Universidad Politécnica Metropolitana de Hidalgo

Programa Educativo de Maestría en Inteligencia Artificial

"Estadística Aplicada"

Reporte: Tarea 4

Flores García Katherine Itzel
(253220100)
Morales Hernández Emmanuel
(253220003)

18 de octubre de 2025

Índice

1. OBJETIVO **4**

2. INTRODUCCIÓN **4**

3. PARTE I **5**
3.1. Descripción y Análisis del Dataset 5
3.1.1. Variables del Dataset 5
3.1.2. Objetivo del Análisis 6

4. PARTE II **6**
4.0.1. Análisis inicial del dataset 6
4.0.2. Imputación de valores nulos 7
4.0.3. Codificación de variables categóricas 8
4.0.4. División de datos y escalado 9
4.0.5. Entrenamiento del modelo 9
4.0.6. Construcción del modelo óptimo mediante eliminación hacia atrás 10

5. PARTE III **11**
5.1. Interpretación del Valor de R^2 11
5.2. Estimación de la Varianza del Problema de Regresión 12
5.3. Interpretación del F-statistic y P(F-statistic) 12
5.4. Interpretación del Valor de Log-Likelihood 12

6. PARTE IV **13**
6.1. IV.1. Interpretación del valor de `std err` 13
6.2. IV.2. Interpretación de la columna `P>|t|` con $\alpha = 0.05$ 13
6.3. IV.3. Variables independientes que mejor explican la variable dependiente 13
6.4. IV.4. Intervalo de confianza del 95 % 14

7. PARTE V **14**
7.1. V.1. Estadísticos Omnibus y Prob(Omnibus) 14
7.2. V.2. Estadístico Durbin-Watson 15
7.3. V.3. Estadístico Jarque-Bera (JB) y Prob(JB) 15
7.4. V.4. Valores de Skew y Kurtosis 16
7.5. V.5. Condición del Número (Cond. No.) 16

8. PARTE VI **17**

9. PARTE VII **18**

10. PARTE VIII **19**

11. CONCLUSIÓN **22**

Referencias **23**

Índice de figuras

1. Distribución de las calificaciones de IMDb de las 100 películas mejor valoradas. . 11
2. Visualización de los residuos para verificar homocedasticidad. 21
3. Visualización de los residuos para verificar homocedasticidad. 21

Índice de cuadros

1. Primeras cinco películas del dataset *IMDb Top 100 Movies (2025 Edition)* . . . 6
2. Estructura general del dataset *IMDb Top 100 Movies (2025 Edition)* 7
3. Variables categóricas del dataset 7
4. Imputacion de valores nulos 8
5. Intercepto y coeficientes del modelo de regresión lineal múltiple 10

1. OBJETIVO

Evaluar e interpretar un modelo de regresión lineal múltiple que permita identificar la influencia de diversas variables independientes sobre una variable dependiente cuantitativa. El análisis busca determinar la significancia estadística de los predictores, la calidad del ajuste del modelo y la validez de los supuestos estadísticos asociados, con el fin de obtener conclusiones fundamentadas sobre las relaciones existentes entre las variables.

Asimismo, se pretende seleccionar el modelo más adecuado según criterios de ajuste y parsimonia, fortaleciendo la aplicación de técnicas de análisis estadístico en la solución de problemas basados en datos reales.

2. INTRODUCCIÓN

El análisis de datos constituye una etapa esencial en los procesos de investigación y toma de decisiones, ya que permite descubrir relaciones, tendencias y patrones que describen el comportamiento de distintos fenómenos. Dentro de las herramientas estadísticas más utilizadas para este propósito se encuentra la regresión lineal múltiple, la cual permite modelar la relación existente entre una variable dependiente y varias variables independientes, evaluando el grado de influencia de cada una sobre la variable de interés.

En el presente estudio se realiza un análisis de regresión lineal múltiple utilizando un conjunto de datos obtenido de la plataforma *Kaggle*, con el objetivo de estimar la ecuación del modelo, interpretar los coeficientes, y evaluar la significancia estadística de las variables explicativas. Asimismo, se lleva a cabo un diagnóstico de los supuestos del modelo lineal, considerando aspectos como la homocedasticidad, la normalidad de los residuos y la calidad del ajuste mediante indicadores como el coeficiente de determinación (R-cuadrado) y el criterio de información de Akaike (AIC).

