UNIVERSIDAD SAN SEBASTÍAN

FACULTAD INGENIERÍA, ARQUITECTURA Y DISEÑO

ESCUELA DE INGENIERÍA

INGENIERÍA CIVIL EN MINAS

SEDE BELLAVISTA

TALLER EN EMPRESA I

"Modelamiento de viscosidad de Yoduro de Potasio con modelos predictivos teóricos"

AUTHIEVRE-CRC

MARÍA JOSÉ JORQUERA ARREPOL

CAMILA MARTÍNEZ FERNÁNDEZ

ANDRÉS SOTO BUBERT

JULIO, 2024

RESUMEN EJECUTIVO

En este ensayo técnico se examinan cuatro modelos diferentes para la predicción de la viscosidad en soluciones de yoduro de potasio (KI), basados en datos experimentales recopilados de una base de datos en formato Excel. Cada modelo ha sido desarrollado utilizando técnicas específicas: la regresión polinómica en Mathematica 5.0, métodos avanzados de machine learning en Python, un modelo termodinámico y reológico especializado para KI, y redes neuronales profundas para inteligencia artificial. La evaluación de estos modelos se ha llevado a cabo mediante la comparación de las viscosidades predichas con los valores experimentales, utilizando métricas para determinar su precisión y fiabilidad.

Este estudio subraya la relevancia de seleccionar el modelo más adecuado según las necesidades específicas del entorno industrial y científico, destacando su utilidad para optimizar el control de las propiedades viscosas en aplicaciones prácticas de soluciones de KI.

ÍNDICE

MOTIVACIONES

La industria del litio, especialmente en regiones desérticas del norte de Chile y Argentina, enfrenta desafíos significativos relacionados con las fluctuaciones térmicas extremas y su composición química. Las oscilaciones entre -10°C y 50°C pueden causar problemas y para eso se tiene como objetivo diseñar y hacer estimaciones de CAPEX y OPEX entre otros. Comprender cómo estas variaciones de temperatura afectan la densidad y la viscosidad de las salmueras es fundamental para diseñar soluciones de transporte seguras y eficientes. Este reto motiva la búsqueda de modelos precisos para predecir el comportamiento de las salmueras en diferentes condiciones ambientales.

La necesidad de modelos precisos se destaca en la industria del litio para simular procesos de producción y transporte. Estos modelos permiten a las empresas anticipar problemas y tomar decisiones informadas, minimizando errores costosos en el diseño de plantas y líneas de producción. La falta de precisión en los datos puede llevar a errores en la implementación de procesos industriales, por lo que contar con modelos que reflejen fielmente la realidad es una motivación importante para el trabajo de investigación.

La industria busca tomar decisiones económicas de motivan usar un método u otra exploración de métodos diferentes a las pozas solares podría mejorar la eficiencia y reducir los costos. Sin embargo, cada método tiene implicaciones para la composición y las propiedades de las salmueras. Estudiar estas implicaciones y cómo afectan el transporte y los procesos químicos es una motivación clave para este trabajo.

La simulación y el modelado se han convertido en herramientas esenciales para explorar diferentes escenarios antes de construir una planta industrial. Al minimizar el error entre experimentación y simulación, las empresas pueden ahorrar recursos y tiempo, evitando errores costosos. Esto motiva el desarrollo de modelos que sean precisos y adaptables a diferentes escenarios industriales.

INTRODUCCIÓN

El transporte de salmueras en la industria del litio plantea desafíos únicos, especialmente en áreas desérticas donde las oscilaciones térmicas pueden ser extremas. Los cambios de temperatura pueden alterar la viscosidad y la densidad de las salmueras, lo que a su vez puede comprometer la integridad de la infraestructura y disminuir la eficiencia de los procesos de producción.

En este contexto, se ha planteado el estudio de rutas de producción de litio mediante pozas solares y extracción directa, evaluando cuál ofrece la mayor seguridad y eficiencia. Dado que las salmueras utilizadas en la industria del litio contienen una amplia gama de componentes químicos, es crucial tener datos precisos sobre su densidad y viscosidad, que pueden variar según la temperatura y la presión atmosférica. La implementación de modelos precisos para simular procesos químicos y líneas de producción es esencial para minimizar errores y optimizar los procesos industriales.

En esta investigación, se trabajará específicamente con yoduro de potasio (KI), utilizando cuatro modelos diferentes incluyendo el de Laliberté, M. Estos modelos considerarán variables como la temperatura, entre 5°C y 95°C, y la composición en molalidad para demostrar cómo varía la viscosidad de las salmueras, proporcionando datos claves para mejorar la eficiencia, predicción y seguridad en la producción de litio.

La problemática central del estudio radica en la necesidad de contar con modelos precisos y actualizados para calcular la viscosidad de salmueras, especialmente en entornos con condiciones climáticas extremas. La capacidad de simular y predecir el comportamiento de las salmueras permite identificar y solucionar problemas antes de que causen daños a la infraestructura o a la producción. A través de este estudio, se busca mejorar la precisión de los modelos y abordar las limitaciones existentes, contribuyendo a la creación de soluciones seguras y eficaces para el transporte de salmueras en la industria del litio.

OBJETIVO GENERAL

Elaborar modelos predictivos teóricos de viscosidad utilizando datos experimentales, que posibiliten predecir esta propiedad en función de la temperatura y la composición.

OBJETIVOS ESPECIFICOS

Minimizar el error entre experimentos y simulación a través de la obtención de datos experimentales precisos. Esto incluye la medición de la viscosidad a diferentes temperaturas, molalidades y comportamiento térmico. Estos datos serán utilizados para calibrar y validar modelos de simulación, asegurando que las predicciones sean fiables.

ANTECEDENTES

En el ámbito de la predicción de la viscosidad de soluciones de yoduro de potasio (KI), se han desarrollado diversos modelos con enfoques y metodologías distintas:

Modelo A (Laliberté, M.): Diseñado específicamente para soluciones acuosas, este modelo combina principios termodinámicos y reológicos para proporcionar predicciones precisas de la viscosidad en función de la molalidad y la temperatura. Su enfoque especializado permite una mayor precisión en contextos específicos de soluciones de KI.

Modelo B (Python): Basado en técnicas de machine learning, este modelo predice la viscosidad identificando patrones en los datos experimentales. Su fortaleza radica en su capacidad para ser robusto frente a variaciones en las condiciones experimentales, adaptándose bien a datos nuevos y complejos.

Modelo C (Mathematica 5.0): Este modelo emplea una regresión polinómica para establecer la relación entre la molalidad y la temperatura con la viscosidad. Ofrece una buena aproximación en un amplio rango de condiciones experimentales, siendo útil para obtener

resultados rápidos y precisos en situaciones variadas.

Modelo D (Inteligencia Artificial): Este modelo emplea algoritmos avanzados de inteligencia artificial, como redes neuronales, para predecir la viscosidad de las soluciones de KI. Utilizando grandes conjuntos de datos experimentales, el modelo aprende las complejas relaciones no lineales entre las variables (molalidad y temperatura) y la viscosidad. Su capacidad para manejar volúmenes de datos y adaptarse a patrones complejos lo convierte en una herramienta poderosa y precisa para la predicción de la viscosidad en diversas condiciones.

Para validar la precisión de estos modelos, se comparan las viscosidades experimentales con las calculadas por cada modelo. Este proceso de validación asegura que los modelos proporcionen predicciones fiables, lo que es crucial para aplicaciones industriales y científicas donde el control preciso de la viscosidad de la salmuera es esencial. Los resultados de estas comparaciones ofrecen una base para la implementación de estos modelos en situaciones práctica

DESARROLLO

Para realizar los siguientes modelos, los criterios de entrada y salida que se utilizaron fue:

Entradas:

- X_1: Molalidad
- X_2: Temperatura

Salida:

- Y_1: Viscosidad

Los parámetros utilizados de molalidad fueron desde 0,005 hasta 8,5773 y la temperatura fue de 5°C a 95°C.

Modelos utilizados

Modelo A (Laliberté): Se ha desarrollado un nuevo modelo para calcular la viscosidad de soluciones acuosas. Los parámetros para 74 solutos se establecieron en base a una revisión crítica de la literatura para soluciones de un soluto en agua, con más de 9000 puntos incluidos. La diferencia media entre la viscosidad calculada y la viscosidad experimental es inferior al 0,1 %, y la desviación típica de esta diferencia es del 3,7 % de la viscosidad experimental media. El modelo se validó mediante la estimación de la viscosidad publicada para sistemas de más de un soluto en agua. La diferencia media entre los valores experimentales y los calculados para 1700 puntos es de -2,7 %, y la desviación estándar de esta diferencia es del 16 % de la viscosidad experimental media. La mediana de la desviación estándar de la diferencia entre los valores experimentales y los calculados es del 3,5 % de la viscosidad experimental. (Laliberte 20070)

name	KH_2PO_4	KI
υ_1	49.178	2.9858
υ_2	2.0655	2.3776
υ_3	1.2471	0.49977
υ_4	0.012350	0.0093425
υ_5	−0.67910	1.7089
υ_6	17.896	−0.0016505
min t/°C	19.95	5
max t/°C	44.95	95
max w_i	0.232	0.627
average viscosity residual δ_η/%	0.63	−0.01
SD of viscosity residual s_{δ_η}/%	9.2	1.2
no. of points in correlation	60	245
no. of inconsistent points	0	0
references	47,[a] 50, 176, 184[a]	43, 62, 94, 142, 164, 179, 183, 196, 223, 228, 250, 253, 254

Imagen 1: Hojas de cálculo para todas las soluciones presentadas en la revisión original. (Ver selección en rojo)

Se encontró que el siguiente modelo era capaz de representar adecuadamente las viscosidades de los solutos:

$$\ln(\eta_i/\text{mPa·s}) = \frac{\upsilon_1(1-w_w)^{\upsilon_2}+\upsilon_3}{(\upsilon_4 t/°\text{C}+1)(\upsilon_5(1-w_w)^{\upsilon_6}+1)} \qquad (13)$$

Ecuación 1: Representación para calcular los solutos

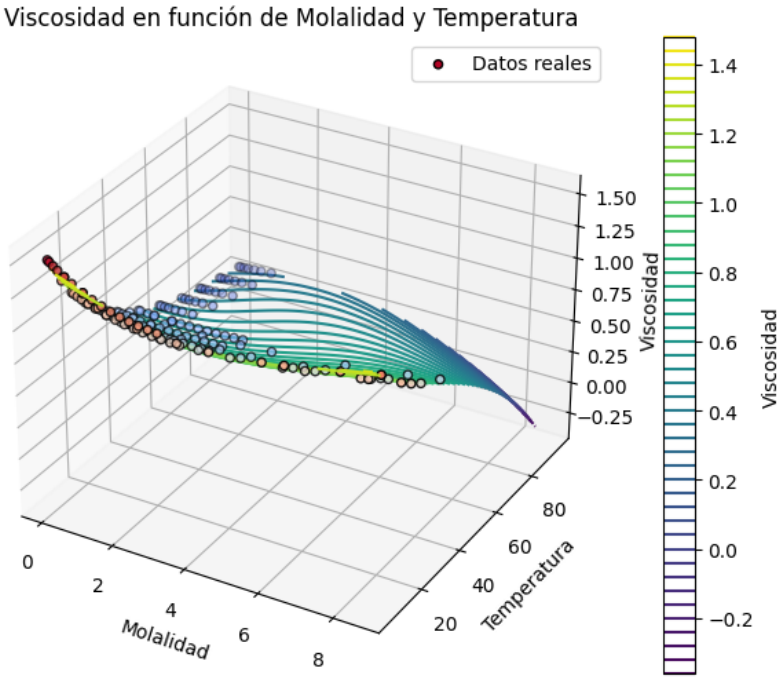

Gráfico 1: Viscosidad en función de Molalidad vs Temperatura

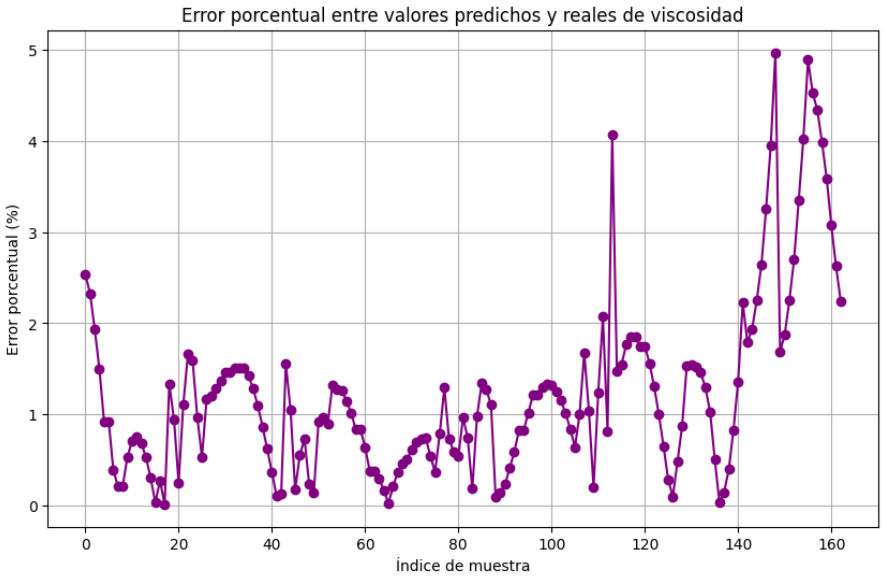

Gráfico 2: Error porcentual entre valores predictivos y reales

Modelo B (Python): Utiliza técnicas de machine learning para predecir la viscosidad basada en patrones identificados en los datos experimentales. Este modelo es robusto frente a variaciones en las condiciones experimentales.

