura 4, que se muestran a continuación:

	0	1
0	0.000000	236994.701964
1	236994.701964	0.000000

Figura 6: Distancia entre los primeros dos registros

Notamos que la diagonal principal es cero porque la distancia entre un punto y ese mismo punto siempre es cero. Más importante aún, notamos que la distancia entre el primer punto y el segundo es igual a la distancia entre el segundo punto y el primero, que a su vez son el mismo valor al que obtuvimos calculando la distancia euclidiana con *distance.euclidean*.

Ahora, calcularemos la distancia entre los puntos del DataFrame *Hipoteca* utilizando la métrica de distancia C*hebyshev*. Para ello, realizaremos un procedimiento similar a lo que hicimos con la distancia euclidiana. Primero utilizaremos *cdist* pasando como parámetro *Hipoteca* dos veces para calcular la distancia de cada uno de los registros del DataFrame con los demás registros y pasando como métrica el valor "chebyshev". De lo anterior, obtenemos:

```
DstChebyshev = cdist(Hipoteca, Hipoteca, metric='chebyshev')
MChebyshev = pd.DataFrame(DstChebyshev)

MChebyshev
```

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
0	0.0	236897.0	78221.0	260933.0	51068.0	39137.0	29812.0	206291.0	108990.0	75902.0	177972.0	27334.0	124971.0	66761.0	241963.0	94878.0	34191.0	193711.0	197411.0
1	236897.0	0.0	315118.0	24036.0	287965.0	276034.0	207085.0	30606.0	345887.0	312799.0	58925.0	209563.0	111926.0	303658.0	5066.0	331775.0	271088.0	43186.0	39486.0
2	78221.0	315118.0	0.0	339154.0	27153.0	39084.0	108033.0	284512.0	30769.0	16852.0	256193.0	105555.0	203192.0	11460.0	320184.0	16657.0	44030.0	271932.0	275632.0
3	260933.0	24036.0	339154.0	0.0	312001.0	300070.0	231121.0	54642.0	369923.0	336835.0	82961.0	233599.0	135962.0	327694.0	18970.0	355811.0	295124.0	67222.0	63522.0
4	51068.0	287965.0	27153.0	312001.0	0.0	11931.0	80880.0	257359.0	57922.0	24834.0	229040.0	78402.0	176039.0	23926.0	293031.0	43810.0	25435.0	244779.0	248479.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
197	39277.0	273777.0	46740.0	297813.0	30789.0	40152.0	66692.0	243171.0	72110.0	39022.0	214852.0	64214.0	161851.0	54715.0	278843.0	57998.0	56224.0	230591.0	234291.0
198	119579.0	356476.0	41358.0	380512.0	68511.0	80442.0	149391.0	325870.0	28623.0	43677.0	297551.0	146913.0	244550.0	52818.0	361542.0	28377.0	85388.0	313290.0	316990.0
199	14435.0	248872.0	66246.0	272908.0	39093.0	27162.0	41787.0	218266.0	97015.0	63927.0	189947.0	39309.0	136946.0	54786.0	253938.0	82903.0	31382.0	205686.0	209386.0
200	30015.0	260005.0	55113.0	284041.0	27960.0	30890.0	52920.0	229399.0	85882.0	52794.0	201080.0	50442.0	148079.0	45453.0	265071.0	71770.0	46962.0	216819.0	220519.0
201	142420.0	379317.0	64199.0	403353.0	91352.0	103283.0	172232.0	348711.0	38523.0	66518.0	320392.0	169754.0	267391.0	75659.0	384383.0	47542.0	108229.0	336131.0	339831.0

202 rows × 202 columns

Figura 7: DataFrame obtenido después de aplicar la distancia Chebyshev al DataFrame *Hipoteca*.

