Análisis de Regresión Lineal

Cristian Ernesto Antonio Santiago

1 de abril de 2025

1. Introducción

La regresión lineal constituye una de las herramientas fundamentales en el análisis predictivo y la ciencia de datos. Este método estadístico nos permite modelar la relación entre una variable dependiente (en nuestro caso, el número de veces que se comparte un artículo en redes sociales) y una o más variables independientes (la cantidad de palabras del artículo). El modelo resultante adopta la forma de una ecuación lineal $y = \beta_0 + \beta_1 x$, donde β_0 representa el intercepto y β_1 la pendiente de la recta que mejor se ajusta a los datos.

En el contexto del marketing digital y la creación de contenido, comprender esta relación resulta particularmente valioso. Los creadores de contenido y editores web frecuentemente se enfrentan al dilema de determinar la extensión óptima de un artículo: mientras que artículos más extensos pueden contener información más completa, también requieren mayor tiempo de lectura y podrían afectar la retención de los lectores. Por otro lado, artículos demasiado breves podrían no proporcionar suficiente valor al lector. Este análisis busca cuantificar empíricamente cómo la longitud del artículo se relaciona con su viralidad en redes sociales, proporcionando insights basados en datos para apoyar decisiones editoriales.

2. Metodología

2.1. Preparación del Entorno de Análisis

El proceso comenzó con la configuración del entorno de programación, importando las bibliotecas esenciales para el análisis:

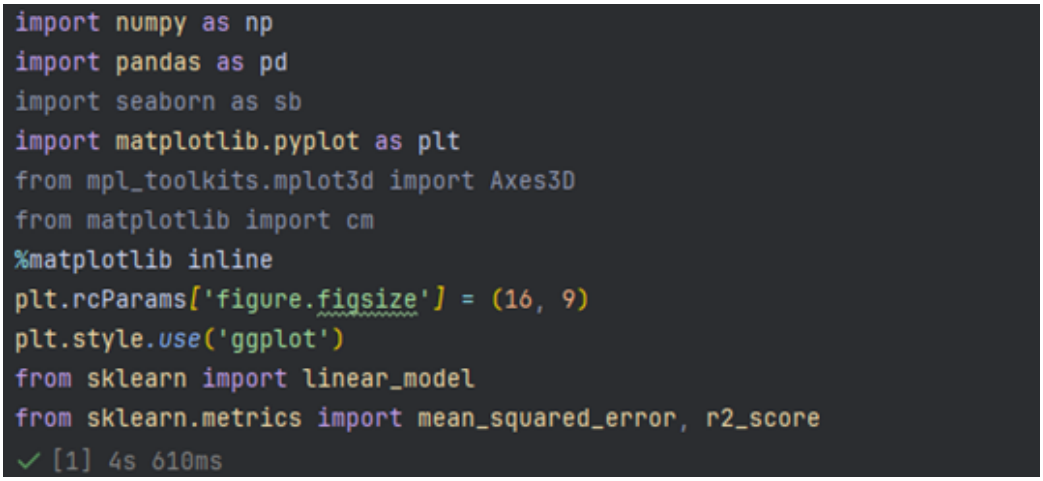

Figura 1: Importación de Librerias

El código comienza importando las bibliotecas esenciales para el análisis de datos y modelado estadístico. NumPy se utiliza para operaciones numéricas eficientes, mientras que Pandas permite manejar los datos en estructuras tabulares, facilitando su limpieza y exploración. Para la visualización, se carga Matplotlib junto con Seaborn, que mejoran la calidad estética de los gráficos. La línea `%matplotlib inline` asegura que todas las visualizaciones se muestren directamente en el entorno de trabajo, ideal para Jupyter Notebook.

La configuración de `plt.rcParams` establece un tamaño predeterminado de 16x9 pulgadas para las figuras, lo que garantiza claridad al presentar los resultados. El estilo `'ggplot'` se aplica para dar a los gráficos un diseño más pulido, con colores atractivos y fondos grises que mejoran la legibilidad.

Finalmente, se importan las herramientas específicas para el modelado: `linear_model` de Scikit-learn permite implementar la regresión lineal, mientras que `mean_squared_error` y `r2_score` son métricas clave para evaluar el rendimiento del modelo. El MSE cuantifica el error promedio entre las predicciones y los valores reales, mientras que el R^2 indica qué porcentaje de la variabilidad en los datos logra explicar el modelo.

