

Últimos salarios en el sector de la ciencia de datos 2020-2025

Yerson Danilo Chilito Inga

28. May 2025

1. Introducción

El análisis de salarios en el campo de la ciencia de datos es fundamental para comprender las dinámicas laborales y las oportunidades de crecimiento profesional. Este proyecto utiliza un enfoque innovador basado en teoría de grafos para explorar la relación entre los tipos de trabajo y sus categorías salariales en el período 2020-2025. Mediante técnicas de agrupamiento y transformación logarítmica, se procesaron 28,375 registros para reducir la asimetría en la distribución de salarios y facilitar su análisis estructural.

El objetivo principal es modelar esta relación como un grafo bipartito (trabajos y rangos salariales) y luego proyectarlo en un grafo de trabajos, donde las conexiones indican que dos roles comparten al menos una categoría salarial. A través de medidas de centralidad (grado, vector propio, cercanía e intermediación), se identifican patrones clave, como la diversidad salarial de los puestos y los roles que actúan como «puentes» entre distintos niveles de remuneración. Adicionalmente, se analizan propiedades como la homofilia y la detección de comunidades para entender agrupaciones naturales en la red. Este enfoque no solo revela desigualdades y oportunidades en el mercado laboral, sino que también proporciona herramientas para que profesionales y empleadores tomen decisiones informadas.

2. Conjunto de datos

Para la realización de este proyecto, se decidió utilizar un conjunto de datos relacionado con los salarios de diversos trabajos en el campo de la ciencia de datos. El dataset fue obtenido de Kaggle. A modo de ejemplo, se presentan algunos de los datos contenidos en este conjunto en la Tabla Principal 1. Dado que un análisis exhaustivo de todos los registros requeriría un estudio más detallado lo cual va más allá del objetivo de este trabajo, nos centraremos exclusivamente en la relación entre el tipo de trabajo y el salario correspondiente. Para ello, se llevará a cabo una «limpieza» de los datos utilizando Python.

```
1 import pandas as pd
2 import numpy as np
3 df = pd.read_csv('salaries_2020-2025.csv')
4 columns_delete = ['work_year', 'experience_level', 'employment_type', 'salary',
5 'salary_currency', 'employee_residence', 'remote_ratio', 'company_location',
6 'company_size']
7 df = df.drop(columns=columns_delete)
8 df = df.drop_duplicates()
9 df.to_csv('salaries_2020-2025(clean).csv', index = False)
```



Así la Tabla Principal 1 nos quedaría de la siguiente manera.

Tabla 1: Ejemplo del conjunto de datos limpio.

Job title	Salary
Research Scientist	147000
Machine Learning	160000
Data Analyst	97600
Data Engineer	200000
Business Intelligence	127000
Applied Scientist	249300
Data Engineer	65141
Data Scientist	48000
BI Data Analyst	55000
Data Scientist	250000
Data Engineer	70139
Data Engineer	33511

Este nuevo conjunto de datos consta de 28,375 registros, que incluyen 317 trabajos diferentes y 9,535 salarios distintos. Dado que la cantidad de salarios es considerablemente alta, es esencial agruparlos. Para ello, observemos como están distribuidos los salarios.

```

1 import seaborn as sns
2 import matplotlib.pyplot as plt
3
4 sns.histplot(df['salary_in_usd'], kde=True)
5 plt.xlabel('Salarios')
6 plt.ylabel('Frecuencia')
7 plt.show()

```

 Python

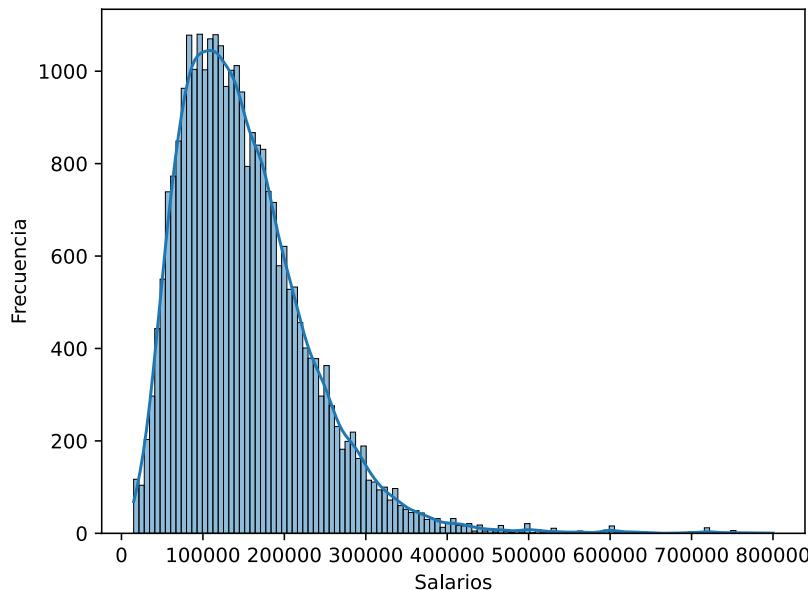


Figura 1: Distribución de salarios.