Para esto fue necesaria la recolección y preparación de los datos experimentales para diferentes combinaciones de la molalidad y temperaturas dentro de los rangos especificados anteriormente.

Se asegura de que los datos estén limpios y listos para su uso en el modelo Python. Luego, se divide el conjunto de datos en dos partes: entrenamiento y prueba, esto es para poder evaluar el rendimiento del modelo.

El siguiente paso es entrenar el modelo con el conjunto de entrenamiento, donde el algoritmo identifica patrones en los datos para relacionarlos con nuestras entradas y salida.

Al evaluar el modelo con el conjunto de prueba, se verificó su precisión y robustez.

Se usaron métricas de rendimiento como el error cuadrático medio, el coeficiente de determinación, entre otros, para poder medir la efectividad de este modelo.

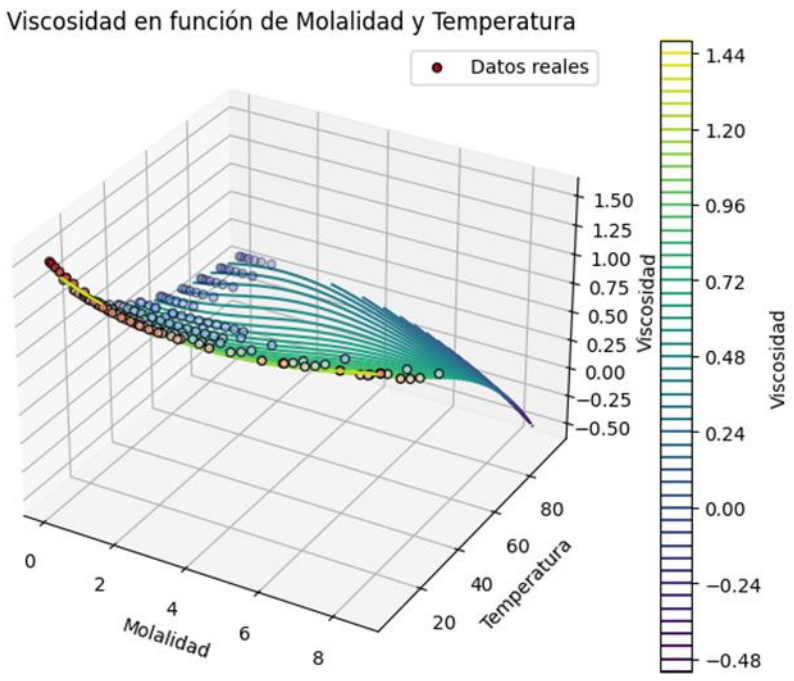

Gráfico 3: Viscosidad en función de Molalidad vs Temperatura

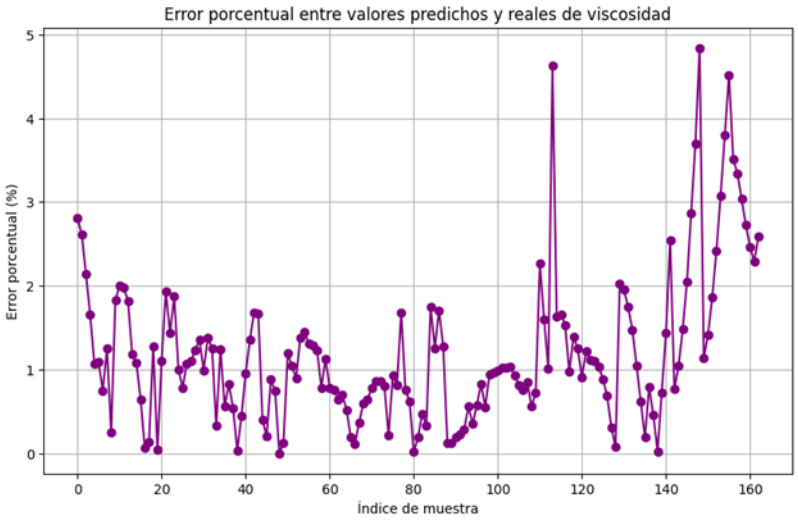

Gráfico 4: Error porcentual entre valores predictivos y reales

La función no solo ajusta los modelos polinómicos y calcula los errores, sino que también imprime la ecuación polinómica obtenida para cada grado. Esto permite comprender mejor cómo se obtienen los resultados y facilita el análisis del modelo ajustado. (VER ANEXO 5)

Modelo C (Mathematica 5.0): Mathematica 5.0 es una herramienta poderosa para modelar y analizar la relación entre molalidad, temperatura y viscosidad, permitiendo la generación de modelos matemáticos precisos y la visualización de resultados.

Para comenzar, se obtienen los datos experimentales para las diferentes combinaciones de molalidad y temperatura según los rangos establecidos.

Seleccionamos un modelo matemático adecuado para describir la relación entre x_1, x_2 y x_3, una vez realizado lo anterior, se utilizan las funciones de ajuste del software para encontrar los parámetros del modelo que mejor se ajusten a los datos experimentales.

El modelo ajustado lo utilizamos para evaluar la viscosidad en diferentes combinaciones según el rango de interés.

En los gráficos podremos visualizar como cambia la viscosidad con respecto a la molalidad y la temperatura.

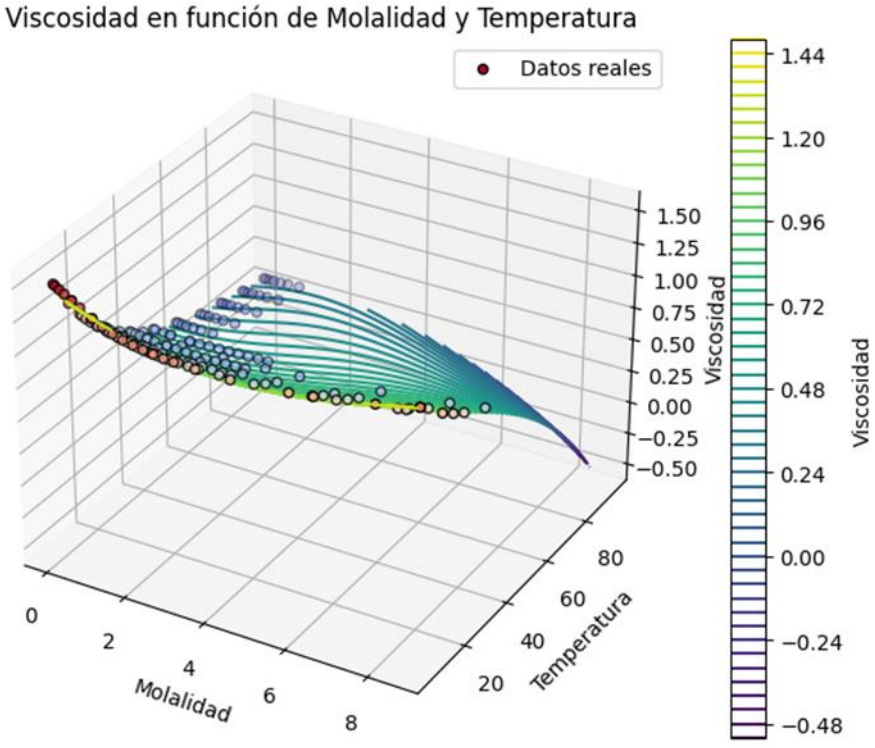

Gráfico 5: Viscosidad en función de Molalidad vs Temperatura

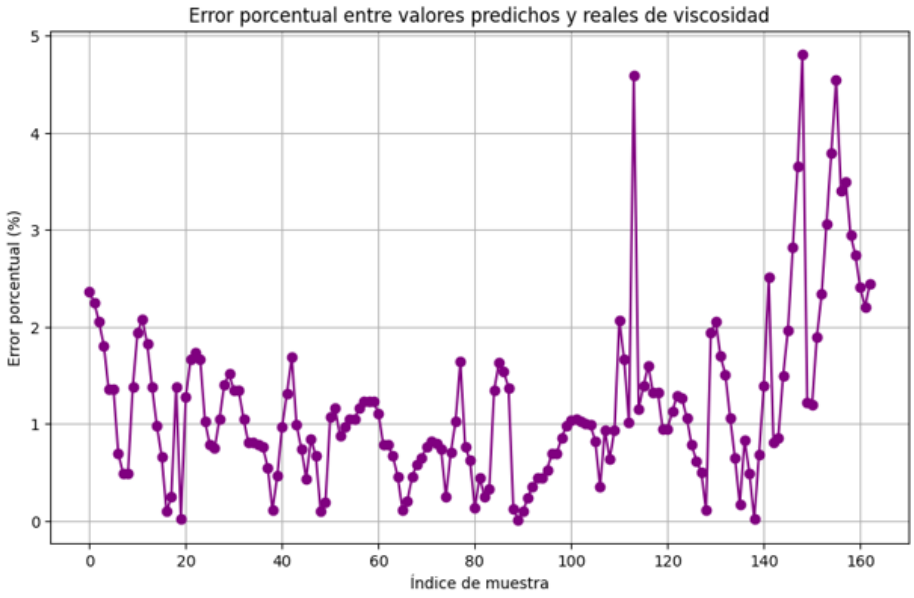

Gráfico 6: Error porcentual entre valores predictivos y reales

La función no solo ajusta los modelos polinómicos y calcula los errores, sino que también imprime la ecuación polinómica obtenida para cada grado. Esto permite comprender mejor cómo se obtienen los resultados y facilita el análisis del modelo ajustado. (VER ANEXO 6)

Modelo D (Inteligencia Artificial):

Basándonos en los otros modelos, se introdujeron las variables de entrada y se creó un nuevo sistema para calcular la viscosidad. No hay parámetros de salida específicos, excepto que la salida debe ser la viscosidad. Este nuevo modelo se desarrolló utilizando inteligencia artificial para garantizar precisión y robustez en los cálculos.

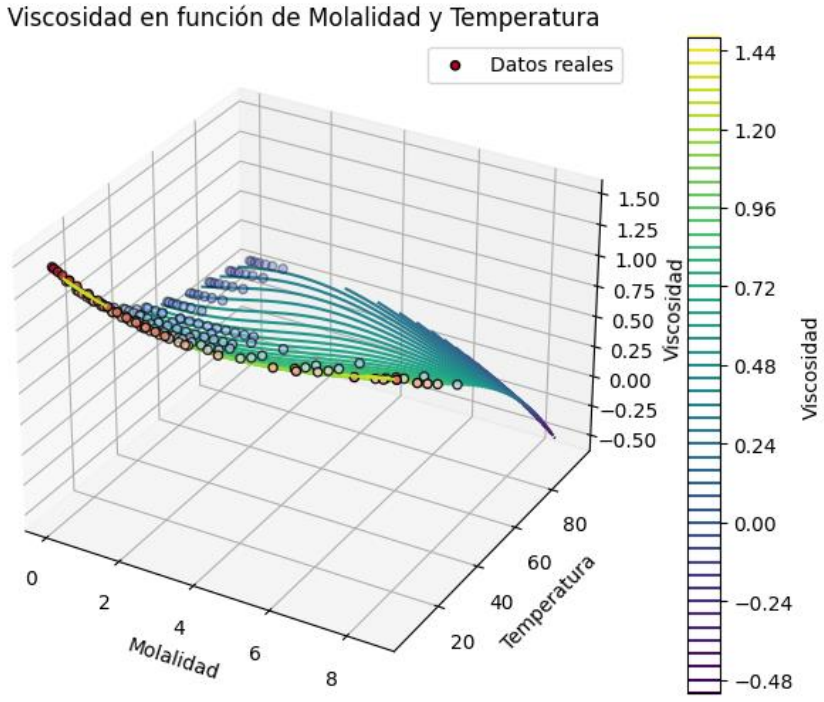

Gráfico 6: Viscosidad en función de Molalidad vs Temperatura

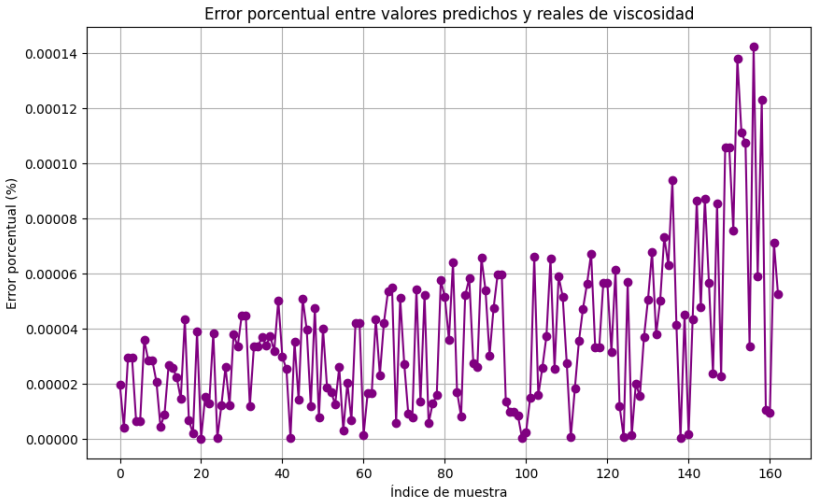

Gráfico 7: Error porcentual entre valores predictivos y reales

La función no solo ajusta los modelos polinómicos y calcula los errores, sino que también imprime la ecuación polinómica obtenida para cada grado. Esto permite comprender mejor cómo se obtienen los resultados y facilita el análisis del modelo ajustado. (VER ANEXO 7)