2.2. Carga y exploración inicial de datos

```
#cargamos los datos de entrada
data = pd.read_csv("./articulos_ml.csv")
#veamos cuantas dimensiones y registros contiene
data.shape
[5]
(161, 8)
```

Figura 2: Carga de archivos

El código realiza la carga del conjunto de datos mediante la función `pd.read_csv()`, que lee archivos en formato CSV y los convierte automáticamente en un DataFrame de Pandas, la estructura fundamental para trabajar con datos tabulares en Python. La ruta `"./articulos_ml.csv"` indica que el archivo se encuentra en el mismo directorio donde se ejecuta el notebook. Es importante destacar que los DataFrames permiten manipular los datos de manera eficiente, ofreciendo operaciones para limpieza, filtrado y análisis.

Posteriormente, el comando `data.shape` muestra la dimensionalidad del DataFrame, devolviendo una tupla con dos valores: el primero (161) indica la cantidad de registros o filas, mientras que el segundo (8) representa el número de variables o columnas disponibles en el conjunto de datos. Esta salida es crucial para entender el alcance del análisis, confirmando que trabajaremos con 161 artículos diferentes, cada uno descrito por 8 características distintas.

```
#son 161 registros con 8 columnas. Veamos los primeros registros
data.head()
[3]
```

	Title	url	Word count	# of Links	# of comments	# Images video	E
0	What is Machine Learning and how d…	https://blog.signals.network/what-…	1888	1	2.0	2	
1	10 Companies Using Machine Learnin…	NaN	1742	9	NaN	9	
2	How Artificial Intelligence Is Rev…	NaN	962	0	0.0	1	
3	Obrain and the Blockchain of Artif…	NaN	1221	3	NaN	2	
4	Nasa finds entire solar system fil…	NaN	2039	1	104.0	4	

Figura 3: Primeras filas del dataset

El código `data.head()` es una herramienta fundamental para realizar una primera inspección de los datos cargados. Esta función muestra por defecto las primeras cinco filas del DataFrame, proporcionando una vista preliminar de la estructura y contenido del conjunto de datos. En este caso particu-

lar, podemos observar que efectivamente se visualizan 5 registros de los 161 existentes, cada uno con sus 8 columnas correspondientes.

```
# Ahora veamos algunas estadísticas de nuestros datos
data.describe()
```

[4]

8 rows × 8 rows × 6 cols

	Word count	# of Links	# of comments	# Images video	Elapsed days	# Shares
count	161.000000	161.000000	129.000000	161.000000	161.000000	161.000000
mean	1808.260870	9.739130	8.782946	3.670807	98.124224	27948.347826
std	1141.919385	47.271625	13.142822	3.418290	114.337535	43408.006839
min	250.000000	0.000000	0.000000	1.000000	1.000000	0.000000
25%	990.000000	3.000000	2.000000	1.000000	31.000000	2800.000000
50%	1674.000000	5.000000	6.000000	3.000000	62.000000	16458.000000
75%	2369.000000	7.000000	12.000000	5.000000	124.000000	35691.000000
max	8401.000000	600.000000	104.000000	22.000000	1002.000000	350000.000000

Figura 4: Estadísticas descriptivas del dataset

El comando `data.describe()` proporciona un resumen estadístico fundamental del conjunto de datos, ofreciendo una visión cuantitativa de la distribución de las variables numéricas. Esta función calcula automáticamente métricas clave para cada columna, permitiendo evaluar rápidamente el comportamiento de los datos.

Los resultados muestran estadísticas importantes para cada variable:

- **Word count**: Los artículos tienen en promedio 1,808 palabras, con una gran variabilidad (desviación estándar de 1,141). El artículo más extenso contiene 8,401 palabras, mientras que el más breve solo 250.
- **# Shares**: Variable objetivo del análisis, muestra una distribución asimétrica. El valor promedio es 27,948 compartidos, pero con una desviación estándar muy alta (43,408), indicando que algunos artículos son extremadamente virales comparados con la mayoría.
- **# of Links**: Presenta una media de 9 enlaces por artículo.

Las diferencias entre medianas y medias en varias columnas (especialmente en "# Sharesƨ "# of comments") revelan distribuciones sesgadas, donde unos pocos artículos con valores extremadamente altos elevan el promedio general. Esto es particularmente evidente en el caso de los compartidos en redes sociales, donde el valor máximo (350,000) es casi diez veces mayor que el percentil 75 (35,691).