Como se puede observar, la distribución de los salarios presenta un sesgo positivo (asimetría hacia la derecha). Para lograr un agrupamiento más representativo y equilibrado, es necesario reducir dicha asimetría. Con este objetivo, se aplicará una transformación logarítmica a los salarios antes de realizar el proceso de agrupamiento. A continuación, se muestra cómo se distribuyen los salarios tras esta transformación.

```
1 df['log_salary'] = np.log10(df['salary_in_usd'])
2 sns.histplot(df['log_salary'], kde=True)
3 plt.xlabel('Salarios en escala logarítmica')
4 plt.ylabel('Frecuencia')
5 plt.show()
```

Python

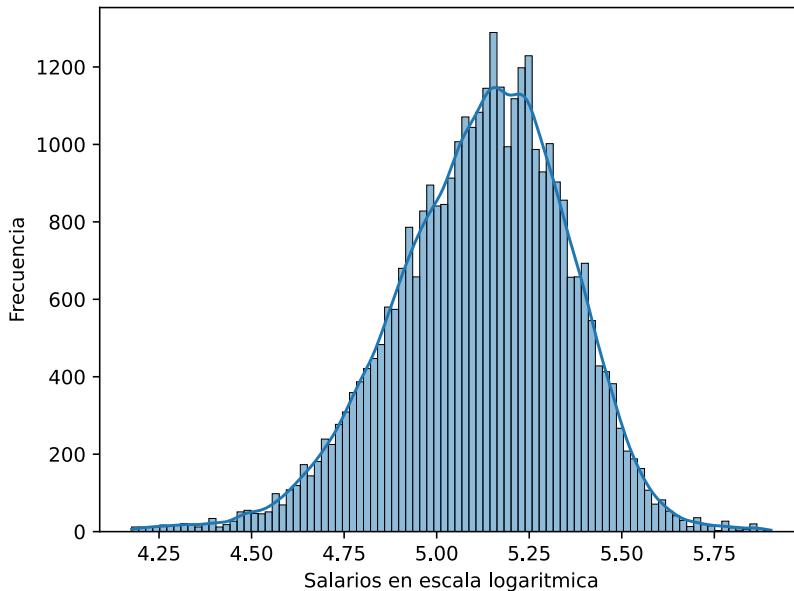


Figura 2: Distribución de salarios después de realizar una transformación logarítmica.

Una vez realizada la transformación logarítmica, se procederá a agrupar los salarios y a asignar cada empleo a su correspondiente grupo salarial.

```
1 k = 128
2
3 bins = np.linspace(np.min(log_salary), np.max(log_salary), k+1)
4 real_bins = 10 ** bins
5 labels = [f'{np.round(real_bins[i], 2)} to {np.round(real_bins[i+1], 2)}' for i in
6 range(k)]
7
8 df['salary'] = pd.cut(log_salary, bins=bins, labels=labels, include_lowest=True)
9
10 df = df.drop('salary_in_USD', axis=1)
11 df = df.drop_duplicates()
12 df.to_csv('salaries_2020-2025(grouped).csv', index = False)
```

Python

Inicialmente se consideró aplicar la regla de Freedman-Diaconis para determinar el número de intervalos salariales (bins). No obstante, dicha regla sugería la creación de 91 bins, lo cual habría resultado en una

agrupación excesivamente «gruesa» de los salarios. Esto, a su vez, habría generado una proyección del grafo de trabajos sobre salarios demasiado densa, dificultando su posterior análisis estructural. Por esta razón, se optó por aumentar el número de bins a 128. A continuación, se presenta un ejemplo ilustrativo de cómo quedan distribuidos los datos bajo esta configuración.

Tabla 2: Ejemplo del conjunto de datos con los salarios agrupados.

Job title	Salary
Research Scientist	144883.93 to 149455.67
Machine Learning	159036.48 to 164054.8
Data Analyst	96743.53 to 99796.22
Data Engineer	197670.56 to 203907.96
Business Intelligence	124039.31 to 127953.31
Applied Scientist	245689.86 to 253442.48
Data Engineer	64598.67 to 66637.05
Data Scientist	47348.02 to 48842.06
BI Data Analyst	53613.05 to 55304.79
Data Scientist	245689.86 to 253442.48
Data Engineer	68739.75 to 70908.8
Data Engineer	32613.37 to 33642.47

Con esta modificación, obtenemos un total de 6,609 registros, distribuidos en 128 tipos de salarios distintos y la misma cantidad de trabajos diferentes (317). A partir de ahora, trabajaremos con esta última variación.

Work year	Experience level	Employment type	Job title	Salary currency	Salary in USD	Employee residence	Remote ratio	Company location	Company size
2025	MI	FT	Research Scientist	USD	147000	US	0	US	M
2025	SE	FT	Machine Learning Engineer	USD	160000	US	0	US	M
2024	SE	FT	Data Analyst	USD	97600	CA	100	CA	M
2024	MI	CT	Data Engineer	USD	200000	US	0	US	M
2023	SE	FT	Business Intelligence	USD	127000	CA	0	CA	M
2023	MI	FT	Applied Scientist	USD	249300	US	0	US	L
2022	MI	FT	Data Engineer	EUR	65141	FR	100	FR	M
2022	MI	FT	Data Scientist	USD	48000	RU	100	US	S
2021	EN	FT	BI Data Analyst	USD	55000	US	50	US	S
2021	EX	FT	Data Scientist	USD	250000	US	0	US	L
2020	MI	FT	Data Engineer	EUR	70139	FR	50	FR	L
2020	SE	FT	Data Engineer	MXN	33511	MX	0	MX	S

Tabla Principal 1: Algunos registros del conjunto de datos a analizar.