CONCLUSIÓN

La tabla presenta una comparación de la viscosidad experimental (V. exp) con varias viscosidades calculadas (V. cal) usando diferentes métodos: Python (VER: ANEXO 2), Mathematica 5.0 (VER: ANEXO 3), Inteligencia Artificial (IA) (VER: ANEXO 4) y el modelo de Laliberté, M. (VER: ANEXO 1). Se evaluaron varios parámetros estadísticos para cada modelo, incluidos el Desvío Absoluto Medio (AAD), la Desviación Estándar (SD) y el Error Porcentual Promedio.

En términos del Desvío Absoluto Medio (AAD), los valores para Python y el modelo de Laliberté son idénticos (0.0046), lo que sugiere una precisión similar entre estos dos métodos. Mathematica 5.0 presenta el menor AAD (0.0012), indicando que este método tiene la menor desviación absoluta de los valores experimentales. Por otro lado, el método de IA tiene el mayor AAD (0.0089), lo que sugiere que tiene la mayor desviación absoluta respecto a los valores experimentales.

La Desviación Estándar (SD) sigue un patrón similar al AAD, donde Mathematica 5.0 presenta la menor desviación estándar (0.0021). Python y Laliberté tienen una desviación estándar idéntica (0.0067), mayor que Mathematica pero menor que IA. El método de IA muestra la mayor desviación estándar (0.0113), reflejando una variabilidad mayor en comparación con los otros métodos.

En cuanto al Error Porcentual, el método de IA tiene un error porcentual de 0.0000, lo que indica una precisión perfecta en términos porcentuales. Python presenta el menor error porcentual después de IA (0.0018). Mathematica tiene un error porcentual mayor (0.0097), pero menor que Laliberté (0.0168).

Finalmente, el Error Porcentual Promedio resalta que el método de IA nuevamente se destaca con un error porcentual promedio de 0.0000, indicando que es el método más preciso en términos de error porcentual promedio. Mathematica y Python tienen errores porcentuales promedio muy similares (1.1867 y 1.2045, respectivamente). El modelo de Laliberté tiene el mayor error porcentual promedio (1.2254).

	V. exp / V. cal Python	V. exp/ V. M1	V. exp/ V. M IA	V. exp / V. Laliberté, M.
AAD	0.0046	0.0012	0.0089	0.0046
SD	0.0067	0.0021	0.0113	0.0067
AAD Promedio	0.0046	0.0012	0.0089	0.0046
SD Promedio	0.0063	0.0021	0.0113	0.0063
Error porcentual	0.0018	0.0097	0.0000	0.0168
Error porcentual promedio	1.2045	1.1867	0.0000	1.2254

Tabla 1: Resumen de parámetros de viscosidades experimentales comparadas con Mathematica, Inteligencia Artificial, Python y el modelo de Laliberté, M.

El método de Inteligencia Artificial se destaca por tener el menor error porcentual promedio, indicando una alta precisión en la predicción de la viscosidad experimental. Mathematica 5.0 también muestra alta precisión con el menor AAD y SD, mientras que los métodos de Python y Laliberté son comparables entre sí en términos de AAD y SD, pero muestran mayor error porcentual promedio que IA y Mathematica.

BIBLIOGRAFÍA

Laliberté, M. (2009). A Model for Calculating the Heat Capacity of Aqueous Solutions, with Updated Density and Viscosity Data. *Journal Of Chemical And Engineering*

Laliberté, M. (2007). Model for Calculating the Viscosity of Aqueous Solutions,. *Journal Of Chemical And Engineering Data/Journal Of Chemical & Engineering Data*, *52*(4), 1507- 1508. https://doi.org/10.1021/je700232s

Base de datos, compartido por OneDrive: Andrés Soto.

ANEXO 1

Modelo A (Laliberté):

				V. exp / V. Laliberté, M.	
molality	t / °C	Viscosity exp / 10^3 Pa*s	V. Laliberté, M.	AAD	SD
0.05	5	1.50693	1.5078	0.0008	-0.0008
0.1	5	1.49561	1.4963	0.0007	-0.0007
0.2	5	1.47235	1.4743	0.0019	-0.0019
0.32	5	1.4467	1.4493	0.0026	-0.0026
0.5	5	1.412	1.4145	0.0025	-0.0025
0.7	5	1.38009	1.3797	0.0003	0.0003
1	5	1.316	1.3345	0.0185	-0.0185
1.25	5	1.281	1.3026	0.0216	-0.0216
1.5	5	1.254	1.2756	0.0216	-0.0216
1.75	5	1.232	1.2530	0.0210	-0.0210
2	5	1.214	1.2343	0.0203	-0.0203
2.25	5	1.204	1.2191	0.0151	-0.0151
2.5	5	1.19	1.2072	0.0172	-0.0172
2.75	5	1.182	1.1982	0.0162	-0.0162
3	5	1.178	1.1918	0.0138	-0.0138
0.0977	10	1.2901	1.2906	0.0005	-0.0005
1.0908	10	1.1597	1.1604	0.0007	-0.0007
2.1515	10	1.0968	1.0890	0.0078	0.0078
3.2215	10	1.0693	1.0660	0.0033	0.0033
4.0789	10	1.072	1.0717	0.0003	0.0003
5.4893	10	1.0972	1.1123	0.0151	-0.0151
6.078	10	1.1272	1.1375	0.0103	-0.0103
7.5368	10	1.2022	1.2147	0.0125	-0.0125
8.5773	10	1.2737	1.2792	0.0055	-0.0055
0.05	15	1.13286	1.1322	0.0007	0.0007
0.1	15	1.12689	1.1259	0.0010	0.0010
0.2	15	1.11434	1.1137	0.0007	0.0007
0.32	15	1.10023	1.0998	0.0005	0.0005
0.5	15	1.086	1.0804	0.0056	0.0056
0.7	15	1.06455	1.0610	0.0035	0.0035
1	15	1.049	1.0358	0.0132	0.0132
1.25	15	1.028	1.0183	0.0097	0.0097
1.5	15	1.009	1.0036	0.0054	0.0054
1.75	15	0.9977	0.9916	0.0061	0.0061
2	15	0.9906	0.9821	0.0085	0.0085
2.25	15	0.9852	0.9749	0.0103	0.0103
2.5	15	0.982	0.9697	0.0123	0.0123
2.75	15	0.9795	0.9664	0.0131	0.0131

3	15	0.9782	0.9648	0.0134	0.0134
0.0977	20	0.9928	0.9929	0.0001	-0.0001
1.0908	20	0.9296	0.9210	0.0086	0.0086
2.1515	20	0.8899	0.8847	0.0052	0.0052
3.2215	20	0.8824	0.8798	0.0026	0.0026
4.0789	20	0.8898	0.8924	0.0026	-0.0026
5.4893	20	0.9267	0.9348	0.0081	-0.0081
6.078	20	0.9518	0.9583	0.0065	-0.0065
7.5368	20	1.0188	1.0272	0.0084	-0.0084
7.7357	20	1.0316	1.0375	0.0059	-0.0059
8.5773	20	1.0816	1.0830	0.0014	-0.0014
0.05	25	0.88757	0.8867	0.0009	0.0009
0.0977	25	0.8834	0.8834	0.0000	0.0000
0.1	25	0.88439	0.8832	0.0012	0.0012
0.2	25	0.87728	0.8766	0.0007	0.0007
0.32	25	0.86941	0.8690	0.0004	0.0004
0.5	25	0.8615	0.8585	0.0030	0.0030
0.7	25	0.8497	0.8480	0.0017	0.0017
1	25	0.8345	0.8345	0.0000	0.0000
1.0908	25	0.8309	0.8309	0.0000	0.0000
1.25	25	0.8257	0.8253	0.0004	0.0004
1.5	25	0.8194	0.8179	0.0015	0.0015
1.75	25	0.8146	0.8123	0.0023	0.0023
2	25	0.8111	0.8082	0.0029	0.0029
2.1515	25	0.8103	0.8065	0.0038	0.0038
2.25	25	0.8093	0.8056	0.0037	0.0037
2.5	25	0.8079	0.8044	0.0035	0.0035
2.75	25	0.8075	0.8045	0.0030	0.0030
3	25	0.8078	0.8057	0.0021	0.0021
3.2215	25	0.8087	0.8078	0.0009	0.0009
4.0789	25	0.818	0.8225	0.0045	-0.0045
5.4893	25	0.8553	0.8651	0.0098	-0.0098
6.078	25	0.8818	0.8879	0.0061	-0.0061
7.5368	25	0.9446	0.9533	0.0087	-0.0087
8.5773	25	1.0042	1.0057	0.0015	-0.0015
0.0977	30	0.7937	0.7923	0.0014	0.0014
1.0908	30	0.7543	0.7550	0.0007	-0.0007
2.1515	30	0.7367	0.7399	0.0032	-0.0032
3.2215	30	0.7442	0.7460	0.0018	-0.0018
4.0789	30	0.7563	0.7624	0.0061	-0.0061
5.4893	30	0.7932	0.8050	0.0118	-0.0118
6.078	30	0.8211	0.8271	0.0060	-0.0060
7.5368	30	0.8792	0.8894	0.0102	-0.0102
7.7357	30	0.8911	0.8986	0.0075	-0.0075

8.5773	30	0.9361	0.9387	0.0026	-0.0026
0.05	35	0.71815	0.7172	0.0009	0.0009
0.1	35	0.71647	0.7155	0.0010	0.0010
0.2	35	0.71247	0.7122	0.0003	0.0003
0.32	35	0.70808	0.7085	0.0004	-0.0004
0.5	35	0.7036	0.7033	0.0003	0.0003
0.7	35	0.69749	0.6982	0.0007	-0.0007
1	35	0.6918	0.6918	0.0000	0.0000
1.25	35	0.688	0.6878	0.0002	0.0002
1.5	35	0.685	0.6850	0.0000	0.0000
1.75	35	0.6833	0.6832	0.0001	0.0001
2	35	0.6829	0.6825	0.0004	0.0004
2.25	35	0.6835	0.6828	0.0007	0.0007
2.5	35	0.6852	0.6841	0.0011	0.0011
2.75	35	0.6872	0.6862	0.0010	0.0010
3	35	0.6901	0.6892	0.0009	0.0009
0.0977	40	0.6494	0.6504	0.0010	-0.0010
1.0908	40	0.6309	0.6346	0.0037	-0.0037
2.1515	40	0.63	0.6329	0.0029	-0.0029
3.2215	40	0.6395	0.6459	0.0064	-0.0064
4.0789	40	0.6517	0.6645	0.0128	-0.0128
6.078	40	0.7181	0.7274	0.0093	-0.0093
7.7357	40	0.7801	0.7928	0.0127	-0.0127
8.5773	40	0.821	0.8289	0.0079	-0.0079
0.05	45	0.59561	0.5951	0.0005	0.0005
0.1	45	0.5948	0.5945	0.0003	0.0003
0.32	45	0.59045	0.5919	0.0015	-0.0015
0.5	45	0.585	0.5901	0.0051	-0.0051
1	45	0.5838	0.5866	0.0028	-0.0028
1.25	45	0.5838	0.5860	0.0022	-0.0022
1.5	45	0.5844	0.5860	0.0016	-0.0016
1.75	45	0.5864	0.5868	0.0004	-0.0004
2	45	0.5889	0.5883	0.0006	0.0006
2.25	45	0.5917	0.5904	0.0013	0.0013
2.5	45	0.5948	0.5933	0.0015	0.0015
2.75	45	0.5975	0.5968	0.0007	0.0007
3	45	0.6006	0.6008	0.0002	-0.0002
0.05	55	0.50399	0.5041	0.0001	-0.0001
0.1	55	0.50368	0.5041	0.0004	-0.0004
0.2	55	0.5029	0.5042	0.0014	-0.0014
0.32	55	0.50198	0.5044	0.0025	-0.0025
0.5	55	0.5009	0.5048	0.0039	-0.0039
0.7	55	0.50042	0.5053	0.0049	-0.0049
1	55	0.49956	0.5066	0.0070	-0.0070