Nuevamente, calcularemos la distancia únicamente para los primeros 10 registros del DataFrame:

```
DstChebyshev = cdist(Hipoteca.iloc[0:10], Hipoteca.iloc[0:10], metric='chebyshev')
MChebyshev = pd.DataFrame(DstChebyshev)
MChebyshev
```

	0	1	2	3	4	5	6	7	8	9
0	0.0	236897.0	78221.0	260933.0	51068.0	39137.0	29812.0	206291.0	108990.0	75902.0
1	236897.0	0.0	315118.0	24036.0	287965.0	276034.0	207085.0	30606.0	345887.0	312799.0
2	78221.0	315118.0	0.0	339154.0	27153.0	39084.0	108033.0	284512.0	30769.0	16852.0
3	260933.0	24036.0	339154.0	0.0	312001.0	300070.0	231121.0	54642.0	369923.0	336835.0
4	51068.0	287965.0	27153.0	312001.0	0.0	11931.0	80880.0	257359.0	57922.0	24834.0
5	39137.0	276034.0	39084.0	300070.0	11931.0	0.0	68949.0	245428.0	69853.0	36765.0
6	29812.0	207085.0	108033.0	231121.0	80880.0	68949.0	0.0	176479.0	138802.0	105714.0
7	206291.0	30606.0	284512.0	54642.0	257359.0	245428.0	176479.0	0.0	315281.0	282193.0
8	108990.0	345887.0	30769.0	369923.0	57922.0	69853.0	138802.0	315281.0	0.0	33088.0
9	75902.0	312799.0	16852.0	336835.0	24834.0	36765.0	105714.0	282193.0	33088.0	0.0

Figura 8: DataFrame obtenido después de aplicar la métrica de distancia Chebyshev a un subconjunto del conjunto original de datos.

Como era de esperarse, cada uno de los elementos de la diagonal principal de la matriz resultante valdrán cero. La métrica chebyshev arrojó un resultado similar a lo que obtuvimos en la distancia euclidiana. Sin embargo, la diferencia entre algunos resultados llega a ser importante, como en la columna 9 de la fila 0, donde Chebyshev muestra una distancia de 75,902 entre estos dos puntos y la distancia euclidiana muestra un valor de 76,488.543.

Obteniendo la distancia entre el primer elemento y el segundo (al igual que como hicimos en la distancia euclidiana), obtenemos 236,897. Lo anterior a diferencia del valor de 236,994 que obtuvimos con la distancia euclidiana.

Para la métrica *Manhattan*, realizamos un procedimiento similar a los dos anteriores para obtener:

```
DstManhattan = cdist(Hipoteca, Hipoteca, metric='cityblock')
MManhattan = pd.DataFrame(DstManhattan)

MManhattan
```

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0	0.0	244759.0	86474.0	267180.0	60166.0	40701.0	34820.0	214460.0	110235.0	87151.0	191067.0	30340.0	143989.0	82848.0	251735.0	96091.0	52061.0	210885.0	215498.0	133517.0
1	244759.0	0.0	330117.0	36279.0	290551.0	284974.0	211391.0	45617.0	354186.0	316302.0	64282.0	220005.0	138098.0	326617.0	7092.0	340486.0	296110.0	66550.0	63355.0	377674.0
2	86474.0	330117.0	0.0	343632.0	43970.0	47121.0	120112.0	284636.0	38493.0	20633.0	276367.0	112964.0	215329.0	20386.0	337027.0	25207.0	55017.0	281047.0	285872.0	48427.0
3	267180.0	36279.0	343632.0	0.0	326816.0	305557.0	239544.0	59000.0	375313.0	351115.0	98823.0	238846.0	151491.0	340280.0	32119.0	362721.0	309539.0	80153.0	75588.0	390915.0
4	60166.0	290551.0	43970.0	326816.0	0.0	22231.0	87564.0	274210.0	67449.0	27261.0	233313.0	90194.0	203477.0	39860.0	295883.0	53395.0	43139.0	270337.0	274720.0	91133.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
197	79103.0	309758.0	91619.0	346023.0	47649.0	45004.0	106529.0	293479.0	115098.0	72986.0	248014.0	107181.0	221538.0	87389.0	314854.0	99726.0	61384.0	289496.0	293675.0	138460.0
198	150173.0	380902.0	79933.0	417009.0	90619.0	111702.0	177657.0	364493.0	41890.0	65914.0	319068.0	178201.0	292982.0	98813.0	385970.0	55458.0	132594.0	360938.0	364819.0	56484.0
199	29640.0	260529.0	91786.0	296786.0	48342.0	45653.0	57422.0	243852.0	115505.0	73547.0	198833.0	57944.0	172611.0	88012.0	265737.0	100463.0	56653.0	240149.0	244584.0	139021.0
200	56348.0	287219.0	96298.0	323494.0	52608.0	50009.0	83992.0	270624.0	119947.0	77993.0	225487.0	84652.0	199147.0	92394.0	292315.0	104763.0	61011.0	266749.0	271148.0	143467.0
201	182937.0	413618.0	112701.0	449329.0	123615.0	144286.0	210313.0	397243.0	74604.0	98540.0	351718.0	210941.0	325602.0	131623.0	418826.0	88552.0	165336.0	393712.0	397469.0	89068.0