3. Análisis

Dado que existe una relación entre trabajos y salarios, al considerar estos dos elementos como nodos, obtenemos un grafo bipartito. Para fines ilustrativos, mostraremos este grafo utilizando Wolfram Mathematica.

```
1 groupedData = Import["salaries_2020-2025(grouped).csv", "CSV"];
2 groupedData = Rest[groupedData];
3 graph = Graph[UndirectedEdge @@@ groupedData]
```

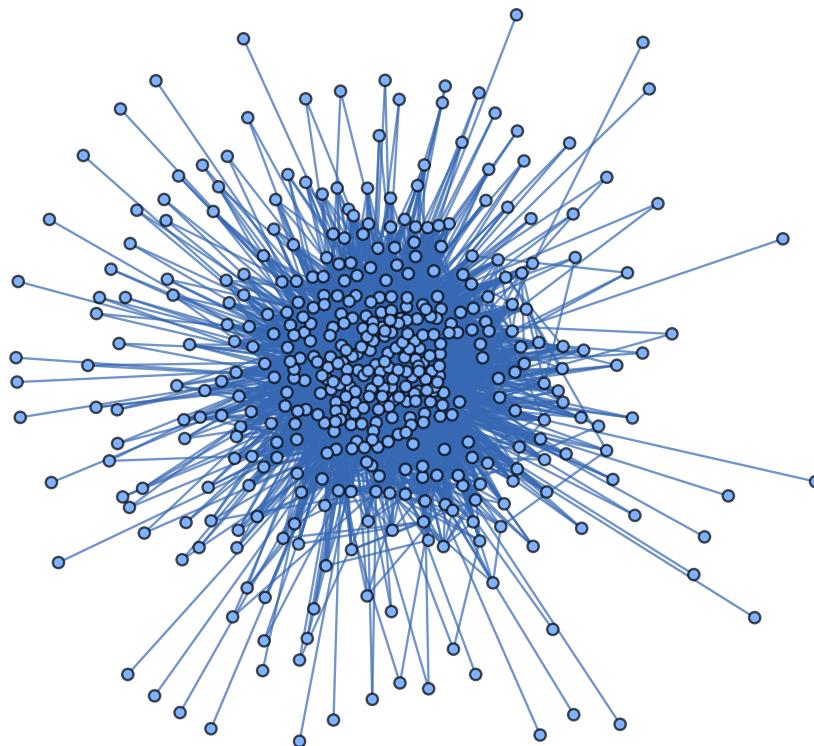


Figura 3: Grafo del conjunto de datos.

Ahora para realizar un análisis a este dataset se procedera a encontrar los conjuntos particionados (ver Código 1).

```
1 partitionedSets = partition[graph];
2 RandomSample[partitionedSets[[1]], 2]
3 RandomSample[partitionedSets[[2]], 2]
4 Length[partitionedSets[[1]]]
5 Length[partitionedSets[[2]]]
```

```
{335203.86 to 345781.05, 45899.68 to 47348.02}
{Quantitative Analyst, Postdoctoral Researcher}
128
317
```

Como era de esperarse, obtenemos los conjuntos particionados correspondientes a la clase de salario y a los trabajos, junto con sus respectivas longitudes. A continuación, procederemos a encontrar el grafo correspondiente a la proyección del conjunto de trabajos sobre el conjunto de salarios (ver Código 2).

```
1 graphV = projection[graph, V];
2 GraphPlot[graphV]
```

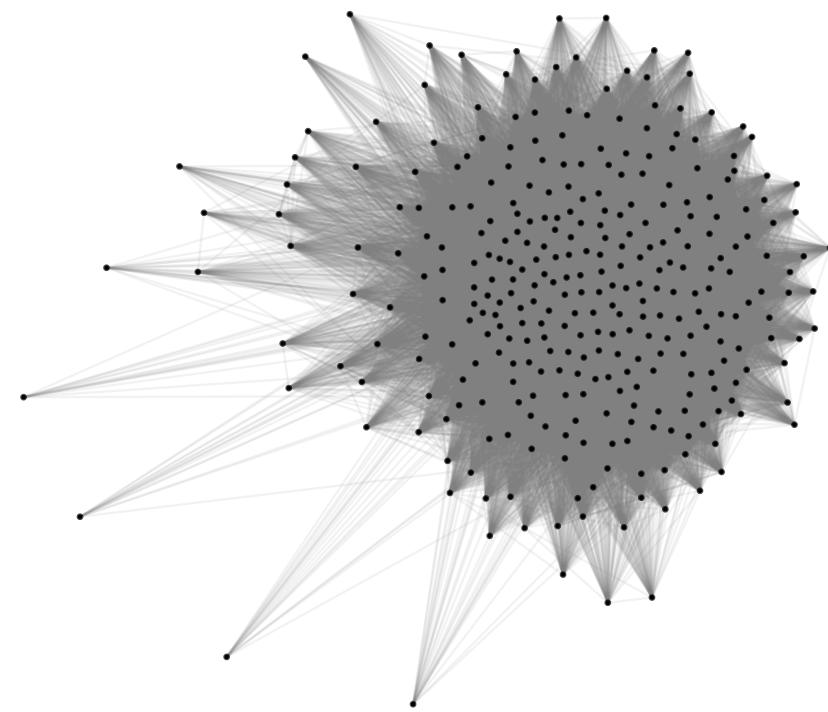


Figura 4: Grafo correspondiente a la proyección del conjunto de trabajos sobre el conjunto de salarios.

A partir del grafo proyectado, se puede deducir que la conexión entre dos nodos indica que ambos trabajos comparten al menos una categoría salarial. Por ejemplo, en la Tabla 2, se observa que *Applied Scientist* y *Data Scientist* pertenecen a una misma categoría salarial, motivo por el cual están conectados en la red. Asimismo, se evidencia una elevada densidad de aristas, lo que sugiere un alto nivel de conectividad entre los nodos. Para respaldar esta observación, a continuación se analizará el histograma de grados del grafo proyectado, junto con su correspondiente diagrama de caja.