0.1	65	0.43375	0.4347	0.0009	-0.0009
0.2	65	0.43386	0.4357	0.0018	-0.0018
0.32	65	0.43379	0.4369	0.0031	-0.0031
0.5	65	0.43389	0.4387	0.0048	-0.0048
0.7	65	0.43446	0.4407	0.0063	-0.0063
1	65	0.43554	0.4440	0.0085	-0.0085
0.05	75	0.37867	0.3792	0.0005	-0.0005
0.1	75	0.37882	0.3800	0.0012	-0.0012
0.2	75	0.37942	0.3816	0.0022	-0.0022
0.32	75	0.37995	0.3835	0.0036	-0.0036
0.5	75	0.38103	0.3863	0.0053	-0.0053
0.7	75	0.38226	0.3894	0.0071	-0.0071
1	75	0.38444	0.3940	0.0096	-0.0096
0.05	85	0.33443	0.3351	0.0007	-0.0007
0.1	85	0.33473	0.3361	0.0014	-0.0014
0.2	85	0.33563	0.3382	0.0025	-0.0025
0.32	85	0.33652	0.3405	0.0040	-0.0040
0.5	85	0.33815	0.3440	0.0058	-0.0058
0.7	85	0.33978	0.3477	0.0079	-0.0079
1	85	0.34269	0.3533	0.0106	-0.0106
0.05	95	0.29833	0.2992	0.0008	-0.0008
0.1	95	0.29877	0.3003	0.0016	-0.0016
0.2	95	0.2998	0.3026	0.0028	-0.0028
0.32	95	0.30098	0.3053	0.0043	-0.0043
0.5	95	0.30303	0.3092	0.0061	-0.0061
0.7	95	0.30502	0.3134	0.0084	-0.0084
1	95	0.30854	0.3196	0.0111	-0.0111

ANEXO 2

Modelo B (Python):

				V. exp / V. cal	
molality	t / °C	Viscosity exp / 10^3 Pa*s	V. Python	AAD	SD
0.05	5	1.50693	1.5078	0.0008	-0.0008
0.1	5	1.49561	1.4963	0.0007	-0.0007
0.2	5	1.47235	1.4743	0.0019	-0.0019
0.32	5	1.4467	1.4493	0.0026	-0.0026
0.5	5	1.412	1.4145	0.0025	-0.0025
0.7	5	1.38009	1.3797	0.0003	0.0003
1	5	1.316	1.3345	0.0185	-0.0185
1.25	5	1.281	1.3026	0.0216	-0.0216
1.5	5	1.254	1.2756	0.0216	-0.0216
1.75	5	1.232	1.2530	0.0210	-0.0210
2	5	1.214	1.2343	0.0203	-0.0203
2.25	5	1.204	1.2191	0.0151	-0.0151
2.5	5	1.19	1.2072	0.0172	-0.0172
2.75	5	1.182	1.1982	0.0162	-0.0162
3	5	1.178	1.1918	0.0138	-0.0138
0.0977	10	1.2901	1.2906	0.0005	-0.0005
1.0908	10	1.1597	1.1604	0.0007	-0.0007
2.1515	10	1.0968	1.0890	0.0078	0.0078
3.2215	10	1.0693	1.0660	0.0033	0.0033
4.0789	10	1.072	1.0717	0.0003	0.0003
5.4893	10	1.0972	1.1123	0.0151	-0.0151
6.078	10	1.1272	1.1375	0.0103	-0.0103
7.5368	10	1.2022	1.2147	0.0125	-0.0125
8.5773	10	1.2737	1.2792	0.0055	-0.0055
0.05	15	1.13286	1.1322	0.0007	0.0007
0.1	15	1.12689	1.1259	0.0010	0.0010
0.2	15	1.11434	1.1137	0.0007	0.0007
0.32	15	1.10023	1.0998	0.0005	0.0005
0.5	15	1.086	1.0804	0.0056	0.0056
0.7	15	1.06455	1.0610	0.0035	0.0035
1	15	1.049	1.0358	0.0132	0.0132
1.25	15	1.028	1.0183	0.0097	0.0097
1.5	15	1.009	1.0036	0.0054	0.0054
1.75	15	0.9977	0.9916	0.0061	0.0061
2	15	0.9906	0.9821	0.0085	0.0085
2.25	15	0.9852	0.9749	0.0103	0.0103
2.5	15	0.982	0.9697	0.0123	0.0123
2.75	15	0.9795	0.9664	0.0131	0.0131

3	15	0.9782	0.9648	0.0134	0.0134
0.0977	20	0.9928	0.9929	0.0001	-0.0001
1.0908	20	0.9296	0.9210	0.0086	0.0086
2.1515	20	0.8899	0.8847	0.0052	0.0052
3.2215	20	0.8824	0.8798	0.0026	0.0026
4.0789	20	0.8898	0.8924	0.0026	-0.0026
5.4893	20	0.9267	0.9348	0.0081	-0.0081
6.078	20	0.9518	0.9583	0.0065	-0.0065
7.5368	20	1.0188	1.0272	0.0084	-0.0084
7.7357	20	1.0316	1.0375	0.0059	-0.0059
8.5773	20	1.0816	1.0830	0.0014	-0.0014
0.05	25	0.88757	0.8867	0.0009	0.0009
0.0977	25	0.8834	0.8834	0.0000	0.0000
0.1	25	0.88439	0.8832	0.0012	0.0012
0.2	25	0.87728	0.8766	0.0007	0.0007
0.32	25	0.86941	0.8690	0.0004	0.0004
0.5	25	0.8615	0.8585	0.0030	0.0030
0.7	25	0.8497	0.8480	0.0017	0.0017
1	25	0.8345	0.8345	0.0000	0.0000
1.0908	25	0.8309	0.8309	0.0000	0.0000
1.25	25	0.8257	0.8253	0.0004	0.0004
1.5	25	0.8194	0.8179	0.0015	0.0015
1.75	25	0.8146	0.8123	0.0023	0.0023
2	25	0.8111	0.8082	0.0029	0.0029
2.1515	25	0.8103	0.8065	0.0038	0.0038
2.25	25	0.8093	0.8056	0.0037	0.0037
2.5	25	0.8079	0.8044	0.0035	0.0035
2.75	25	0.8075	0.8045	0.0030	0.0030
3	25	0.8078	0.8057	0.0021	0.0021
3.2215	25	0.8087	0.8078	0.0009	0.0009
4.0789	25	0.818	0.8225	0.0045	-0.0045
5.4893	25	0.8553	0.8651	0.0098	-0.0098
6.078	25	0.8818	0.8879	0.0061	-0.0061
7.5368	25	0.9446	0.9533	0.0087	-0.0087
8.5773	25	1.0042	1.0057	0.0015	-0.0015
0.0977	30	0.7937	0.7923	0.0014	0.0014
1.0908	30	0.7543	0.7550	0.0007	-0.0007
2.1515	30	0.7367	0.7399	0.0032	-0.0032
3.2215	30	0.7442	0.7460	0.0018	-0.0018
4.0789	30	0.7563	0.7624	0.0061	-0.0061
5.4893	30	0.7932	0.8050	0.0118	-0.0118
6.078	30	0.8211	0.8271	0.0060	-0.0060
7.5368	30	0.8792	0.8894	0.0102	-0.0102
7.7357	30	0.8911	0.8986	0.0075	-0.0075

8.5773	30	0.9361	0.9387	0.0026	-0.0026
0.05	35	0.71815	0.7172	0.0009	0.0009
0.1	35	0.71647	0.7155	0.0010	0.0010
0.2	35	0.71247	0.7122	0.0003	0.0003
0.32	35	0.70808	0.7085	0.0004	-0.0004
0.5	35	0.7036	0.7033	0.0003	0.0003
0.7	35	0.69749	0.6982	0.0007	-0.0007
1	35	0.6918	0.6918	0.0000	0.0000
1.25	35	0.688	0.6878	0.0002	0.0002
1.5	35	0.685	0.6850	0.0000	0.0000
1.75	35	0.6833	0.6832	0.0001	0.0001
2	35	0.6829	0.6825	0.0004	0.0004
2.25	35	0.6835	0.6828	0.0007	0.0007
2.5	35	0.6852	0.6841	0.0011	0.0011
2.75	35	0.6872	0.6862	0.0010	0.0010
3	35	0.6901	0.6892	0.0009	0.0009
0.0977	40	0.6494	0.6504	0.0010	-0.0010
1.0908	40	0.6309	0.6346	0.0037	-0.0037
2.1515	40	0.63	0.6329	0.0029	-0.0029
3.2215	40	0.6395	0.6459	0.0064	-0.0064
4.0789	40	0.6517	0.6645	0.0128	-0.0128
6.078	40	0.7181	0.7274	0.0093	-0.0093
7.7357	40	0.7801	0.7928	0.0127	-0.0127
8.5773	40	0.821	0.8289	0.0079	-0.0079
0.05	45	0.59561	0.5951	0.0005	0.0005
0.1	45	0.5948	0.5945	0.0003	0.0003
0.32	45	0.59045	0.5919	0.0015	-0.0015
0.5	45	0.585	0.5901	0.0051	-0.0051
1	45	0.5838	0.5866	0.0028	-0.0028
1.25	45	0.5838	0.5860	0.0022	-0.0022
1.5	45	0.5844	0.5860	0.0016	-0.0016
1.75	45	0.5864	0.5868	0.0004	-0.0004
2	45	0.5889	0.5883	0.0006	0.0006
2.25	45	0.5917	0.5904	0.0013	0.0013
2.5	45	0.5948	0.5933	0.0015	0.0015
2.75	45	0.5975	0.5968	0.0007	0.0007
3	45	0.6006	0.6008	0.0002	-0.0002
0.05	55	0.50399	0.5041	0.0001	-0.0001
0.1	55	0.50368	0.5041	0.0004	-0.0004
0.2	55	0.5029	0.5042	0.0014	-0.0014
0.32	55	0.50198	0.5044	0.0025	-0.0025
0.5	55	0.5009	0.5048	0.0039	-0.0039
0.7	55	0.50042	0.5053	0.0049	-0.0049
1	55	0.49956	0.5066	0.0070	-0.0070

0.1	65	0.43375	0.4347	0.0009	-0.0009
0.2	65	0.43386	0.4357	0.0018	-0.0018
0.32	65	0.43379	0.4369	0.0031	-0.0031
0.5	65	0.43389	0.4387	0.0048	-0.0048
0.7	65	0.43446	0.4407	0.0063	-0.0063
1	65	0.43554	0.4440	0.0085	-0.0085
0.05	75	0.37867	0.3792	0.0005	-0.0005
0.1	75	0.37882	0.3800	0.0012	-0.0012
0.2	75	0.37942	0.3816	0.0022	-0.0022
0.32	75	0.37995	0.3835	0.0036	-0.0036
0.5	75	0.38103	0.3863	0.0053	-0.0053
0.7	75	0.38226	0.3894	0.0071	-0.0071
1	75	0.38444	0.3940	0.0096	-0.0096
0.05	85	0.33443	0.3351	0.0007	-0.0007
0.1	85	0.33473	0.3361	0.0014	-0.0014
0.2	85	0.33563	0.3382	0.0025	-0.0025
0.32	85	0.33652	0.3405	0.0040	-0.0040
0.5	85	0.33815	0.3440	0.0058	-0.0058
0.7	85	0.33978	0.3477	0.0079	-0.0079
1	85	0.34269	0.3533	0.0106	-0.0106
0.05	95	0.29833	0.2992	0.0008	-0.0008
0.1	95	0.29877	0.3003	0.0016	-0.0016
0.2	95	0.2998	0.3026	0.0028	-0.0028
0.32	95	0.30098	0.3053	0.0043	-0.0043
0.5	95	0.30303	0.3092	0.0061	-0.0061
0.7	95	0.30502	0.3134	0.0084	-0.0084
1	95	0.30854	0.3196	0.0111	-0.0111