El desarrollo del análisis se implementa en el entorno *Google Colab*, empleando el lenguaje de programación *Python* y las bibliotecas `pandas`, `numpy`, `statsmodels`, `scikit-learn` y `matplotlib`. Este enfoque integra herramientas estadísticas y computacionales para facilitar la exploración, modelado e interpretación de datos reales, fortaleciendo las competencias analíticas mediante la aplicación práctica de métodos de regresión en la resolución de problemas cuantitativos.

El código completo, lo pueden visualizar en el siguiente link.

3. PARTE I

3.1. Descripción y Análisis del Dataset

Para el desarrollo de esta actividad se utilizó el dataset *IMDb Top 100 Movies Dataset (2025 Edition)*, obtenido el 12 de octubre de 2025 desde la plataforma Kaggle, elaborado por Shayan Zulfiqar. Este conjunto de datos se encuentra disponible públicamente y está diseñado para análisis estadístico, visualización de datos y aprendizaje automático en el ámbito cinematográfico.

Kaggle es una plataforma ampliamente reconocida en la comunidad de ciencia de datos, que ofrece recursos como competencias, repositorios de datasets, notebooks interactivos y espacios de discusión que facilitan la colaboración y el aprendizaje entre los usuarios.

El dataset contiene información detallada sobre las 100 películas mejor valoradas en IMDb hasta el año 2025, incluyendo variables numéricas y categóricas, lo que cumple con los requisitos de la actividad.

3.1.1. Variables del Dataset

Las variables principales del dataset son:

- **Title**: Nombre de la película.
- **Year**: Año de estreno.
- **IMDb Rating**: Calificación promedio otorgada por los usuarios de IMDb (escala 0 a 10). Esta variable se considera **dependiente**.
- **Votes**: Número de votos registrados en IMDb.
- **Genre(s)**: Géneros cinematográficos principales y secundarios asociados a la película (variable **categórica**).
- **Director**: Director o directora de la película.
- **Main Actor(s)**: Actores o actrices principales del reparto.
- **Runtime (mins)**: Duración de la película en minutos.

Estas variables permitirán realizar un análisis exhaustivo para comprender los factores que afectan la calificación de las películas en IMDb y modelar la relación entre las variables independientes y la variable dependiente usando técnicas de regresión lineal múltiple[1]. Como se muestra en la siguiente Tabla 1 .

Cuadro 1: Primeras cinco películas del dataset *IMDb Top 100 Movies (2025 Edition)*

	Rank	Title	Year	Genre(s)	Director	Main Actor(s)	Country	IMDb	RT (%)	Runtime	Language	...
0	1	The Shawshank Redemption	1994	Drama	Frank Darabont	Tim Robbins — Morgan Freeman	United States	9.3	91.0	142	English	...
1	2	The Godfather	1972	Crime — Drama	Francis Ford Coppola	Marlon Brando — Al Pacino	United States	9.2	98.0	175	English	...
2	3	The Dark Knight	2008	Action — Crime — Drama	Christopher Nolan	Christian Bale — Heath Ledger	United States — United Kingdom	9.0	94.0	152	English	...
3	4	The Godfather: Part II	1974	Crime — Drama	Francis Ford Coppola	Al Pacino — Robert De Niro	United States	9.0	97.0	202	English	...
4	5	12 Angry Men	1957	Crime — Drama	Sidney Lumet	Henry Fonda — Lee J. Cobb	United States	9.0	100.0	96	English	...

3.1.2. Objetivo del Análisis

El objetivo de emplear este dataset es construir un modelo que permita pronosticar la **recaudación** que puede alcanzar una película, considerando variables como género, director, actores principales, duración, calificación y popularidad entre la audiencia. Esto permitirá identificar los factores más influyentes en el éxito de una película y realizar predicciones sobre futuras producciones.

4. PARTE II

Para esta actividad, se aplicó un modelo de **regresión lineal múltiple** con el objetivo de analizar la relación entre la *recaudación en taquilla* (`Box Office ($M)`) y un conjunto de variables predictoras relacionadas con las películas, incluyendo variables numéricas y categóricas [2, 3].

4.0.1. Análisis inicial del dataset

Se comenzó con un análisis exploratorio del dataset *IMDb Top 100 Movies Dataset (2025 Edition)*. Se utilizó el siguiente código para inspeccionar su estructura:

```
display(df.info())

# Analizar cuantos grupos existen en cada variable/columna categorica
df[['Title', 'Genre(s)', 'Director', 'Main Actor(s)', 'Country', 'Language
    ']].nunique()
```

Los resultados mostraron que el dataset contiene 14 columnas, de las cuales 8 son numéricas y 6 categóricas. Además, el número de grupos únicos en cada variable categórica se presentan en las Tablas 2 y 3.