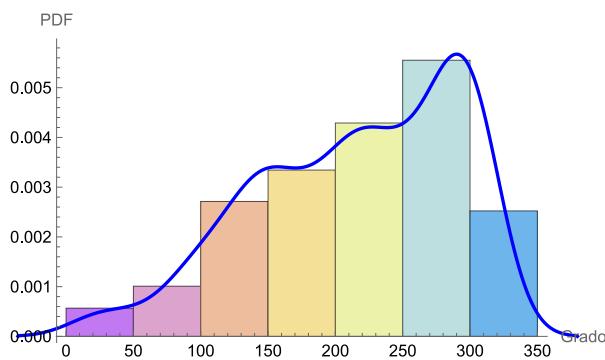


Figura 5: Histograma con estimación de densidad.

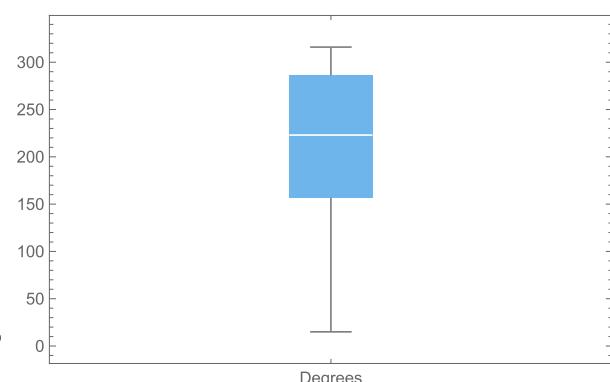


Figura 6: BoxPlot de los grados de los nodos.

Efectivamente, se puede observar que hay una gran cantidad de nodos con grados altos y solo unos pocos con grados bajos. Esto sugiere que la mayoría de los trabajos comparten tipos de salarios similares. Sin embargo, es importante destacar que, incluso dentro de un mismo puesto, puede existir una amplia diversidad salarial. Esta variabilidad puede atribuirse a factores como el tamaño de la empresa, la ubicación geográfica o la experiencia del empleado. Tales diferencias podrían explicar la alta densidad del grafo proyectado mostrado en la Figura 4 (68.32%).

Por otro lado, los nodos con grados bajos podrían representar empleos con menor variabilidad salarial, lo que los convierte en casos especialmente relevantes para un análisis posterior. A continuación, exploraremos algunas estadísticas adicionales.

Tabla 3: Estadísticas descriptivas de la distribución de grados.

Mínimo	15
Máximo	316
Media	215.89
Mediana	223
Moda	294
Desviación estándar	74.88
Q1	157
Q3	286

Lo primero que se observa es que la media, la mediana y la moda reflejan claramente el sesgo negativo hacia la izquierda en el histograma mostrado en la Figura 5. Al comparar estas medidas con el grado mínimo y los cuartiles, se evidencia que hay muy pocos nodos con grados bajos, lo que contribuye a la densidad del grafo. Además, la desviación estándar es considerablemente alta en relación con el coeficiente de variación (véase la Ecuación 1), lo que indica una alta variabilidad en los grados de los nodos.

$$CV = \frac{\sigma}{\text{Media}} = \frac{74.88}{215.89} \simeq 0.347 \quad (1)$$

3.1. Medidas de centralidad

Las medidas de centralidad permiten evaluar la importancia relativa de cada nodo en una red, ayudando a identificar aquellos que ocupan posiciones estratégicas dentro de su estructura. A continuación, se presentan algunas de las principales medidas de centralidad aplicadas a la red haciendo uso de *Wolfram Mathematica*.

3.1.1. Degree Centrality

En el caso de nuestra red, donde los nodos representan empleos en el ámbito de la ciencia de datos, la centralidad por grado se interpreta como un indicador del nivel de diversidad salarial de cada puesto en comparación con otros dentro de la misma rama profesional. Procedamos a ver ahora algunos de los trabajos con mayor centralidad por grado.

Tabla 4: 20 primeros puesto de la centralidad por grado.

Nodo	Centralidad	Nodo	Centralidad
Software Engineer	316	Research Scientist	312
Data Scientist	316	Research Engineer	312
Machine Learning Engineer	316	Analytics Engineer	312
Data Engineer	316	Data Specialist	311
Engineer	315	Developer	310
Manager	315	Consultant	310
Data Analyst	315	Business Intelligence	310
Associate	314	Research Analyst	310
Analyst	313	Software Developer	310
Data Architect	312	Data Manager	309

En la tabla anterior se identifican trabajos con valores de centralidad notablemente altos. Esta característica se explica por el hecho de que algunos empleos están asociados a una amplia variedad de categorías salariales, lo que genera múltiples conexiones con otros trabajos dentro del mismo rango. Dicha diversidad salarial puede deberse a características intrínsecas del empleo o al hecho de que se trata de ocupaciones ampliamente representadas en el conjunto de datos. Los 20 trabajos con mayor centralidad se distinguen precisamente por esta amplitud en la distribución salarial, lo que sugiere que su remuneración puede variar considerablemente según factores como la experiencia, la ubicación geográfica o el sector económico. A continuación, se procederá a analizar los trabajos con menor diversidad salarial, es decir, aquellos con menos conexiones en la red o que se encuentran en regiones alejadas.

Tabla 5: 20 últimos puesto de la centralidad por grado.