2.3. Análisis de distribuciones mediante histogramas

```
# Visualizamos rápidamente las caracteristicas de entrada
data.drop(['Title','url', 'Elapsed days'],axis = 1).hist()
plt.show()
[7]
```

Figura 5: Visualizar histogramas

Tras haber examinado las estadísticas básicas con `describe()`, procedimos a visualizar las distribuciones de las variables numéricas. Para esto, ejecutamos la celda que genera los histogramas, eliminando primero las columnas no numéricas con `data.drop()` para enfocarnos en las variables relevantes.

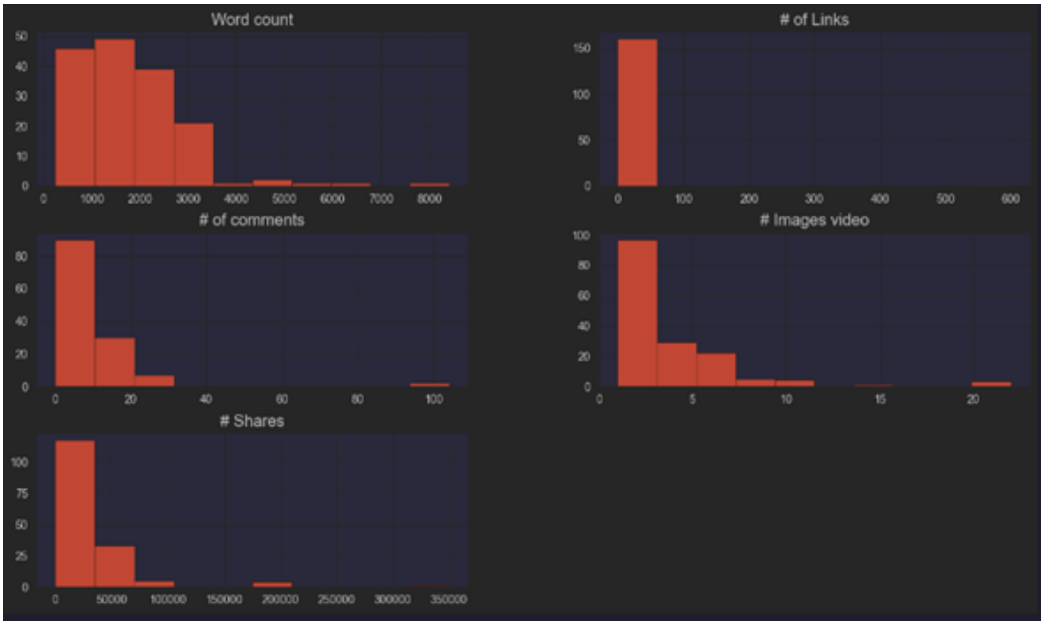

Figura 6: Histogramas de las variables numéricas

Los histogramas resultantes nos mostraron patrones clave en los datos. Observamos que la mayoría de artículos tenían entre 1,000 y 2,000 palabras, con algunos casos excepcionales más extensos. El número de compartidos presentó una distribución muy desigual, donde pocos artículos superaban ampliamente el promedio.

Esta visualización confirmó lo indicado por las estadísticas previas: la necesidad de procesar los valores extremos antes del modelado. Los gráficos también revelaron comportamientos interesantes en los comentarios y enlaces, mostrando concentraciones en valores específicos.

2.4. Filtrado y visualización de datos

```
# Vamos a RECORTAR los datos en la zona donde se concentran más los puntos
# esto es en el eje X: entre 0 y 3.500
# y en el eje Y: entre 0 y 80.000
filtered_data = data[(data['Word count'] <= 3500) & (data['# Shares'] <= 80000)]

colores=['orange','blue']
tamanios=[30,60]

f1 = filtered_data['Word count'].values
f2 = filtered_data['# Shares'].values

# Vamos a pintar en colores los puntos por debajo y por encima de la media de Cantidad
# de Palabras
asignar=[]
for index, row in filtered_data.iterrows():
    if(row['Word count']>1808):
        asignar.append(colores[0])
    else:
        asignar.append(colores[1])

plt.scatter(f1, f2, c=asignar, s=tamanios[0])
plt.show()
[8]
```

Figura 7: Visualzación de datos

Tras analizar las distribuciones, procedimos a limpiar el dataset eliminando valores extremos. El código aplica un filtro para conservar solo artículos con menos de 3,500 palabras y menos de 80,000 compartidos, lo que permite enfocarnos en la zona donde se concentra la mayoría de los datos. Este paso es crucial para mejorar la calidad del modelo, ya que reduce el efecto distorsionador de los outliers.