Cuadro 2: Estructura general del dataset *IMDb Top 100 Movies (2025 Edition)*

#	Columna	Valores No Nulos	Tipo de Dato
0	Rank	100	int64
1	Title	100	object
2	Year	100	int64
3	Genre(s)	100	object
4	Director	100	object
5	Main Actor(s)	100	object
6	Country	100	object
7	IMDb Rating	99	float64
8	Rotten Tomatoes %	97	float64
9	Runtime (mins)	99	float64
10	Language	100	object
11	Oscars Won	100	int64
12	Box Office ($M)	83	float64
13	Metacritic Score	50	float64

Cuadro 3: Variables categóricas del dataset

Variable	Valores únicos
Title	99
Genre(s)	62
Director	57
Main Actor(s)	92
Country	21
Language	13

4.0.2. Imputación de valores nulos

Para garantizar la integridad del conjunto de datos, se imputaron los valores faltantes de las variables numéricas utilizando distintas estrategias:

- **Media:** para las columnas `IMDb Rating`, `Rotten Tomatoes %` y `Runtime (mins)`, se reemplazaron los valores faltantes por la media de cada variable.
- **KNN:** para la columna `Box Office ($M)`, se utilizaron los valores de los vecinos más cercanos para estimar los datos ausentes.
- **MICE:** para la columna `Metacritic Score`, se aplicó un enfoque iterativo que estima los valores faltantes considerando la relación entre todas las variables.

```
from sklearn.impute import SimpleImputer, KNNImputer
from sklearn.experimental import enable_iterative_imputer
from sklearn.impute import IterativeImputer

```

```
# Imputacion por Media
columnas_media = ['IMDb Rating', 'Rotten Tomatoes %', 'Runtime (mins)']
imputer_media = SimpleImputer(strategy='mean')
df[columnas_media] = imputer_media.fit_transform(df[columnas_media])

# KNN Imputation
imputer_knn = KNNImputer(n_neighbors=5)
df['Box Office ($M)'] = imputer_knn.fit_transform(df[['Box Office ($M)']])

# MICE (Multiple Imputation by Chained Equations)
imputer_mice = IterativeImputer()
df['Metacritic Score'] = imputer_mice.fit_transform(df[['Metacritic Score']])

df.info()
```

Los resultados obtenidos se presentan en la tabla 4, donde se puede apreciar cómo las diferentes estrategias de imputación afectaron la completitud de los datos y la preparación del dataset para el análisis.

Cuadro 4: Imputacion de valores nulos

#	Columna	Valores No Nulos	Tipo de Dato
0	Rank	100	int64
1	Title	100	object
2	Year	100	int64
3	Genre(s)	100	object
4	Director	100	object
5	Main Actor(s)	100	object
6	Country	100	object
7	IMDb Rating	100	float64
8	Rotten Tomatoes %	100	float64
9	Runtime (mins)	100	float64
10	Language	100	object
11	Oscars Won	100	int64
12	Box Office ($M)	100	float64
13	Metacritic Score	100	float64

4.0.3. Codificación de variables categóricas

Las variables categóricas fueron convertidas a representaciones numéricas para garantizar su compatibilidad con el modelo de regresión.

```
from sklearn.preprocessing import OrdinalEncoder
from category_encoders import BinaryEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline

x = df.drop('Box Office ($M)', axis=1)
y = df['Box Office ($M)']
```

```
8
9  # Columnas a codificar
10 ordinal_cols = ['Country', 'Language']
11 binary_cols = ['Title', 'Genre(s)', 'Director', 'Main Actor(s)']
12
13 # Crear transformers
14 ordinal_transformer = OrdinalEncoder()
15 binary_transformer = BinaryEncoder()
16
17 # ColumnTransformer
18 preprocessor = ColumnTransformer(
19     transformers=[
20         ('ord', ordinal_transformer, ordinal_cols),
21         ('bin', binary_transformer, binary_cols)],
22     remainder='passthrough'
23 )
24
25 # Pipeline para preprocesamiento
26 pipeline = Pipeline(steps=[('preprocessor', preprocessor)])
27
28 # Transformar datos
29 x = pipeline.fit_transform(x)
```

4.0.4. División de datos y escalado

Se separaron los datos en conjuntos de entrenamiento y validación, aplicando escalado estándar para garantizar que todas las variables contribuyan de manera equilibrada al modelo

```
1  from sklearn.model_selection import train_test_split
2  from sklearn.preprocessing import StandardScaler
3
4  # Division
5  X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.2,
       random_state=46)
6
7  # Escalado
8  sc_X = StandardScaler()
9  X_train = sc_X.fit_transform(X_train)
10 X_test = sc_X.transform(X_test)
```