Nodo	Centralidad	Nodo	Centralidad
AI Software Development Engineer	15	AWS Data Architect	71
Analytics Analyst	15	Applied AI ML Lead	76
Analytics Engineering Manager	18	Marketing Data Engineer	76
Principal Data Architect	22	Applied Research Scientist	77
Data Science Tech Lead	27	Research Data Manager	86
People Data Analyst	35	Sales Data Analyst	86
Platform Data Engineer	45	Customer Success Manager	86
CRM Data Analyst	45	BI Data Engineer	86
Quantitative Research Analyst	49	Cloud Data Architect	87
Clinical Data Operator	52	Data Management Coordinator	87

A diferencia de lo observado en la Tabla 4, en esta tabla se presentan los trabajos con menor diversidad salarial según los datos disponibles. Estos empleos corresponden a los nodos con menor centralidad por grado. La baja diversidad salarial puede deberse a varios factores, como que se trate de ocupaciones estandarizadas (donde la experiencia u otras variables no tienen tanto peso), a una saturación del mercado laboral en ese sector, a una baja demanda o, simplemente, a una falta de datos en nuestro conjunto. Sea cual sea la causa, es importante prestar atención a estos trabajos, ya que suelen presentar salarios fijos que no varían significativamente con aspectos clave como la antigüedad, la experiencia u otras características relevantes.

3.1.2. Eigenvector centrality

La centralidad de vector propio permite determinar el nivel de importancia de los nodos en función de a quiénes están conectados. En este contexto, esta medida nos ofrece un indicio de qué ocupaciones están vinculadas con otras que presentan una mayor diversidad salarial. A continuación, se presenta la siguiente tabla:

Tabla 6: 20 primeros puesto de la centralidad por vectores propios.

Nodo	Centralidad	Nodo	Centralidad
Software Engineer	0.004183	Analytics Engineer	0.004177
Data Scientist	0.004183	Research Engineer	0.004175
Machine Learning Engineer	0.004183	Business Intelligence	0.004174
Data Engineer	0.004183	Developer	0.004174
Data Analyst	0.004182	Software Developer	0.004174
Engineer	0.004182	Data Architect	0.004174
Manager	0.004182	Research Analyst	0.004172
Associate	0.004180	Cloud Engineer	0.004171
Analyst	0.004179	Data Specialist	0.004169
Research Scientist	0.004178	Consultant	0.004164

Al comparar estos resultados con los obtenidos mediante la centralidad por grado (Tabla 4), se observa que no hay variaciones significativas en los primeros puestos. Esto era esperable, dado que existen nodos con grados muy altos que se conectan con una gran cantidad de otros trabajos, manteniendo así su posición destacada en ambas medidas. No obstante, la centralidad de vectores propios introduce una dimensión adicional de análisis: una ocupación con baja diversidad salarial puede adquirir relevancia si está conectada con otras de mayor diversidad, lo que podría facilitar una transición hacia empleos con mayores oportunidades salariales. En este sentido, esta medida permite identificar trayectorias potenciales de movilidad dentro de la red ocupacional. Un ejemplo de ello es *Cloud Engineer*, que no figura entre los 20 primeros puestos según la centralidad por grado, pero sí aparece en estas posiciones cuando se analiza la centralidad por vectores propios. Esto sugiere que su relevancia se incrementa al estar conectado con empleos más diversos salarialmente.

3.1.3. Closeness centrality

La centralidad por cercanía mide que tan cerca en términos de pasos está un nodo con respecto a los demás nodos de la red. En la siguiente tabla se muestran los trabajos con mayor centralidad por cercanía.

Tabla 7: 20 primeros puesto de la centralidad por cercanía.

Nodo	Centralidad	Nodo	Centralidad
Software Engineer	1	Research Scientist	0.9875
Data Scientist	1	Research Engineer	0.9875
Machine Learning Engineer	1	Analytics Engineer	0.9875
Data Engineer	1	Data Specialist	0.9844
Engineer	0.9968	Developer	0.9813
Manager	0.9968	Consultant	0.9813
Data Analyst	0.9968	Business Intelligence	0.9813
Associate	0.9937	Research Analyst	0.9813
Analyst	0.9905	Software Developer	0.9813
Data Architect	0.9875	Data Manager	0.9783

Una vez más, los primeros lugares se mantienen sin variaciones notables, lo que concuerda con el comportamiento ya descrito en el análisis de la centralidad de vectores propios. En esta medida, una alta centralidad por cercanía (closeness) significa que un trabajo está estructuralmente cerca de muchos otros trabajos en la red, es decir, que puede alcanzarlos con pocos pasos o conexiones. Sin embargo,

es importante destacar que esta cercanía es estructural, más no necesariamente salarial. Dos empleos pueden estar cerca en la red (en términos de número de conexiones), pero muy alejados en cuanto a los valores salariales reales. Esto ocurre, por ejemplo, cuando existe un nodo “puente” que conecta trabajos con salarios muy distintos. Por tanto, una centralidad alta por cercanía no implica directamente que el salario del trabajo sea representativo o intermedio, sino que ocupa una posición estructural que le permite acceder rápidamente al resto de la red.

3.1.4. Betweenness centrality

La centralidad de intermediación mide la relevancia de un nodo según la cantidad de veces que aparece en los caminos más cortos entre pares de nodos del grafo. En nuestra red, esta medida refleja la capacidad de un trabajo para servir como intermediario entre otros empleos, lo que implica también la posibilidad de conectar diferentes niveles salariales.

Tabla 8: 20 primeros puesto de la centralidad de intermediación.