202 rows × 202 columns

Figura 9: DataFrame obtenido después de aplicar la distancia *Manhattan* al DataFrame *Hipoteca*.

Aplicando únicamente a las primeras 10 columnas, tenemos:

```
DstManhattan = cdist(Hipoteca.iloc[0:10], Hipoteca.iloc[0:10], metric='cityblock')
MManhattan = pd.DataFrame(DstManhattan)
MManhattan
```

	0	1	2	3	4	5	6	7	8	9
0	0.0	244759.0	86474.0	267180.0	60166.0	40701.0	34820.0	214460.0	110235.0	87151.0
1	244759.0	0.0	330117.0	36279.0	290551.0	284974.0	211391.0	45617.0	354186.0	316302.0
2	86474.0	330117.0	0.0	343632.0	43970.0	47121.0	120112.0	284636.0	38493.0	20633.0
3	267180.0	36279.0	343632.0	0.0	326816.0	305557.0	239544.0	59000.0	375313.0	351115.0
4	60166.0	290551.0	43970.0	326816.0	0.0	22231.0	87564.0	274210.0	67449.0	27261.0
5	40701.0	284974.0	47121.0	305557.0	22231.0	0.0	74941.0	253389.0	71574.0	48998.0
6	34820.0	211391.0	120112.0	239544.0	87564.0	74941.0	0.0	188566.0	143573.0	112221.0
7	214460.0	45617.0	284636.0	59000.0	274210.0	253389.0	188566.0	0.0	323035.0	300521.0
8	110235.0	354186.0	38493.0	375313.0	67449.0	71574.0	143573.0	323035.0	0.0	44100.0
9	87151.0	316302.0	20633.0	351115.0	27261.0	48998.0	112221.0	300521.0	44100.0	0.0

Figura 10: DataFrame obtenido después de aplicar la distancia *Manhattan* a un subconjunto de datos del DataFrame original.

Y, finalmente, obtenemos la distancia entre los dos primeros puntos para obtener que es de 244,759, a diferencia de los valores de 236,994 y 236,897, obtenidos con la distancia euclidiana y distancia Chebyshev, respectivamente.

Por último, realizamos el mismo ejercicio para la métrica de distancia *Minkowski*. Esta métrica es una generalización de las anteriores y, por lo tanto, recibe un parámetro más al momento de utilizar el método *cdist*, el parámetro *p*. Dicho parámetro es el orden para calcular la distancia de tres formas diferentes (1 para distancia Manhattan, 2 para distancia Euclidiana y 3 para distancia Chebyshev). Sin embargo, se pueden utilizar valores intermedios para proporcionar un equilibrio entre distintas medidas. Nosotros utilizaremos *p=1.5* justamente con ese propósito en mente.

```
DstMinkowski = cdist(Hipoteca, Hipoteca, metric='minkowski', p=1.5)
MMinkowski = pd.DataFrame(DstMinkowski)
```

```
MMinkowski
```