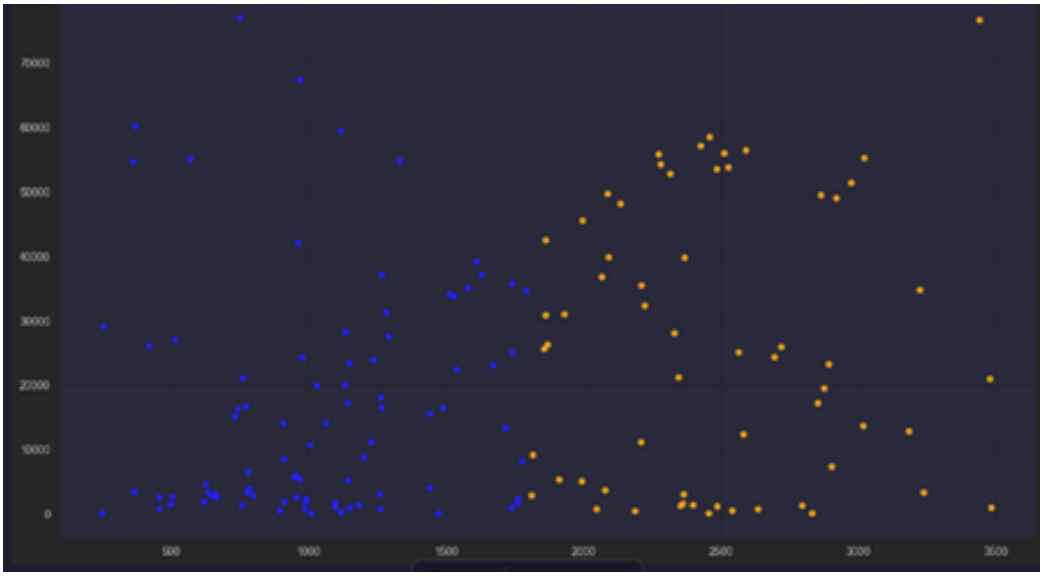

Figura 8: Relación entre palabras y compartidos

El gráfico de dispersión resultante muestra claramente la relación entre ambas variables. Los puntos se colorean según si superan (naranja) o no (azul) la media de palabras (1,808). Se observa que:

- La mayoría de puntos se agrupan en la zona inferior izquierda
- Existe una ligera tendencia positiva: artículos más largos tienden a tener más compartidos
- La dispersión aumenta conforme crece el número de palabras

2.5. Implementación del modelo de regresión lineal

```python
# Asignamos nuestra variable de entrada X para entrenamiento y las etiquetas Y.
dataX =filtered_data[["Word count"]]
X_train = np.array(dataX)
y_train = filtered_data['# Shares'].values

# Creamos el objeto de Regresión Linear
regr = linear_model.LinearRegression()

# Entrenamos nuestro modelo
regr.fit(X_train, y_train)

# Hacemos las predicciones que en definitiva una linea (en este caso, al ser 2D)
y_pred = regr.predict(X_train)

# Veamos los coeficienetes obtenidos, En nuestro caso, serán la Tangente
print('Coefficients: \n', regr.coef_)
# Este es el valor donde corta el eje Y (en X=0)
print('Independent term: \n', regr.intercept_)
# Error Cuadrado Medio
print("Mean squared error: %.2f" % mean_squared_error(y_train, y_pred))
# Puntaje de Varianza. El mejor puntaje es un 1.0
print('Variance score: %.2f' % r2_score(y_train, y_pred))
[9]
```

Figura 9: Construcción del modelo predictivo

Con los datos ya preparados, procedimos a construir el modelo predictivo. Primero asignamos las variables: utilizamos 'Word count' como variable independiente (X_train) y '# Shares' como variable objetivo (y_train). Estas se convirtieron a arrays de NumPy para optimizar el procesamiento. El modelo se creó instanciando `LinearRegression()` y se entrenó con el método `fit()`.