Nodo	Centralidad	Nodo	Centralidad
Software Engineer	247.4694	Consultant	190.7283
Data Scientist	247.4694	Research Engineer	188.5090
Machine Learning Engineer	247.4694	Data Specialist	187.7867
Data Engineer	247.4694	Analytics Engineer	183.8052
Engineer	232.3255	AI Engineer	180.5593
Manager	232.3255	Research Scientist	171.5328
Data Analyst	229.8198	Product Manager	168.1288
Associate	213.6264	Data Manager	163.0069
Data Architect	201.8114	Business Analyst	161.1501
Analyst	195.7660	Research Analyst	160.6246

Aquí una alta centralidad de intermediación (betweenness centrality) sugiere que un trabajo actúa como puente entre diferentes categorías salariales. En nuestro caso, en el ámbito de la ciencia de datos, es posible transitar entre clases de empleos con distintos niveles de remuneración mediante la adquisición de ciertos conocimientos adicionales o ajustes en la rutina laboral. Esta propiedad es particularmente relevante, ya que estos “trabajos puente” pueden representar una vía para pasar de un salario anual de, por ejemplo, 15.000 USD a uno de 800.000 USD, que corresponden al salario mínimo y máximo, respectivamente, dentro del conjunto de datos analizado.

A continuación, se presenta una visualización de los nodos más relevantes de nuestra red según las distintas medidas de centralidad previamente mencionadas, donde el tamaño de cada nodo representa su nivel de importancia conforme a la medida correspondiente.

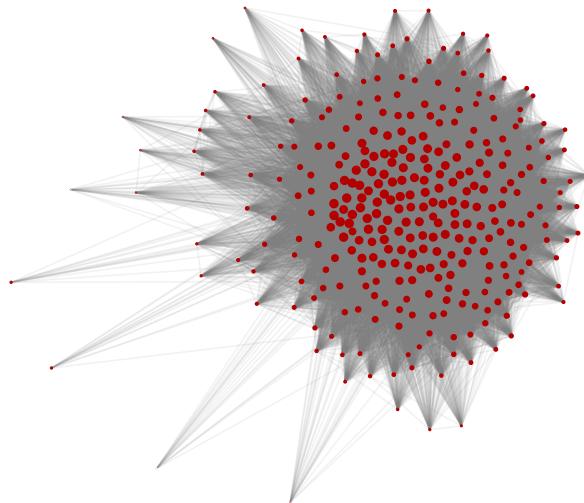


Figura 7: Importancia relativa de los trabajos según la centralidad por grados.

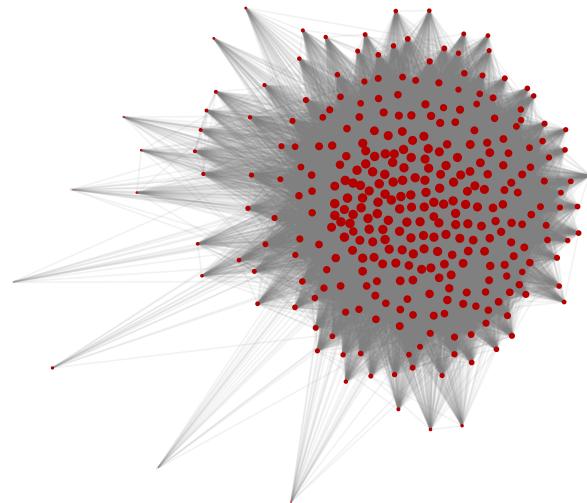


Figura 8: Importancia relativa de los trabajos según la centralidad de vector propio.

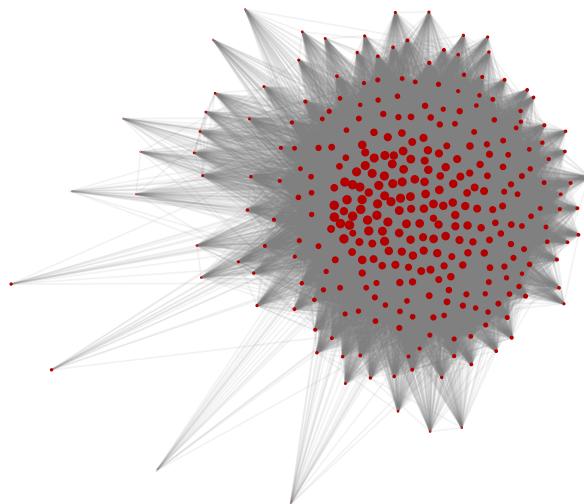


Figura 9: Importancia relativa de los trabajos según la centralidad por cercanía.

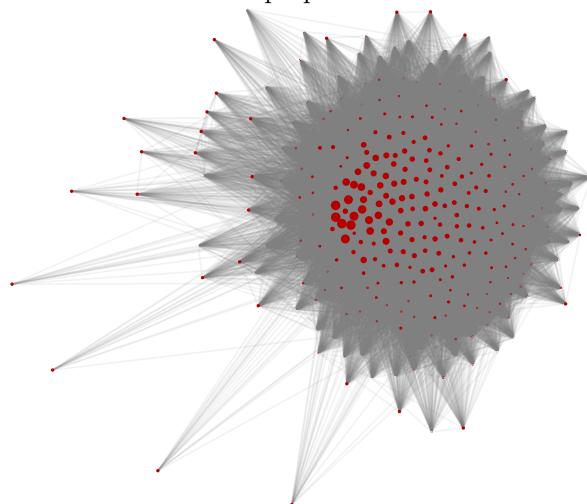


Figura 10: Importancia relativa de los trabajos según la centralidad de intermedación.