ANEXO 3:

Modelo C (Mathematica 5.0):

				V. exp / V. Math	
molality	t / °C	Viscosity exp / 10^3 Pa*s	V. Mathematicas	AAD	SD
0.05	5	1.50693	1.5008	0.0061	0.0061
0.1	5	1.49561	1.4909	0.0047	0.0047
0.2	5	1.47235	1.4719	0.0005	0.0005
0.32	5	1.4467	1.4497	0.0030	-0.0030
0.5	5	1.412	1.4167	0.0047	-0.0047
0.7	5	1.38009	1.3799	0.0002	0.0002
1	5	1.316	1.3266	0.0106	-0.0106
1.25	5	1.281	1.2872	0.0062	-0.0062
1.5	5	1.254	1.2551	0.0011	-0.0011
1.75	5	1.232	1.2311	0.0009	0.0009
2	5	1.214	1.2143	0.0003	-0.0003
2.25	5	1.204	1.2019	0.0021	0.0021
2.5	5	1.19	1.1914	0.0014	-0.0014
2.75	5	1.182	1.1820	0.0000	0.0000
3	5	1.178	1.1778	0.0002	0.0002
0.0977	10	1.2901	1.2922	0.0021	-0.0021
1.0908	10	1.1597	1.1590	0.0007	0.0007
2.1515	10	1.0968	1.0968	0.0000	0.0000
3.2215	10	1.0693	1.0712	0.0019	-0.0019
4.0789	10	1.072	1.0692	0.0028	0.0028
5.4893	10	1.0972	1.1006	0.0034	-0.0034
6.078	10	1.1272	1.1248	0.0024	0.0024
7.5368	10	1.2022	1.2025	0.0003	-0.0003
8.5773	10	1.2737	1.2737	0.0001	0.0001
0.05	15	1.13286	1.1374	0.0045	-0.0045
0.1	15	1.12689	1.1282	0.0013	-0.0013
0.2	15	1.11434	1.1132	0.0012	0.0012
0.32	15	1.10023	1.0992	0.0010	0.0010
0.5	15	1.086	1.0829	0.0031	0.0031
0.7	15	1.06455	1.0673	0.0027	-0.0027
1	15	1.049	1.0445	0.0045	0.0045
1.25	15	1.028	1.0260	0.0020	0.0020
1.5	15	1.009	1.0100	0.0010	-0.0010
1.75	15	0.9977	0.9979	0.0002	-0.0002
2	15	0.9906	0.9901	0.0005	0.0005
2.25	15	0.9852	0.9855	0.0003	-0.0003
2.5	15	0.982	0.9822	0.0002	-0.0002
2.75	15	0.9795	0.9792	0.0003	0.0003

3	15	0.9782	0.9783	0.0001	-0.0001
0.0977	20	0.9928	1.0002	0.0074	-0.0074
1.0908	20	0.9296	0.9268	0.0028	0.0028
2.1515	20	0.8899	0.8920	0.0021	-0.0021
3.2215	20	0.8824	0.8820	0.0004	0.0004
4.0789	20	0.8898	0.8891	0.0007	0.0007
5.4893	20	0.9267	0.9276	0.0009	-0.0009
6.078	20	0.9518	0.9512	0.0006	0.0006
7.5368	20	1.0188	1.0200	0.0012	-0.0012
7.7357	20	1.0316	1.0304	0.0012	0.0012
8.5773	20	1.0816	1.0817	0.0001	-0.0001
0.05	25	0.88757	0.8918	0.0043	-0.0043
0.0977	25	0.8834	0.8875	0.0041	-0.0041
0.1	25	0.88439	0.8873	0.0030	-0.0030
0.2	25	0.87728	0.8790	0.0017	-0.0017
0.32	25	0.86941	0.8699	0.0005	-0.0005
0.5	25	0.8615	0.8582	0.0033	0.0033
0.7	25	0.8497	0.8474	0.0023	0.0023
1	25	0.8345	0.8346	0.0001	-0.0001
1.0908	25	0.8309	0.8314	0.0005	-0.0005
1.25	25	0.8257	0.8265	0.0008	-0.0008
1.5	25	0.8194	0.8203	0.0009	-0.0009
1.75	25	0.8146	0.8155	0.0009	-0.0009
2	25	0.8111	0.8119	0.0008	-0.0008
2.1515	25	0.8103	0.8102	0.0001	0.0001
2.25	25	0.8093	0.8093	0.0000	0.0000
2.5	25	0.8079	0.8077	0.0002	0.0002
2.75	25	0.8075	0.8070	0.0005	0.0005
3	25	0.8078	0.8072	0.0006	0.0006
3.2215	25	0.8087	0.8081	0.0006	0.0006
4.0789	25	0.818	0.8182	0.0002	-0.0002
5.4893	25	0.8553	0.8572	0.0019	-0.0019
6.078	25	0.8818	0.8800	0.0018	0.0018
7.5368	25	0.9446	0.9450	0.0004	-0.0004
8.5773	25	1.0042	1.0041	0.0001	0.0001
0.0977	30	0.7937	0.7942	0.0005	-0.0005
1.0908	30	0.7543	0.7534	0.0009	0.0009
2.1515	30	0.7367	0.7386	0.0019	-0.0019
3.2215	30	0.7442	0.7424	0.0018	0.0018
4.0789	30	0.7563	0.7562	0.0001	0.0001
5.4893	30	0.7932	0.7963	0.0031	-0.0031
6.078	30	0.8211	0.8181	0.0030	0.0030
7.5368	30	0.8792	0.8807	0.0015	-0.0015
7.7357	30	0.8911	0.8903	0.0008	0.0008

8.5773	30	0.9361	0.9360	0.0001	0.0001
0.05	35	0.71815	0.7177	0.0005	0.0005
0.1	35	0.71647	0.7162	0.0003	0.0003
0.2	35	0.71247	0.7129	0.0004	-0.0004
0.32	35	0.70808	0.7089	0.0008	-0.0008
0.5	35	0.7036	0.7030	0.0006	0.0006
0.7	35	0.69749	0.6973	0.0001	0.0001
1	35	0.6918	0.6910	0.0008	0.0008
1.25	35	0.688	0.6874	0.0006	0.0006
1.5	35	0.685	0.6851	0.0001	-0.0001
1.75	35	0.6833	0.6836	0.0003	-0.0003
2	35	0.6829	0.6830	0.0001	-0.0001
2.25	35	0.6835	0.6834	0.0001	0.0001
2.5	35	0.6852	0.6849	0.0003	0.0003
2.75	35	0.6872	0.6875	0.0003	-0.0003
3	35	0.6901	0.6900	0.0001	0.0001
0.0977	40	0.6494	0.6470	0.0024	0.0024
1.0908	40	0.6309	0.6314	0.0005	-0.0005
2.1515	40	0.63	0.6304	0.0004	-0.0004
3.2215	40	0.6395	0.6380	0.0015	0.0015
4.0789	40	0.6517	0.6529	0.0012	-0.0012
6.078	40	0.7181	0.7177	0.0004	0.0004
7.7357	40	0.7801	0.7803	0.0002	-0.0002
8.5773	40	0.821	0.8209	0.0001	0.0001
0.05	45	0.59561	0.5929	0.0028	0.0028
0.1	45	0.5948	0.5934	0.0014	0.0014
0.32	45	0.59045	0.5910	0.0006	-0.0006
0.5	45	0.585	0.5871	0.0021	-0.0021
1	45	0.5838	0.5820	0.0018	0.0018
1.25	45	0.5838	0.5832	0.0006	0.0006
1.5	45	0.5844	0.5853	0.0009	-0.0009
1.75	45	0.5864	0.5873	0.0009	-0.0009
2	45	0.5889	0.5889	0.0000	0.0000
2.25	45	0.5917	0.5909	0.0008	0.0008
2.5	45	0.5948	0.5942	0.0006	0.0006
2.75	45	0.5975	0.5985	0.0010	-0.0010
3	45	0.6006	0.6003	0.0003	0.0003
0.05	55	0.50399	0.5035	0.0004	0.0004
0.1	55	0.50368	0.5042	0.0006	-0.0006
0.2	55	0.5029	0.5026	0.0003	0.0003
0.32	55	0.50198	0.5021	0.0001	-0.0001
0.5	55	0.5009	0.5009	0.0000	0.0000
0.7	55	0.50042	0.5004	0.0000	0.0000
1	55	0.49956	0.4996	0.0000	0.0000

0.1	65	0.43375	0.4338	0.0001	-0.0001
0.2	65	0.43386	0.4339	0.0000	0.0000
0.32	65	0.43379	0.4338	0.0000	0.0000
0.5	65	0.43389	0.4339	0.0000	0.0000
0.7	65	0.43446	0.4345	0.0000	0.0000
1	65	0.43554	0.4355	0.0000	0.0000
0.05	75	0.37867	0.3783	0.0004	0.0004
0.1	75	0.37882	0.3793	0.0004	-0.0004
0.2	75	0.37942	0.3791	0.0003	0.0003
0.32	75	0.37995	0.3801	0.0002	-0.0002
0.5	75	0.38103	0.3810	0.0001	0.0001
0.7	75	0.38226	0.3823	0.0000	0.0000
1	75	0.38444	0.3844	0.0000	0.0000
0.05	85	0.33443	0.3339	0.0005	0.0005
0.1	85	0.33473	0.3352	0.0005	-0.0005
0.2	85	0.33563	0.3353	0.0003	0.0003
0.32	85	0.33652	0.3366	0.0001	-0.0001
0.5	85	0.33815	0.3382	0.0000	0.0000
0.7	85	0.33978	0.3398	0.0000	0.0000
1	85	0.34269	0.3427	0.0000	0.0000
0.05	95	0.29833	0.2979	0.0004	0.0004
0.1	95	0.29877	0.2993	0.0005	-0.0005
0.2	95	0.2998	0.2995	0.0003	0.0003
0.32	95	0.30098	0.3011	0.0001	-0.0001
0.5	95	0.30303	0.3030	0.0001	0.0001
0.7	95	0.30502	0.3050	0.0000	0.0000
1	95	0.30854	0.3085	0.0000	0.0000

ANEXO 4:

Modelo D (Inteligencia Artificial):

				V. exp / V. M IA	
molality	t / °C	Viscosity exp / 10^3 Pa*s	V. Modelo IA	AAD	SD
0.05	5	1.50693	1.4646	0.0424	0.0424
0.1	5	1.49561	1.4566	0.0390	0.0390
0.2	5	1.47235	1.4409	0.0314	0.0314
0.32	5	1.4467	1.4228	0.0239	0.0239
0.5	5	1.412	1.3968	0.0152	0.0152
0.7	5	1.38009	1.3697	0.0104	0.0104
1	5	1.316	1.3325	0.0165	-0.0165
1.25	5	1.281	1.3045	0.0235	-0.0235
1.5	5	1.254	1.2791	0.0251	-0.0251
1.75	5	1.232	1.2563	0.0243	-0.0243
2	5	1.214	1.2361	0.0221	-0.0221
2.25	5	1.204	1.2183	0.0143	-0.0143
2.5	5	1.19	1.2028	0.0128	-0.0128
2.75	5	1.182	1.1897	0.0077	-0.0077
3	5	1.178	1.1788	0.0008	-0.0008
0.0977	10	1.2901	1.2883	0.0018	0.0018
1.0908	10	1.1597	1.1745	0.0148	-0.0148
2.1515	10	1.0968	1.0964	0.0004	0.0004
3.2215	10	1.0693	1.0574	0.0119	0.0119
4.0789	10	1.072	1.0513	0.0207	0.0207
5.4893	10	1.0972	1.0814	0.0158	0.0158
6.078	10	1.1272	1.1061	0.0211	0.0211
7.5368	10	1.2022	1.1902	0.0120	0.0120
8.5773	10	1.2737	1.2637	0.0100	0.0100
0.05	15	1.13286	1.1451	0.0122	-0.0122
0.1	15	1.12689	1.1393	0.0124	-0.0124
0.2	15	1.11434	1.1281	0.0137	-0.0137
0.32	15	1.10023	1.1151	0.0149	-0.0149
0.5	15	1.086	1.0968	0.0108	-0.0108
0.7	15	1.06455	1.0779	0.0134	-0.0134
1	15	1.049	1.0524	0.0034	-0.0034
1.25	15	1.028	1.0338	0.0058	-0.0058
1.5	15	1.009	1.0174	0.0084	-0.0084
1.75	15	0.9977	1.0031	0.0054	-0.0054
2	15	0.9906	0.9909	0.0003	-0.0003
2.25	15	0.9852	0.9808	0.0044	0.0044
2.5	15	0.982	0.9726	0.0094	0.0094
2.75	15	0.9795	0.9663	0.0133	0.0133