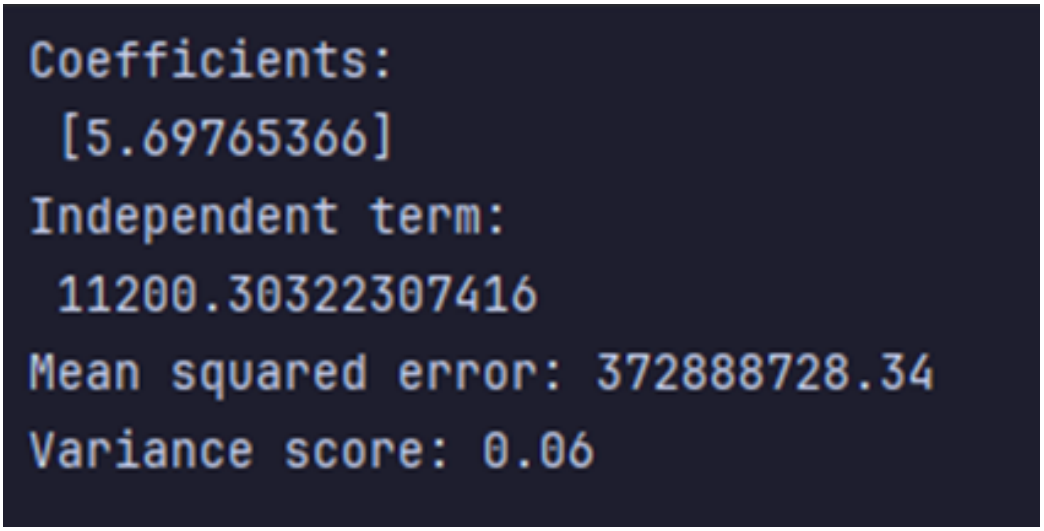

Figura 10: Resultados del modelo

Durante este proceso, el algoritmo calculó la relación óptima entre las palabras y los compartidos, generando una línea de mejor ajuste. Los resultados mostraron que:

- Por cada palabra adicional, se predicen aproximadamente 5.70 compartidos más (coeficiente)
- El intercepto en 11,200 sugiere que incluso artículos muy breves tendrían esta base de compartidos
- El MSE de 372,888,728 indica un error considerable en las predicciones
- El R^2 de 0.06 revela que solo el 6 % de la variación en compartidos se explica por la longitud

Los valores obtenidos confirman lo observado en gráficos anteriores: existe una relación positiva pero débil entre ambas variables. El bajo score R^2 sugiere que la longitud del artículo por sí sola no es un predictor fuerte de viralidad, lo que indica la necesidad de incorporar variables adicionales en análisis futuros.

3. Resultados

3.1. Visualización de los resultados del modelo

```
# Graficar los puntos de datos
plt.scatter(f1, f2, c=asignar, s=tamanios[0], alpha=0.6, edgecolors='w', linewidth=0.5)

# Graficar la línea de regresión
plt.plot(X_train, y_pred, color='red', linewidth=3, label='Línea de regresión')

# Títulos y etiquetas
plt.title('Relación Lineal', fontsize=14, pad=20)
plt.xlabel('Cantidad de Palabras', fontsize=12)
plt.ylabel('Compartidos en Redes', fontsize=12)

# Mostrar el gráfico
plt.show()
[17]
```

Figura 11: Generación del gráfico de regresión lineal

Para concluir el análisis, generamos un gráfico que combina los datos originales con la línea de regresión obtenida. El código utiliza `plt.scatter()` para mostrar los puntos filtrados anteriormente, manteniendo la codificación por colores (naranja para artículos sobre la media de palabras, azul para los demás). La transparencia (`alpha=0.6`) ayuda a visualizar mejor la densidad de puntos en zonas concurridas.

La línea de regresión se traza en rojo con `plt.plot()`, usando las predicciones del modelo (y_pred) contra los valores reales de X_train. Esta línea representa la relación lineal estimada: parte de aproximadamente 11,200 compartidos para artículos breves y tiene una pendiente suave de 5.7 compartidos por palabra adicional.