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0.000000	237690.995925	79782.466760	261389.573558	53372.216100	39260.690697	30673.683784	207250.873149	109035.213044	78197.161473	179814.568818	27775.257935	129506.805753
1	237690.995925	0.000000	317144.541987	28999.550044	288074.733923	276926.258979	207408.636739	36799.022688	346609.614856	312975.503513	59672.714056	210824.273310	119625.572092
2	79782.466760	317144.541987	0.000000	339376.378955	34836.105302	40977.188827	110319.616169	284512.877279	32977.126225	17574.226078	259461.241924	106538.873153	204840.494513
3	261389.573558	28999.550044	339376.378955	0.000000	313818.956903	300409.654429	232080.294553	55184.304245	370236.872408	338719.479124	87251.942453	233927.059630	138891.097666
4	53372.216100	288074.733923	34836.105302	313818.956903	0.000000	17046.898384	81840.258784	260012.361790	60284.016224	25100.249754	229261.198445	80976.179684	182840.887336
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
197	60770.233816	281405.644842	70353.660441	309154.836773	37318.860168	40811.991276	83771.074585	256837.573004	90959.177099	55489.213622	222082.525626	84726.123266	183955.430019
198	128687.635109	360119.702102	61510.907561	387055.019657	75607.413634	91839.027927	156357.233170	333809.911618	33189.501979	52021.246348	300612.255283	155864.097331	257818.768980
199	21714.620373	250061.119850	74860.624904	276559.126021	41172.682830	34857.378658	46295.299016	223131.869044	101254.942780	65519.114930	190734.530982	46282.188072	147501.667931
200	42815.775409	264889.398939	74602.554581	292321.617039	39959.337646	38607.596589	65536.701429	239632.877207	97975.555632	61712.729444	205463.256801	66359.213366	165977.464267
201	155395.390030	385435.511309	87986.061870	412690.548292	102457.030136	118784.257654	182746.119425	359717.588847	57493.076692	78915.637485	325932.271432	182455.001493	284307.155460

202 rows × 202 columns

Figura 11: DataFrame obtenido después de aplicar la distancia Minkowski al DataFrame *Hipoteca*.

Obtenemos también la distancia para únicamente los primeros 10 registros:

```
DstMinkowski = cdist(Hipoteca.iloc[0:10], Hipoteca.iloc[0:10], metric='minkowski', p=1.5)
MMinkowski = pd.DataFrame(DstMinkowski)
MMinkowski
```

	0	1	2	3	4	5	6	7	8	9
0	0.000000	237690.995925	79782.466760	261389.573558	53372.216100	39260.690697	30673.683784	207250.873149	109035.213044	78197.161473
1	237690.995925	0.000000	317144.541987	28999.550044	288074.733923	276926.258979	207408.636739	36799.022688	346609.614856	312975.503513
2	79782.466760	317144.541987	0.000000	339376.378955	34836.105302	40977.188827	110319.616169	284512.877279	32977.126225	17574.226078
3	261389.573558	28999.550044	339376.378955	0.000000	313818.956903	300409.654429	232080.294553	55184.304245	370236.872408	338719.479124
4	53372.216100	288074.733923	34836.105302	313818.956903	0.000000	17046.898384	81840.258784	260012.361790	60284.016224	25100.249754
5	39260.690697	276926.258979	40977.188827	300409.654429	17046.898384	0.000000	69749.100955	246187.491676	69936.944305	40487.806354
6	30673.683784	207408.636739	110319.616169	232080.294553	81840.258784	69749.100955	0.000000	178267.963453	139247.210167	106708.022220
7	207250.873149	36799.022688	284512.877279	55184.304245	260012.361790	246187.491676	178267.963453	0.000000	315980.319533	284964.264428
8	109035.213044	346609.614856	32977.126225	370236.872408	60284.016224	69936.944305	139247.210167	315980.319533	0.000000	36693.205417
9	78197.161473	312975.503513	17574.226078	338719.479124	25100.249754	40487.806354	106708.022220	284964.264428	36693.205417	0.000000

Figura 12: DataFrame obtenido después de aplicar la distancia *Minkowski* a un subconjunto de datos del DataFrame original.

y el valor de la distancia entre los primeros dos registros utilizando *distance.minkowski*:

```
Objeto1 = Hipoteca.iloc[0]
Objeto2 = Hipoteca.iloc[1]
dstMinkowski = distance.minkowski(Objeto1, Objeto2)
dstMinkowski
```

```
236994.70196398906
```

Figura 13: Distancia entre los dos primeros registros utilizando *distance.minkowski*.