3	15	0.9782	0.9617	0.0165	0.0165
0.0977	20	0.9928	1.0094	0.0166	-0.0166
1.0908	20	0.9296	0.9333	0.0037	-0.0037
2.1515	20	0.8899	0.8881	0.0018	0.0018
3.2215	20	0.8824	0.8746	0.0078	0.0078
4.0789	20	0.8898	0.8832	0.0066	0.0066
5.4893	20	0.9267	0.9267	0.0000	0.0000
6.078	20	0.9518	0.9530	0.0012	-0.0012
7.5368	20	1.0188	1.0310	0.0122	-0.0122
7.7357	20	1.0316	1.0424	0.0108	-0.0108
8.5773	20	1.0816	1.0913	0.0097	-0.0097
0.05	25	0.88757	0.8999	0.0123	-0.0123
0.0977	25	0.8834	0.8962	0.0128	-0.0128
0.1	25	0.88439	0.8960	0.0116	-0.0116
0.2	25	0.87728	0.8886	0.0113	-0.0113
0.32	25	0.86941	0.8801	0.0107	-0.0107
0.5	25	0.8615	0.8683	0.0068	-0.0068
0.7	25	0.8497	0.8564	0.0067	-0.0067
1	25	0.8345	0.8408	0.0063	-0.0063
1.0908	25	0.8309	0.8367	0.0058	-0.0058
1.25	25	0.8257	0.8300	0.0043	-0.0043
1.5	25	0.8194	0.8210	0.0016	-0.0016
1.75	25	0.8146	0.8137	0.0009	0.0009
2	25	0.8111	0.8081	0.0030	0.0030
2.1515	25	0.8103	0.8055	0.0048	0.0048
2.25	25	0.8093	0.8041	0.0052	0.0052
2.5	25	0.8079	0.8016	0.0063	0.0063
2.75	25	0.8075	0.8005	0.0070	0.0070
3	25	0.8078	0.8008	0.0070	0.0070
3.2215	25	0.8087	0.8022	0.0065	0.0065
4.0789	25	0.818	0.8163	0.0017	0.0017
5.4893	25	0.8553	0.8632	0.0079	-0.0079
6.078	25	0.8818	0.8890	0.0072	-0.0072
7.5368	25	0.9446	0.9605	0.0159	-0.0159
8.5773	25	1.0042	1.0118	0.0076	-0.0076
0.0977	30	0.7937	0.7986	0.0049	-0.0049
1.0908	30	0.7543	0.7542	0.0001	0.0001
2.1515	30	0.7367	0.7353	0.0014	0.0014
3.2215	30	0.7442	0.7407	0.0035	0.0035
4.0789	30	0.7563	0.7588	0.0025	-0.0025
5.4893	30	0.7932	0.8071	0.0139	-0.0139
6.078	30	0.8211	0.8314	0.0103	-0.0103
7.5368	30	0.8792	0.8942	0.0150	-0.0150
7.7357	30	0.8911	0.9025	0.0114	-0.0114

8.5773	30	0.9361	0.9350	0.0011	0.0011
0.05	35	0.71815	0.7173	0.0008	0.0008
0.1	35	0.71647	0.7151	0.0014	0.0014
0.2	35	0.71247	0.7108	0.0016	0.0016
0.32	35	0.70808	0.7061	0.0020	0.0020
0.5	35	0.7036	0.6996	0.0040	0.0040
0.7	35	0.69749	0.6935	0.0040	0.0040
1	35	0.6918	0.6861	0.0057	0.0057
1.25	35	0.688	0.6815	0.0065	0.0065
1.5	35	0.685	0.6784	0.0066	0.0066
1.75	35	0.6833	0.6765	0.0068	0.0068
2	35	0.6829	0.6759	0.0070	0.0070
2.25	35	0.6835	0.6765	0.0070	0.0070
2.5	35	0.6852	0.6781	0.0071	0.0071
2.75	35	0.6872	0.6808	0.0064	0.0064
3	35	0.6901	0.6845	0.0056	0.0056
0.0977	40	0.6494	0.6445	0.0049	0.0049
1.0908	40	0.6309	0.6255	0.0054	0.0054
2.1515	40	0.63	0.6265	0.0035	0.0035
3.2215	40	0.6395	0.6441	0.0046	-0.0046
4.0789	40	0.6517	0.6665	0.0148	-0.0148
6.078	40	0.7181	0.7296	0.0115	-0.0115
7.7357	40	0.7801	0.7722	0.0079	0.0079
8.5773	40	0.821	0.7830	0.0380	0.0380
0.05	45	0.59561	0.5858	0.0098	0.0098
0.1	45	0.5948	0.5849	0.0099	0.0099
0.32	45	0.59045	0.5814	0.0090	0.0090
0.5	45	0.585	0.5793	0.0057	0.0057
1	45	0.5838	0.5765	0.0073	0.0073
1.25	45	0.5838	0.5767	0.0071	0.0071
1.5	45	0.5844	0.5779	0.0065	0.0065
1.75	45	0.5864	0.5799	0.0065	0.0065
2	45	0.5889	0.5828	0.0061	0.0061
2.25	45	0.5917	0.5864	0.0053	0.0053
2.5	45	0.5948	0.5907	0.0041	0.0041
2.75	45	0.5975	0.5956	0.0019	0.0019
3	45	0.6006	0.6010	0.0004	-0.0004
0.05	55	0.50399	0.4938	0.0102	0.0102
0.1	55	0.50368	0.4939	0.0098	0.0098
0.2	55	0.5029	0.4941	0.0088	0.0088
0.32	55	0.50198	0.4946	0.0074	0.0074
0.5	55	0.5009	0.4956	0.0053	0.0053
0.7	55	0.50042	0.4973	0.0031	0.0031
1	55	0.49956	0.5006	0.0010	-0.0010

0.1	65	0.43375	0.4304	0.0034	0.0034
0.2	65	0.43386	0.4319	0.0019	0.0019
0.32	65	0.43379	0.4339	0.0001	-0.0001
0.5	65	0.43389	0.4370	0.0032	-0.0032
0.7	65	0.43446	0.4407	0.0063	-0.0063
1	65	0.43554	0.4466	0.0110	-0.0110
0.05	75	0.37867	0.3816	0.0029	-0.0029
0.1	75	0.37882	0.3827	0.0039	-0.0039
0.2	75	0.37942	0.3850	0.0056	-0.0056
0.32	75	0.37995	0.3878	0.0078	-0.0078
0.5	75	0.38103	0.3919	0.0109	-0.0109
0.7	75	0.38226	0.3964	0.0142	-0.0142
1	75	0.38444	0.4030	0.0186	-0.0186
0.05	85	0.33443	0.3382	0.0038	-0.0038
0.1	85	0.33473	0.3394	0.0047	-0.0047
0.2	85	0.33563	0.3418	0.0062	-0.0062
0.32	85	0.33652	0.3446	0.0081	-0.0081
0.5	85	0.33815	0.3486	0.0105	-0.0105
0.7	85	0.33978	0.3527	0.0129	-0.0129
1	85	0.34269	0.3582	0.0155	-0.0155
0.05	95	0.29833	0.2878	0.0105	0.0105
0.1	95	0.29877	0.2888	0.0100	0.0100
0.2	95	0.2998	0.2907	0.0091	0.0091
0.32	95	0.30098	0.2928	0.0082	0.0082
0.5	95	0.30303	0.2955	0.0075	0.0075
0.7	95	0.30502	0.2980	0.0070	0.0070
1	95	0.30854	0.3005	0.0080	0.0080

ANEXO 5:
Función obtenida:

```
import numpy as np
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
from sklearn.metrics import mean_squared_error
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

def analyze_viscosity(data):
    # Crear el DataFrame
    df = pd.DataFrame(data)

    # Grados de los modelos polinómicos
    degrees = [1, 2, 3, 4, 5]

    for degree in degrees:
        # Crear características polinómicas
        poly = PolynomialFeatures(degree=degree)
        X_poly = poly.fit_transform(df[['molality', 'temperature']])

        # Ajustar el modelo de regresión polinómica
        model = LinearRegression()
        model.fit(X_poly, df['viscosity'])

        # Coeficientes del modelo
        coefs = model.coef_
        intercept = model.intercept_

        # Crear la función polinómica en formato legible
        terms = ["1"]
        for i in range(1, len(coefs)):
            term = " + ".join([f"{coefs[i]:.4f} * x{i}" for i in range(1, len(coefs))])
            terms.append(term)

        polynomial_function = f"f(x) = {intercept:.4f} + " + " + ".join(terms)
        print(f"Modelo de grado {degree}:")
        print(polynomial_function)
        print("\n")

        # Predecir los valores de viscosidad
        y_pred = model.predict(X_poly)

        # Calcular el error porcentual
        error_percentual = np.abs((df['viscosity'] - y_pred) / df['viscosity']) * 100

        # Encontrar el menor porcentaje de error
        min_error = np.min(error_percentual)
        print(f"Error porcentual mínimo para el modelo de grado {degree}: {min_error:.2f}%")
```

```
    # Calcular el error porcentual promedio
    error_percentual_promedio = np.mean(error_percentual)
    print(f"Error porcentual promedio para el modelo de grado {degree}:
{error_percentual_promedio:.2f}%")

    # Crear una malla de puntos para la visualización
    molality_range = np.linspace(df['molality'].min(), df['molality'].max(), 100)
    temperature_range = np.linspace(df['temperature'].min(), df['temperature'].max(), 100)
    M, T = np.meshgrid(molality_range, temperature_range)
    Z = model.predict(poly.fit_transform(np.array([M.flatten(), T.flatten()]).T)).reshape(M.shape)

    # Configurar la figura 3D
    fig = plt.figure(figsize=(10, 6))
    ax = fig.add_subplot(111, projection='3d')

    # Graficar el plano de contorno
    contour = ax.contour3D(M, T, Z, 50, cmap='viridis')
    fig.colorbar(contour, ax=ax, label='Viscosidad')
    ax.scatter(df['molality'], df['temperature'], df['viscosity'], c=df['viscosity'], cmap='coolwarm',
edgecolor='black', linewidth=1, label='Datos reales')
    ax.set_xlabel('Molalidad')
    ax.set_ylabel('Temperatura')
    ax.set_zlabel('Viscosidad')
    ax.set_title(f'Viscosidad en función de Molalidad y Temperatura (Grado {degree})')
    plt.legend()
    plt.grid(True)
    plt.show()

    # Gráfica de Error
    plt.figure(figsize=(10, 6))
    plt.plot(df.index, error_percentual, marker='o', linestyle='-', color='purple')
    plt.xlabel('Índice de muestra')
    plt.ylabel('Error porcentual (%)')
    plt.title(f'Error porcentual entre valores predichos y reales de viscosidad (Grado {degree})')
    plt.grid(True)
    plt.show()

    # Crear un DataFrame con los datos y los errores porcentuales
    data_with_errors = {
       'molality': df['molality'],
       'temperature': df['temperature'],
       'viscosity': df['viscosity'],
       'error_percentual': error_percentual
    }

    # Crear un DataFrame a partir del diccionario
    df_with_errors = pd.DataFrame(data_with_errors)

    # Calcular la desviación estándar del error porcentual
    desviacion_estandar = np.std(error_percentual)
```

```
        # Añadir la desviación estándar al DataFrame
        df_with_errors['desviacion_estandar'] = desviacion_estandar

        # Mostrar la tabla actualizada
        print(df_with_errors.to_string())

        # Gráfica de Error con desviación estándar
        plt.figure(figsize=(10, 6))
        plt.errorbar(df.index, error_percentual, yerr=desviacion_estandar, fmt='o', color='purple', ecolor='gray',
capsize=5)
        plt.xlabel('Índice de muestra')
        plt.ylabel('Error porcentual (%)')
        plt.title(f'Error porcentual entre valores predichos y reales de viscosidad con desviación estándar (Grado
{degree})')
        plt.grid(True)
        plt.show()

# Datos proporcionados
data = {
    'molality': [0.05, 0.1, 0.2, 0.32, 0.5, 0.5, 0.7, 1, 1, 1.25, 1.5, 1.75, 2, 2.25, 2.5, 2.75, 3,
    0.0977, 1.0908, 2.1515, 3.2215, 4.0789, 5.4893, 6.078, 7.5368, 8.5773,
    0.05, 0.1, 0.2, 0.32, 0.5, 0.5, 0.7, 1, 1, 1.25, 1.5, 1.75, 2, 2.25, 2.5, 2.75, 3,
    0.0977, 1.0908, 2.1515, 3.2215, 4.0789, 5.4893, 6.078, 7.5368, 7.7357, 8.5773,
    0.05, 0.0977, 0.1, 0.2, 0.32, 0.5, 0.5, 0.7, 1, 1, 1.0908, 1.25, 1.5, 1.75, 2, 2.1515, 2.25, 2.5, 2.75, 3, 3.2215,
4.0789, 5.4893, 6.078, 7.5368, 8.5773,
    0.0977, 1.0908, 2.1515, 3.2215, 4.0789, 5.4893, 6.078, 7.5368, 7.7357, 8.5773,
    0.05, 0.1, 0.2, 0.32, 0.5, 0.5, 0.7, 1, 1, 1.25, 1.5, 1.75, 2, 2.25, 2.5, 2.75, 3,
    0.0977, 1.0908, 2.1515, 3.2215, 4.0789, 6.078, 7.7357, 8.5773,
    0.05, 0.1, 0.32, 0.5, 0.5, 1, 1, 1.25, 1.5, 1.75, 2, 2.25, 2.5, 2.75, 3,
    0.05, 0.1, 0.2, 0.32, 0.5, 0.7, 1,
    0.1, 0.2, 0.32, 0.5, 0.7, 1,
    0.05, 0.1, 0.2, 0.32, 0.5, 0.7, 1,
    0.05, 0.1, 0.2, 0.32, 0.5, 0.7, 1,
    0.05, 0.1, 0.2, 0.32, 0.5, 0.7, 1,
],
    'temperature': [5, 5, 5, 5, 5, 5, 5, 5, 5, 5,
    10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
    15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15,
    20, 20, 20, 20, 20, 20, 20, 20,
    25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25,
    30, 30, 30, 30, 30, 30, 30, 30, 30, 30,
    35, 35, 35, 35, 35, 35, 35, 35, 35, 35, 35, 35, 35, 35, 35, 35, 35,
    40, 40, 40, 40, 40, 40, 40, 40,
    45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45,
    55, 55, 55, 55, 55, 55, 55,
    65, 65, 65, 65, 65, 65,
    75, 75, 75, 75, 75, 75, 75,
    85, 85, 85, 85, 85, 85, 85,
    95, 95, 95, 95, 95, 95, 95
],
    'viscosity': [1.5069, 1.4956, 1.4724, 1.4467, 1.4120, 1.4122, 1.3801, 1.3160, 1.3358, 1.2810,
     1.2540, 1.2320, 1.2140, 1.2040, 1.1900, 1.1820, 1.1780, 1.2901, 1.1597, 1.0968,
```

```
    1.0693, 1.0720, 1.0972, 1.1272, 1.2022, 1.2737,
   1.1329, 1.1269, 1.1143, 1.1002, 1.0860, 1.0818, 1.0646, 1.0490, 1.0395, 1.0280,
    1.0090, 0.9977, 0.9906, 0.9852, 0.9820, 0.9795, 0.9782, 0.9928, 0.9296, 0.8899,
    0.8824, 0.8898, 0.9267, 0.9518, 1.0188, 1.0316, 1.0816,
   0.8876, 0.8834, 0.8844, 0.8773, 0.8694, 0.8615, 0.8586, 0.8497, 0.8345, 0.8355,
    0.8309, 0.8257, 0.8194, 0.8146, 0.8111, 0.8103, 0.8093, 0.8079, 0.8075, 0.8078,
    0.8087, 0.8180, 0.8553, 0.8818, 0.9446, 1.0042,
   0.7937, 0.7543, 0.7367, 0.7442, 0.7563, 0.7932, 0.8211, 0.8792, 0.8911, 0.9361,
   0.7182, 0.7165, 0.7125, 0.7081, 0.7036, 0.7021, 0.6975, 0.6918, 0.6899, 0.6880,
    0.6850, 0.6833, 0.6829, 0.6835, 0.6852, 0.6872, 0.6901,
   0.6494, 0.6309, 0.6300, 0.6395, 0.6517, 0.7181, 0.7801, 0.8210,
   0.5956, 0.5948, 0.5905, 0.5850, 0.5875, 0.5838, 0.5818, 0.5838, 0.5844, 0.5864,
    0.5889, 0.5917, 0.5948, 0.5975, 0.6006,
   0.5040, 0.5037, 0.5029, 0.5020, 0.5009, 0.5004, 0.4996,
   0.4338, 0.4339, 0.4338, 0.4339, 0.4345, 0.4355,
   0.3787, 0.3788, 0.3794, 0.3800, 0.3810, 0.3823, 0.3844,
   0.3344, 0.3347, 0.3356, 0.3365, 0.3382, 0.3398, 0.3427,
   0.2983, 0.2988, 0.2998, 0.3010, 0.3030, 0.3050, 0.3085
   ]
}

# Llamar a la función con los datos proporcionados
analyze_viscosity(data)
```