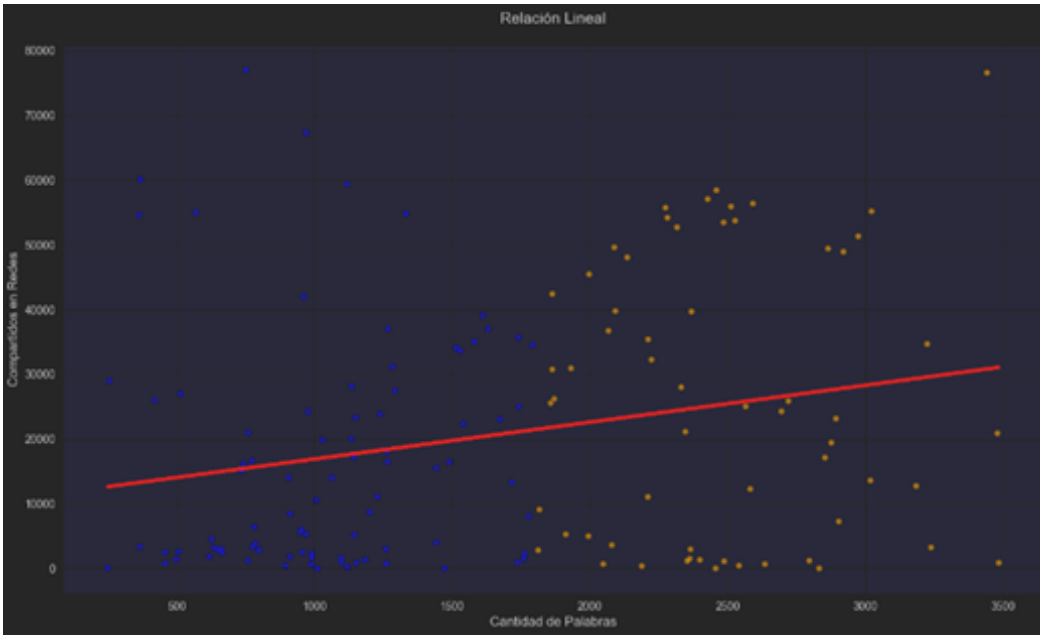

Figura 12: Modelo de regresión lineal superpuesto a los datos

3.2. Hallazgos clave

- La línea confirma la relación positiva pero débil entre variables
- La dispersión de puntos alrededor de la línea explica el bajo R^2 (0.06)
- Se observan varios artículos "sobreperformers"(puntos naranjas altos) que superan ampliamente las predicciones
- La mayoría de artículos cortos (¡1800 palabras) obtienen menos de 20,000 compartidos

El gráfico evidencia las limitaciones de usar solo la longitud del artículo como predictor. Si bien captura una tendencia general, la gran variabilidad residual sugiere que otros factores (calidad del contenido, tema, autor) influyen significativamente en el número de compartidos. Esto justifica el MSE elevado obtenido previamente.

3.3. Predicción con el modelo entrenado

```
#Vamos a comprobar:
# Quiero predecir cuántos "Shares" voy a obtener por un artículo con 2.000 palabras,
# según nuestro modelo, hacemos:
y_Dosmil = regr.predict([[2000]])
print(int(y_Dosmil))
[18]
 22595
```

Figura 13: Validación del modelo

Para validar el modelo en un caso práctico, realizamos una predicción puntual de cuántos compartidos obtendría un artículo de 2,000 palabras. El código utiliza el método `predict()` sobre el modelo entrenado (`regr`), pasando como entrada el valor 2000 en un array bidimensional (formato requerido por scikit-learn).

El resultado obtenido fue 22,595 compartidos, lo que significa que:

- Según el modelo, un artículo de esta longitud estaría ligeramente por encima del promedio observado
- La predicción sigue la tendencia lineal identificada (11,200 + 5.7*2000 = 22,600)
- Este valor debe interpretarse considerando el error cuadrático medio del modelo

4. Conclusión

El modelo de regresión lineal desarrollado identificó una relación estadísticamente significativa pero débil entre la longitud de los artículos y su desempeño en redes sociales. Si bien se confirmó que artículos más extensos tienden a recibir más compartidos (con un incremento promedio de 5.7 compartidos por palabra adicional), el bajo coeficiente de determinación (R^2 = 0.06) indica que la variable "Word count.explica solo una mínima parte de la variabilidad observada.

Los resultados obtenidos tienen importantes implicaciones prácticas:

- **Para creadores de contenido**: Sugieren que aumentar la longitud de los artículos podría tener un efecto positivo marginal en su viralidad,

pero que otros factores cualitativos (como el tema, calidad del contenido o autor) probablemente ejercen mayor influencia.

- **Para el modelo predictivo**: Evidencian la necesidad de incorporar variables adicionales que capturen dimensiones cualitativas del contenido, o explorar técnicas de modelado más sofisticadas que puedan manejar mejor la distribución asimétrica de los datos.

Las limitaciones principales encontradas fueron:

- La presencia de valores atípicos extremos que afectan la estabilidad del modelo
- La distribución no lineal de los datos alrededor de la línea de regresión
- La gran variabilidad residual no explicada por la variable predictora