Comparando los resultados de distancia entre el primer y segundo registros utilizando cada una de las métricas de distancia:

Métrica	Euclidiana	Chebyshev	Manhattan	Minkowski
Distancia	**236,994.7019**	**236,897**	**244,759**	**236,994.7019**

Tabla 1: Comparativa entre las métricas de distancia utilizadas.

notando las diferencias que existen entre cada una de las métricas, que puede ser considerable si comparamos la métrica Manhattan con cualquiera de las otras, por ejemplo. No obstante, cada una de estas métricas tiene un sentido y el saber cuál utilizar dependerá de la situación en la que se esté trabajando.

Conclusiones

Como era de esperarse, cada una de las métricas arrojó valores diferentes de distancia al medir entre datos de un mismo DataFrame. Todas las métricas utilizadas tienen en común el que la diagonal principal del DataFrame resultante se componga de ceros (ya que la distancia entre un punto y ese mismo punto es cero) y el ser simétricas (la distancia de un punto A a un punto B es igual que la distancia de B a A).

A partir de lo realizado en la práctica, me percaté de la importancia de estas métricas para ser utilizadas como parte fundamental de algoritmos de aprendizaje automático como *K-means Clustering* o *K-Nearest-Neighbours* donde la distancia juega un papel fundamental para conocer qué tan relacionado se encuentra un elemento con otro, permitiéndole al algoritmo reconocer similitudes entre contenidos. Un ejemplo sencillo de lo anterior se muestra en el siguiente código, el cual permite hacer procesamiento de lenguaje natural utilizando la métrica de distancia *cosine_similarity*, para filtrar los resultados de una búsqueda y obtener los resultados más significativos:

```
vectorizer = TfidfVectorizer()

corpus = [
 'the brown fox jumped over the brown dog',
 'the quick brown fox',
 'the brown brown dog',
 'the fox ate the dog'
]

query = ["brown"]

X = vectorizer.fit_transform(corpus)
Y = vectorizer.transform(query)
```

```
cosine_similarity(Y, X.toarray())

Results:
array([[0.54267123, 0.44181486, 0.84003859, 0.        ]])
```

Figura 14: Uso de una métrica de distancia para procesamiento de lenguaje. (Fuente: Sharma, N. *Importance of Distance Metrics in Machine Learning Modelling*. Obtenido el 4/10/21 de https://towardsdatascience.com/importance-of-distance-metrics-in-machine-learning-modelling-e51395ffe60d)

viendo que el tercer valor del vector *corpus* es el que tiene una mayor similitud a lo que se buscó.

A lo largo de la práctica me percaté de las diferencias entre cada una de las métricas de distancia, las cuales pueden ser considerables. No obstante, cada una de estas métricas tiene un sentido y el saber cuál utilizar dependerá de la situación en la que se esté trabajando.

Algunas aplicaciones de algoritmos que utilicen estas métricas de distancia pueden ser: reconocimiento facial, sistemas de recomendación, clasificación de imágenes, entre otros. Una buena métrica de distancia ayudará a mejorar el rendimiento de estas aplicaciones.

En lo personal, creo que es interesante que Python ya cuente con todas estas métricas para poderlas utilizar rápidamente. Sin duda esto, como tantas otras bibliotecas y funciones que tiene Python, es uno de los factores por los que es considerado el lenguaje para aplicaciones de análisis de datos o de Machine Learning.

En general, puedo concluir que se cumplieron los objetivos de esta práctica y que, mediante la implementación de distintas métricas de distancia, logré entender de mejor manera su utilización en aplicaciones de inteligencia artificial.