ANEXO 6:
Función obtenida:

```
import numpy as np
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
from sklearn.metrics import mean_squared_error
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

def analyze_viscosity(data):
    # Crear el DataFrame
    df = pd.DataFrame(data)

    # Grados de los modelos polinómicos
    degrees = [1, 2, 3, 4, 5]

    for degree in degrees:
        # Crear características polinómicas
        poly = PolynomialFeatures(degree=degree)
        X_poly = poly.fit_transform(df[['molality', 'temperature']])

        # Ajustar el modelo de regresión polinómica
        model = LinearRegression()
        model.fit(X_poly, df['viscosity'])

        # Coeficientes del modelo
        coefs = model.coef_
        intercept = model.intercept_

        # Crear la función polinómica en formato legible
        terms = ["1"]
        for i in range(1, len(coefs)):
            term = " + ".join([f"{coefs[i]:.4f} * x{i}" for i in range(1, len(coefs))])
            terms.append(term)

        polynomial_function = f"f(x) = {intercept:.4f} + " + " + ".join(terms)
        print(f"Modelo de grado {degree}:")
        print(polynomial_function)
        print("\n")

        # Predecir los valores de viscosidad
        y_pred = model.predict(X_poly)

        # Calcular el error porcentual
        error_percentual = np.abs((df['viscosity'] - y_pred) / df['viscosity']) * 100

        # Encontrar el menor porcentaje de error
        min_error = np.min(error_percentual)
        print(f"Error porcentual mínimo para el modelo de grado {degree}: {min_error:.2f}%")
```

```
    # Calcular el error porcentual promedio
    error_percentual_promedio = np.mean(error_percentual)
    print(f"Error porcentual promedio para el modelo de grado {degree}:
{error_percentual_promedio:.2f}%")

    # Crear una malla de puntos para la visualización
    molality_range = np.linspace(df['molality'].min(), df['molality'].max(), 100)
    temperature_range = np.linspace(df['temperature'].min(), df['temperature'].max(), 100)
    M, T = np.meshgrid(molality_range, temperature_range)
    Z = model.predict(poly.fit_transform(np.array([M.flatten(), T.flatten()]).T)).reshape(M.shape)

    # Configurar la figura 3D
    fig = plt.figure(figsize=(10, 6))
    ax = fig.add_subplot(111, projection='3d')

    # Graficar el plano de contorno
    contour = ax.contour3D(M, T, Z, 50, cmap='viridis')
    fig.colorbar(contour, ax=ax, label='Viscosidad')
    ax.scatter(df['molality'], df['temperature'], df['viscosity'], c=df['viscosity'], cmap='coolwarm',
edgecolor='black', linewidth=1, label='Datos reales')
    ax.set_xlabel('Molalidad')
    ax.set_ylabel('Temperatura')
    ax.set_zlabel('Viscosidad')
    ax.set_title(f'Viscosidad en función de Molalidad y Temperatura (Grado {degree})')
    plt.legend()
    plt.grid(True)
    plt.show()

    # Gráfica de Error
    plt.figure(figsize=(10, 6))
    plt.plot(df.index, error_percentual, marker='o', linestyle='-', color='purple')
    plt.xlabel('Índice de muestra')
    plt.ylabel('Error porcentual (%)')
    plt.title(f'Error porcentual entre valores predichos y reales de viscosidad (Grado {degree})')
    plt.grid(True)
    plt.show()

    # Crear un DataFrame con los datos y los errores porcentuales
    data_with_errors = {
       'molality': df['molality'],
       'temperature': df['temperature'],
       'viscosity': df['viscosity'],
       'error_percentual': error_percentual
    }

    # Crear un DataFrame a partir del diccionario
    df_with_errors = pd.DataFrame(data_with_errors)

    # Calcular la desviación estándar del error porcentual
    desviacion_estandar = np.std(error_percentual)
```

```
        # Añadir la desviación estándar al DataFrame
        df_with_errors['desviacion_estandar'] = desviacion_estandar

        # Mostrar la tabla actualizada
        print(df_with_errors.to_string())

        # Gráfica de Error con desviación estándar
        plt.figure(figsize=(10, 6))
        plt.errorbar(df.index, error_percentual, yerr=desviacion_estandar, fmt='o', color='purple', ecolor='gray',
capsize=5)
        plt.xlabel('Índice de muestra')
        plt.ylabel('Error porcentual (%)')
        plt.title(f'Error porcentual entre valores predichos y reales de viscosidad con desviación estándar (Grado
{degree})')
        plt.grid(True)
        plt.show()

# Datos proporcionados
data = {
    'molality': [0.05, 0.1, 0.2, 0.32, 0.5, 0.5, 0.7, 1, 1, 1.25, 1.5, 1.75, 2, 2.25, 2.5, 2.75, 3,
    0.0977, 1.0908, 2.1515, 3.2215, 4.0789, 5.4893, 6.078, 7.5368, 8.5773,
    0.05, 0.1, 0.2, 0.32, 0.5, 0.5, 0.7, 1, 1, 1.25, 1.5, 1.75, 2, 2.25, 2.5, 2.75, 3,
    0.0977, 1.0908, 2.1515, 3.2215, 4.0789, 5.4893, 6.078, 7.5368, 7.7357, 8.5773,
    0.05, 0.0977, 0.1, 0.2, 0.32, 0.5, 0.5, 0.7, 1, 1, 1.0908, 1.25, 1.5, 1.75, 2, 2.1515, 2.25, 2.5, 2.75, 3, 3.2215,
4.0789, 5.4893, 6.078, 7.5368, 8.5773,
    0.0977, 1.0908, 2.1515, 3.2215, 4.0789, 5.4893, 6.078, 7.5368, 7.7357, 8.5773,
    0.05, 0.1, 0.2, 0.32, 0.5, 0.5, 0.7, 1, 1, 1.25, 1.5, 1.75, 2, 2.25, 2.5, 2.75, 3,
    0.0977, 1.0908, 2.1515, 3.2215, 4.0789, 6.078, 7.7357, 8.5773,
    0.05, 0.1, 0.32, 0.5, 0.5, 1, 1, 1.25, 1.5, 1.75, 2, 2.25, 2.5, 2.75, 3,
    0.05, 0.1, 0.2, 0.32, 0.5, 0.7, 1,
    0.1, 0.2, 0.32, 0.5, 0.7, 1,
    0.05, 0.1, 0.2, 0.32, 0.5, 0.7, 1,
    0.05, 0.1, 0.2, 0.32, 0.5, 0.7, 1,
    0.05, 0.1, 0.2, 0.32, 0.5, 0.7, 1,
],
    'temperature': [5, 5, 5, 5, 5, 5, 5, 5, 5, 5,
    10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
    15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15,
    20, 20, 20, 20, 20, 20, 20, 20,
    25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25,
    30, 30, 30, 30, 30, 30, 30, 30, 30, 30,
    35, 35, 35, 35, 35, 35, 35, 35, 35, 35, 35, 35, 35, 35, 35, 35, 35,
    40, 40, 40, 40, 40, 40, 40, 40,
    45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45,
    55, 55, 55, 55, 55, 55, 55,
    65, 65, 65, 65, 65, 65,
    75, 75, 75, 75, 75, 75, 75,
    85, 85, 85, 85, 85, 85, 85,
    95, 95, 95, 95, 95, 95, 95
],
    'viscosity': [1.50079734903498, 1.49091277030473, 1.47186807864054, 1.44967032049260,
1.41665165803900,
```