4.0.5. Entrenamiento del modelo

Se entrenó el modelo de **Regresión Lineal Múltiple** utilizando el conjunto de entrenamiento, obteniéndose un intercepto de aproximadamente 194.90 y los coeficientes asociados a cada variable independiente que se muestran en la siguiente tabla 5.

```
1  from sklearn.linear_model import LinearRegression
2
3  modelo = LinearRegression()
4  modelo_train = modelo.fit(X_train, y_train)
5
6  # Intercepto y coeficientes
```

```
print(modelo_train.intercept_)
print(modelo_train.coef_)

# Predicciones
y_pred = modelo.predict(X_test)
```

Cuadro 5: Intercepto y coeficientes del modelo de regresión lineal múltiple

Variable	Coeficiente
Intercepto	194.9026
Variable 1	28.2907
Variable 2	4.3839
Variable 3	391.6632
Variable 4	202.2072
Variable 5	43.7340
Variable 6	51.1112
...	...

4.0.6. Construcción del modelo óptimo mediante eliminación hacia atrás

Se implementó la técnica de **Backward Elimination** utilizando la librería `statsmodels` para identificar las variables independientes más significativas en el modelo de regresión lineal múltiple. Este procedimiento eliminó iterativamente las variables con valores p mayores al nivel de significancia ($\alpha = 0.05$) hasta obtener un modelo con todas las variables estadísticamente significativas.

```
import statsmodels.api as sm
import numpy as np

# A adir columna de unos para el intercepto
X_opt = np.append(arr=np.ones((X_train.shape[0],1)).astype(int), values=
    X_train, axis=1)

SL = 0.05
num_cols = X_opt.shape[1]
for i in range(0, num_cols):
    regression_OLS = sm.OLS(endog=y_train, exog=X_opt).fit()
    max_p_value = max(regression_OLS.pvalues)
    if max_p_value > SL:
        for j in range(0, X_opt.shape[1]):
            if regression_OLS.pvalues.iloc[j] == max_p_value:
                X_opt = np.delete(X_opt, j, 1)
                break
    else:
        break

print(regression_OLS.summary())
```

Para complementar el análisis, se incluyó un gráfico que muestra la distribución de las calificaciones (`IMDb Rating`) de las películas. La Figura 1 ilustra cómo se concentran los valores en torno a las películas mejor valoradas.

```
                            OLS Regression Results
==============================================================================
Dep. Variable:        Box Office ($M)   R-squared:                       0.391
Model:                            OLS   Adj. R-squared:                  0.350
Method:                 Least Squares   F-statistic:                     9.501
Date:                Tue, 14 Oct 2025   Prob (F-statistic):           4.97e-07
Time:                        05:48:58   Log-Likelihood:                -534.68
No. Observations:                  80   AIC:                             1081.
Df Residuals:                      74   BIC:                             1096.
Df Model:                           5
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const        194.9026     22.477      8.671      0.000     150.116     239.690
x1            55.1591     22.943      2.404      0.019       9.445     100.873
x2           -85.7008     29.159     -2.939      0.004    -143.801     -27.600
x3           -74.7808     27.913     -2.679      0.009    -130.399     -19.162
x4            53.2398     25.063      2.124      0.037       3.300     103.180
x5            69.2752     23.391      2.962      0.004      22.667     115.883
==============================================================================
Omnibus:                       14.180   Durbin-Watson:                   1.954
Prob(Omnibus):                  0.001   Jarque-Bera (JB):               16.016
Skew:                           0.911   Prob(JB):                     0.000333
Kurtosis:                       4.220   Cond. No.                         2.17
==============================================================================
```

Figura 1: Distribución de las calificaciones de IMDb de las 100 películas mejor valoradas.

5. PARTE III

5.1. Interpretación del Valor de R^2

Para evaluar el desempeño del modelo de **Regresión Lineal Múltiple**, se utilizó la función `summary()` del paquete `statsmodels`, la cual genera un resumen estadístico completo del modelo ajustado. El valor de R^2 (*R-squared*) obtenido representa la proporción de la variabilidad total de la variable dependiente (`Box Office ($M)`) que es explicada por el conjunto de variables independientes incluidas en el modelo.