Como se mencionó anteriormente, las medidas de centralidad por grado, vectores propios y cercanía no presentan variaciones significativas entre sí. En contraste, la centralidad de intermedación revela cambios importantes en la importancia relativa de los nodos. Esta medida resulta especialmente relevante en el contexto del análisis, ya que permite identificar trabajos que actúan como puente entre distintas categorías salariales y regiones estructurales de la red.

3.2. Homophily and assortative mixing

La homofilia es una propiedad característica de muchas redes, que describe la tendencia de los nodos a conectarse con otros del mismo tipo. Esta afinidad estructural puede cuantificarse mediante el coeficiente de mezcla asortativa (assortative mixing), el cual mide si los nodos con cierto atributo tienden a relacionarse con otros que comparten dicho atributo.

En nuestro caso, evaluamos la homofilia considerando el grado de los nodos, es decir, analizamos si los trabajos tienden a conectarse con otros que presentan una diversidad salarial similar. Para ello,

utilizamos las funciones de Wolfram Mathematica, obteniendo un coeficiente de mezcla asortativa de aproximadamente -0.295 .

Este valor negativo indica una tendencia disasortativa, lo que significa que los trabajos con alta diversidad salarial tienden a conectarse con otros de menor diversidad, y viceversa. En otras palabras, los nodos no se agrupan con otros de su mismo grado, sino que establecen conexiones con aquellos que difieren en su nivel de diversidad salarial.

Este resultado es especialmente interesante, ya que verifica la existencia de posibles puentes entre empleos con distintas condiciones salariales, como se mencionó en el apartado de la centralidad de intermediación. Esto abre la posibilidad de movilidad laboral dentro de la red: por ejemplo, trabajadores en empleos de baja remuneración podrían estar estructuralmente conectados con ocupaciones mejor pagadas, lo que sugiere trayectorias de transición entre distintos niveles salariales.

3.3. Comunidades

En muchas redes del mundo real es común la presencia de comunidades, es decir, grupos de nodos que presentan una mayor densidad de conexiones entre sí que con el resto de la red. En nuestro contexto, la existencia de comunidades sugiere que ciertos conjuntos de trabajos comparten más categorías salariales entre ellos que con otros trabajos fuera del grupo. Existen diversos métodos para la detección de comunidades; sin embargo, Wolfram Mathematica emplea por defecto un enfoque basado en el principio de modularidad, que busca particionar la red maximizando la diferencia entre la densidad de conexiones dentro de los grupos y la esperada en una red aleatoria equivalente. A continuación, se presentan de forma visual las comunidades identificadas en nuestra red mediante este enfoque.

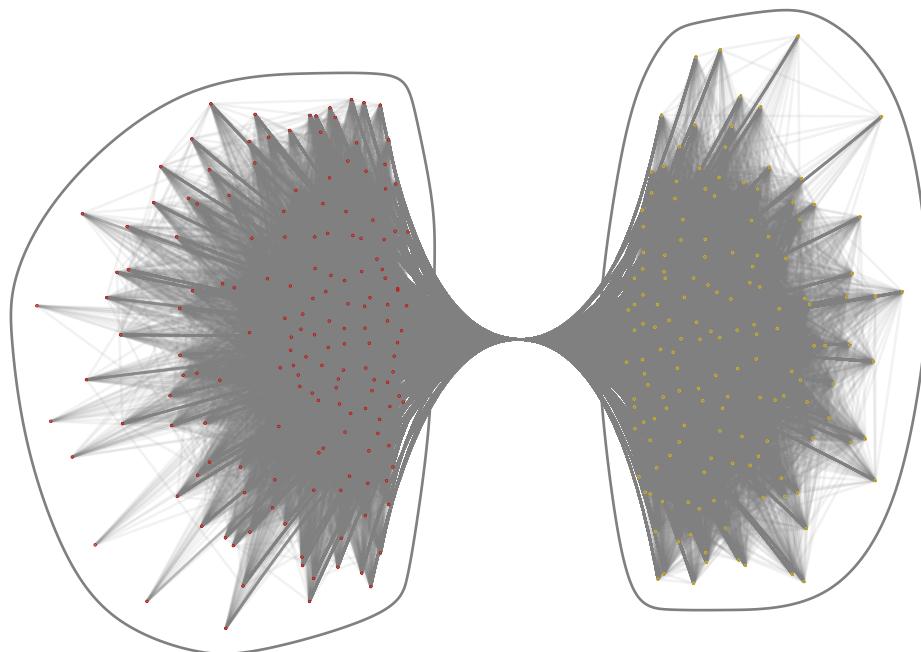


Figura 11: Comunidades detectadas en la red de trabajos mediante el criterio de modularidad.

Se identificaron dos comunidades principales en la red, como se visualiza en la figura Figura 11. Tal como se explicó anteriormente, esto indica que existen grupos de trabajos que tienden a compartir categorías salariales similares, lo que da lugar a una división natural en dos agrupaciones.

La pertenencia de ciertos trabajos a una misma comunidad puede explicarse por diversos factores. Por ejemplo, dos empleos pueden agruparse si se desarrollan en la misma región geográfica, ya que esto influye

directamente en el nivel salarial. Asimismo, algunos trabajos presentan rangos salariales más acotados, lo cual limita su conexión con empleos que poseen mayor variabilidad.

Otra posible causa tiene que ver con las formas de remuneración: podría existir una tendencia en uno de los grupos hacia trabajos mejor pagos, ya sea por la alta demanda, la moda del sector o la capacidad de las empresas para ofrecer mejores condiciones.