1.41665165803900, 1.37991642828836, 1.32663599465911, 1.32663599465911, 1.28723105931918,
1.25510526216620, 1.23114059506573, 1.21426052749773, 1.20192476839569, 1.19143224635203,
1.18203230818974, 1.17784413590034, 1.29224157445556, 1.15897141052752, 1.09682708014411,
1.07115895883952, 1.06916432189912, 1.10056393440786, 1.12484864787010, 1.20251302102059,
1.27365007268241, 1.13739518853640, 1.12823530951524, 1.11317532091102, 1.09918729374568,
1.08286356322438, 1.08286356322438, 1.06727420731728, 1.04449460164885, 1.04449460164885,
1.02604939098689, 1.00996994245316, 0.99786049462200, 0.99006588255778, 0.98548478227680,
0.98223648270696, 0.97918118514526, 0.97829383021320, 1.00019691074221, 0.92679186151144,
0.89203807917728, 0.88200597739303, 0.88907797534640, 0.92755411871348, 0.95121457628654,
1.01999495147390, 1.03042269309933, 1.08170791814695, 0.89184327272281, 0.88754862448713,
0.88734645703064, 0.87897424250056, 0.86993551101602, 0.85820155316320, 0.85820155316320,
0.84736236060606, 0.83461255387522, 0.83461255387522, 0.83143831169898, 0.82653142317424,
0.82026063658114, 0.81545475949462, 0.81186685843812, 0.81021194192159, 0.80933011021317,
0.80774099911736, 0.80704410222731, 0.80721846274617, 0.80810173965397, 0.81817677738890,
0.85719398448203, 0.87999219113195, 0.94497694629885, 1.00412633884459, 0.79418417357045,
0.75338528292140, 0.73857648740706, 0.74240763577039, 0.75624113789109, 0.79634155663953,
0.81805509933947, 0.88067714431246, 0.89026273904397, 0.93604557945671, 0.71769279704751,
0.71617602485150, 0.71291809304062, 0.70886045935577, 0.70300265627001, 0.70300265627001,
0.69734827397727, 0.69099397484101, 0.69099397484101, 0.68744714225107, 0.68507817308179,
0.68360336663093, 0.68297803804576, 0.68336362827226, 0.68493112880801, 0.68750082125886,
0.69001833169925, 0.64699180612560, 0.63141830548666, 0.63038433622716, 0.63802734053084,
0.65290764683124, 0.71774703020391, 0.78026304170528, 0.82094701961118, 0.59285264130345,
0.59335466961316, 0.59100338827151, 0.58713239648574, 0.58713239648574, 0.58200074883869,
0.58200074883869, 0.58322990935321, 0.58533296905182, 0.58726088378756, 0.58888695667329,
0.59092595757692, 0.59423539621769, 0.59849894886371, 0.60029203863054, 0.50354560390979,
0.50423806241577, 0.50260406567133, 0.50212050259784, 0.50087369472634, 0.50040410782180,
0.49959981675893, 0.43380000000000, 0.43390000000000, 0.43380000000000, 0.43390000000000,
0.43450000000000, 0.43550000000000, 0.37830760963754, 0.37926463979532, 0.37914444811257,
0.38010405912169, 0.38097728427667, 0.38230354727895, 0.38439984176354, 0.33393834836014,
0.33524665390381, 0.33529934035291, 0.33662242671779, 0.33817327469804, 0.33980417341326,
0.34269981383303, 0.29791523613724, 0.29925560905553, 0.29954941501946, 0.30110203662843,
0.30297772577951, 0.30500347833403, 0.30849984483902

]
}

# Llamar a la función con los datos proporcionados
analyze_viscosity(data)

ANEXO 6:
Función obtenida:

```
import numpy as np
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
from sklearn.metrics import mean_squared_error
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

def analyze_viscosity(data):
    # Crear el DataFrame
    df = pd.DataFrame(data)

    # Grados de los modelos polinómicos
    degrees = [1, 2, 3, 4, 5]

    for degree in degrees:
        # Crear características polinómicas
        poly = PolynomialFeatures(degree=degree)
        X_poly = poly.fit_transform(df[['molality', 'temperature']])

        # Ajustar el modelo de regresión polinómica
        model = LinearRegression()
        model.fit(X_poly, df['viscosity'])

        # Coeficientes del modelo
        coefs = model.coef_
        intercept = model.intercept_

        # Crear la función polinómica en formato legible
        terms = ["1"]
        for i in range(1, len(coefs)):
            term = " + ".join([f"{coefs[i]:.4f} * x{i}" for i in range(1, len(coefs))])
            terms.append(term)

        polynomial_function = f"f(x) = {intercept:.4f} + " + " + ".join(terms)
        print(f"Modelo de grado {degree}:")
        print(polynomial_function)
        print("\n")

        # Predecir los valores de viscosidad
        y_pred = model.predict(X_poly)

        # Calcular el error porcentual
        error_percentual = np.abs((df['viscosity'] - y_pred) / df['viscosity']) * 100

        # Encontrar el menor porcentaje de error
        min_error = np.min(error_percentual)
        print(f"Error porcentual mínimo para el modelo de grado {degree}: {min_error:.2f}%")
```

```
    # Calcular el error porcentual promedio
    error_percentual_promedio = np.mean(error_percentual)
    print(f"Error porcentual promedio para el modelo de grado {degree}:
{error_percentual_promedio:.2f}%")

    # Crear una malla de puntos para la visualización
    molality_range = np.linspace(df['molality'].min(), df['molality'].max(), 100)
    temperature_range = np.linspace(df['temperature'].min(), df['temperature'].max(), 100)
    M, T = np.meshgrid(molality_range, temperature_range)
    Z = model.predict(poly.fit_transform(np.array([M.flatten(), T.flatten()]).T)).reshape(M.shape)

    # Configurar la figura 3D
    fig = plt.figure(figsize=(10, 6))
    ax = fig.add_subplot(111, projection='3d')

    # Graficar el plano de contorno
    contour = ax.contour3D(M, T, Z, 50, cmap='viridis')
    fig.colorbar(contour, ax=ax, label='Viscosidad')
    ax.scatter(df['molality'], df['temperature'], df['viscosity'], c=df['viscosity'], cmap='coolwarm',
edgecolor='black', linewidth=1, label='Datos reales')
    ax.set_xlabel('Molalidad')
    ax.set_ylabel('Temperatura')
    ax.set_zlabel('Viscosidad')
    ax.set_title(f'Viscosidad en función de Molalidad y Temperatura (Grado {degree})')
    plt.legend()
    plt.grid(True)
    plt.show()

    # Gráfica de Error
    plt.figure(figsize=(10, 6))
    plt.plot(df.index, error_percentual, marker='o', linestyle='-', color='purple')
    plt.xlabel('Índice de muestra')
    plt.ylabel('Error porcentual (%)')
    plt.title(f'Error porcentual entre valores predichos y reales de viscosidad (Grado {degree})')
    plt.grid(True)
    plt.show()

    # Crear un DataFrame con los datos y los errores porcentuales
    data_with_errors = {
       'molality': df['molality'],
       'temperature': df['temperature'],
       'viscosity': df['viscosity'],
       'error_percentual': error_percentual
    }

    # Crear un DataFrame a partir del diccionario
    df_with_errors = pd.DataFrame(data_with_errors)

    # Calcular la desviación estándar del error porcentual
    desviacion_estandar = np.std(error_percentual)
```

```
        # Añadir la desviación estándar al DataFrame
        df_with_errors['desviacion_estandar'] = desviacion_estandar

        # Mostrar la tabla actualizada
        print(df_with_errors.to_string())

        # Gráfica de Error con desviación estándar
        plt.figure(figsize=(10, 6))
        plt.errorbar(df.index, error_percentual, yerr=desviacion_estandar, fmt='o', color='purple', ecolor='gray',
capsize=5)
        plt.xlabel('Índice de muestra')
        plt.ylabel('Error porcentual (%)')
        plt.title(f'Error porcentual entre valores predichos y reales de viscosidad con desviación estándar (Grado
{degree})')
        plt.grid(True)
        plt.show()

# Datos proporcionados
data = {
    'molality': [0.05, 0.1, 0.2, 0.32, 0.5, 0.5, 0.7, 1, 1, 1.25, 1.5, 1.75, 2, 2.25, 2.5, 2.75, 3,
    0.0977, 1.0908, 2.1515, 3.2215, 4.0789, 5.4893, 6.078, 7.5368, 8.5773,
    0.05, 0.1, 0.2, 0.32, 0.5, 0.5, 0.7, 1, 1, 1.25, 1.5, 1.75, 2, 2.25, 2.5, 2.75, 3,
    0.0977, 1.0908, 2.1515, 3.2215, 4.0789, 5.4893, 6.078, 7.5368, 7.7357, 8.5773,
    0.05, 0.0977, 0.1, 0.2, 0.32, 0.5, 0.5, 0.7, 1, 1, 1.0908, 1.25, 1.5, 1.75, 2, 2.1515, 2.25, 2.5, 2.75, 3, 3.2215,
4.0789, 5.4893, 6.078, 7.5368, 8.5773,
    0.0977, 1.0908, 2.1515, 3.2215, 4.0789, 5.4893, 6.078, 7.5368, 7.7357, 8.5773,
    0.05, 0.1, 0.2, 0.32, 0.5, 0.5, 0.7, 1, 1, 1.25, 1.5, 1.75, 2, 2.25, 2.5, 2.75, 3,
    0.0977, 1.0908, 2.1515, 3.2215, 4.0789, 6.078, 7.7357, 8.5773,
    0.05, 0.1, 0.32, 0.5, 0.5, 1, 1, 1.25, 1.5, 1.75, 2, 2.25, 2.5, 2.75, 3,
    0.05, 0.1, 0.2, 0.32, 0.5, 0.7, 1,
    0.1, 0.2, 0.32, 0.5, 0.7, 1,
    0.05, 0.1, 0.2, 0.32, 0.5, 0.7, 1,
    0.05, 0.1, 0.2, 0.32, 0.5, 0.7, 1,
    0.05, 0.1, 0.2, 0.32, 0.5, 0.7, 1,
],
    'temperature': [5, 5, 5, 5, 5, 5, 5, 5, 5, 5,
    10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
    15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15,
    20, 20, 20, 20, 20, 20, 20, 20,
    25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25,
    30, 30, 30, 30, 30, 30, 30, 30, 30, 30,
    35, 35, 35, 35, 35, 35, 35, 35, 35, 35, 35, 35, 35, 35, 35, 35, 35,
    40, 40, 40, 40, 40, 40, 40, 40,
    45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45, 45,
    55, 55, 55, 55, 55, 55, 55,
    65, 65, 65, 65, 65, 65,
    75, 75, 75, 75, 75, 75, 75,
    85, 85, 85, 85, 85, 85, 85,
    95, 95, 95, 95, 95, 95, 95
],
    'viscosity': [1.464565, 1.456564, 1.440921, 1.422775, 1.396823, 1.396823, 1.369738, 1.332489, 1.332489,
1.304462,
```

```
    1.279099, 1.256326, 1.23607, 1.218257, 1.202816, 1.189672, 1.178752, 1.288276, 1.17449, 1.09637,
    1.057443, 1.051307, 1.081387, 1.106103, 1.190159, 1.263712, 1.14508, 1.139307, 1.12807, 1.115124,
    1.096786, 1.096786, 1.077903, 1.052447, 1.052447, 1.033783, 1.017357, 1.003096, 0.990929, 0.98078,
    0.972579, 0.96625, 0.961722, 1.009361, 0.93329, 0.888055, 0.874563, 0.883198, 0.926716, 0.953046,
    1.030962, 1.042406, 1.091306, 0.899873, 0.896197, 0.896022, 0.888577, 0.880091, 0.868261, 0.868261,
    0.856351, 0.840845, 0.840845, 0.836694, 0.830005, 0.82098, 0.813695, 0.808079, 0.805457, 0.804059,
    0.80156, 0.80051, 0.800836, 0.802216, 0.816259, 0.863245, 0.889025, 0.960507, 1.011849, 0.798627,
    0.754165, 0.735296, 0.74066, 0.758795, 0.807082, 0.831407, 0.894214, 0.902479, 0.934953, 0.717333,
    0.715096, 0.710829, 0.706066, 0.699636, 0.699636, 0.69347, 0.68607, 0.68607, 0.681518, 0.678356,
    0.676511, 0.67591, 0.67648, 0.678147, 0.680839, 0.684482, 0.644459, 0.625502, 0.626478, 0.644122,
    0.666486, 0.729572, 0.772166, 0.783041, 0.585849, 0.584918, 0.581435, 0.579301, 0.579301, 0.57651,
    0.57651, 0.576709, 0.577874, 0.579931, 0.582808, 0.586431, 0.590728, 0.595625, 0.601049, 0.493807,
    0.493876, 0.494118, 0.494588, 0.495642, 0.497274, 0.500552, 0.430357, 0.431932, 0.433912, 0.437047,
    0.440734, 0.446586, 0.381603, 0.382749, 0.385043, 0.387795, 0.391905, 0.396417, 0.402998, 0.338218,
    0.33944, 0.341838, 0.344624, 0.348603, 0.352712, 0.358177, 0.287827, 0.288819, 0.290705, 0.292789,
    0.29553, 0.298005, 0.300511

    ]
}

# Llamar a la función con los datos proporcionados
analyze_viscosity(data)
```