Finalmente, un factor clave es el nivel de experiencia requerido. Es razonable suponer que empleos con niveles similares de experiencia (Nivel de entrada, intermedio, senior o ejecutivo) tenderán a compartir categorías salariales similares, lo que favorecería su agrupación en una misma comunidad. Por ejemplo, es posible que los trabajos del grupo uno correspondan mayoritariamente a posiciones de entrada o intermedias, mientras que los del grupo dos estén asociados con roles que exigen mayor experiencia, como perfiles senior o ejecutivos. A continuación, mostramos algunos de los trabajos pertenecientes a estas dos comunidades.

Tabla 9: Algunos trabajos pertenecientes a las comunidades.

Grupo 1	Grupo 2
Research Analyst	Product Designer
DataOps Engineer	Applied Machine Learning Scientist
Decision Scientist	Safety Data Management Specialist
Analytics Lead	Data Strategy Manager
Applied AI ML Lead	Financial Data Analyst
Autonomous Vehicle Technician	Solutions Engineer
Data Quality Lead	Data Operations Manager
Enterprise Account Executive	Marketing Analyst
Customer Success Manager	Stage
Principal Data Scientist	Data Strategy Lead

4. Conclusiones

El análisis de la red de trabajos y salarios en ciencia de datos permitió identificar patrones significativos:

1. Diversidad salarial: Roles como Software Engineer, Data Scientist, Machine Learning Engineer y Data Engineer mostraron la mayor centralidad de grado, reflejando una amplia variabilidad en sus remuneraciones, posiblemente asociada a factores como experiencia, ubicación o sector. En contraste, trabajos como AI Software Development Engineer presentaron baja diversidad, sugiriendo mercados más estandarizados.
2. Roles estratégicos: Las medidas de intermediación destacaron puestos como Data Architect y Product Manager como «puentes» entre categorías salariales, indicando posibles trayectorias de movilidad laboral hacia salarios más altos.
3. Estructura de la red: El coeficiente de mezcla assortativa negativo (-0.295) confirmó que los trabajos tienden a conectarse con otros de diversidad salarial opuesta, reforzando la existencia de transiciones entre niveles extremos (ej: de 15,000 a 800,000 USD).
4. Comunidades: La detección de dos comunidades principales sugirió una división natural entre roles con rangos salariales similares, posiblemente vinculada a niveles de experiencia o regiones geográficas.

Limitaciones y futuras direcciones:

1. El análisis no consideró variables como experiencia o tamaño de la empresa, lo que podría refinarse integrando más datos del dataset original.
2. Sería valioso explorar la evolución temporal de la red para identificar tendencias en la disparidad salarial.

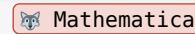
Este proyecto demuestra que la teoría de grafos es una herramienta poderosa para mapear desigualdades y oportunidades en el mercado laboral, ofreciendo datos prácticos para profesionales que buscan crecimiento y para empresas que diseñan políticas de compensación.

5. Códigos

```

1  partition[g_] := Module[{vertex, u, v, c, list},
2    vertex = VertexList[g];
3    u = {};
4    v = {};
5    c = Complement[vertex, Join[u, v]]; (* Vértices que no están en u ni en v *)
6
7    While[c != {}, (* Mientras haya vértices no asignados *)
8      list = GraphDistance[g, c[[1]]]; (* Distancias de c[[1]] a todos los vértices *)
9
10     (* Iteramos sobre la lista de distancias *)
11     For[j = 1, j <= Length[list], j++,
12       If[NumericQ[list[[j]]], (* Verificamos si la distancia es numérica *)
13         If[EvenQ[list[[j]]],
14           AppendTo[u, vertex[[j]]], (* Si la distancia es par, lo asignamos a u *)
15           AppendTo[v, vertex[[j]]] (* Si la distancia es impar, lo asignamos a v *)
16         ]
17       ]
18     ];
19
20     c = Complement[vertex, Join[u, v]] (* Actualizamos c con los vértices no asignados *)
21   ];
22
23   {u, v} (* Devolvemos los dos conjuntos de vértices *)
24 ]

```

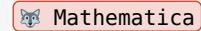


Código 1: Función que retorna los conjuntos particionados de un grafo bipartito (Wolfram).

```

1  projections[graph_, setUorV_] :=
2      Module[{set, l, projectedGraph, list, list1, added, list2},
3
4          set = Switch[setUorV,
5              U, partition[graph][[1]],
6              V, partition[graph][[2]]
7          ];
8          l = Length[set];
9          projectedGraph = Graph[{}];
10         (* Lista de vecinos de cada vértice del conjunto seleccionado *)
11         list = AdjacencyList[graph, #] & /@ set;
12         (* Itera sobre cada vértice en el conjunto seleccionado (set). *)
13         For[i = 1, i <= l, i++,
14
15             list1 = list[[i]];
16             added = False;
17
18             (* Verifica si hay intersección entre vecinos de i y j *)
19             For[j = i + 1, j <= l, j++,
20                 list2 = list[[j]];
21
22                 If[Intersection[list1, list2] != {},
23                     projectedGraph = EdgeAdd[projectedGraph, set[[i]] <-> set[[j]]];
24                     added = True
25                 ]
26
27             ];
28
29             (* Si el vértice no fue conectado a nadie, se agrega como nodo aislado *)
30             If[!added,
31                 projectedGraph = VertexAdd[projectedGraph, set[[i]]]
32             ]
33         ];
34
35         projectedGraph
36     ]

```



Código 2: Función que retorna el grafo proyectado elegido (Wolfram).