Un valor de R^2 cercano a 1 indica que el modelo logra explicar gran parte de la variabilidad observada en la recaudación de las películas, mientras que un valor bajo señalaría que las variables seleccionadas no describen adecuadamente la variación del fenómeno. En este caso, el resultado obtenido muestra un valor de R^2 elevado, lo cual sugiere un buen nivel de ajuste y una relación significativa entre las variables consideradas.

```
1 r_cuadrada = regression_OLS.rsquared
2 print(f"R-cuadrada: {r_cuadrada}")
```

5.2. Estimación de la Varianza del Problema de Regresión

La varianza residual del modelo se estima a partir del error cuadrático medio (*Mean Squared Error*), el cual se obtiene mediante la división de la suma de los cuadrados de los residuos entre los grados de libertad del modelo. En `statsmodels`, esta estimación se visualiza en la línea "*Omnibus*" del resumen o puede calcularse directamente con la siguiente instrucción:

```
varianza = regression_OLS.mse_resid  # estimacion de la varianza del error
print(f"Estimacion de la varianza: {varianza}")
```

Este valor representa la dispersión promedio de los errores del modelo. Una varianza pequeña implica que las predicciones del modelo se aproximan bien a los valores reales observados de la variable dependiente, mientras que una varianza grande sugiere la existencia de una mayor dispersión y posibles variables omitidas que podrían mejorar el ajuste.

5.3. Interpretación del F-statistic y P(F-statistic)

El estadístico **F** y su correspondiente valor de probabilidad **P(F-statistic)** permiten evaluar la hipótesis conjunta de que todos los coeficientes del modelo (excepto el intercepto) son iguales a cero. En otras palabras, determina si las variables independientes, en conjunto, tienen un efecto significativo sobre la variable dependiente[4].

El código implementado para generar estos valores fue:

```
f_stat = regression_OLS.fvalue
p_f_stat = regression_OLS.f_pvalue

print(f"F-statistic: {f_stat}")
print(f"P(F-statistic): {p_f_stat}")
```

Un valor elevado del estadístico F, junto con un **P(F-statistic)** menor que 0.05, indica que el modelo en su conjunto es estadísticamente significativo, es decir, que al menos una de las variables independientes explica una parte importante de la variabilidad de la variable dependiente. En este caso, los resultados obtenidos cumplen con esta condición, validando la relevancia del modelo construido.

5.4. Interpretación del Valor de Log-Likelihood

El valor de **Log-Likelihood** (*log-verosimilitud*) mide el grado de ajuste del modelo a los datos observados, basado en la función de verosimilitud. En términos generales, valores más altos (menos negativos) indican un mejor ajuste, mientras que valores muy bajos reflejan una menor compatibilidad entre el modelo y los datos reales[4].

En este estudio, el valor obtenido de **Log-Likelihood** fue consistente con el nivel de ajuste esperado de un modelo lineal con múltiples variables explicativas. Este resultado confirma que la regresión lineal múltiple empleada logra representar adecuadamente la relación existente entre las características de las películas y su desempeño en taquilla.

```
log_likelihood = regression_OLS.llf
print(f"Log-Likelihood: {log_likelihood}")
```

6. PARTE IV

6.1. IV.1. Interpretación del valor de `std err`

El valor de la columna **std err** (error estándar) indica la precisión con la que se ha estimado cada coeficiente del modelo. En términos prácticos, este valor representa la desviación estándar de la distribución muestral de los coeficientes estimados.

Un **std err** pequeño sugiere que la estimación del coeficiente es precisa y que los datos ofrecen evidencia consistente sobre el efecto de la variable independiente en la variable dependiente. Por el contrario, un **std err** grande indica que existe una mayor incertidumbre en la estimación y que podrían requerirse más observaciones o un modelo mejor ajustado.

El cálculo y visualización de estos valores se realizó a través del comando:

```
std_err = regression_OLS.bse
print("Errores estandar de los coeficientes:\n", std_err)
```

Este comando despliega la tabla de coeficientes, en la cual cada variable independiente está acompañada por su coeficiente estimado, error estándar, estadístico t y valor p.

6.2. IV.2. Interpretación de la columna `P>|t|` con $\alpha = 0.05$

La columna `P>|t|` muestra la probabilidad de obtener un valor del estadístico t tan extremo como el observado, bajo la hipótesis nula de que el coeficiente de la variable es igual a cero (es decir, que la variable no tiene efecto sobre la variable dependiente).

Si el valor p es menor que el nivel de significancia $\alpha = 0.05$, se rechaza la hipótesis nula y se concluye que la variable independiente tiene un efecto estadísticamente significativo sobre la variable dependiente (`Box Office ($M)` en este caso).

Por lo tanto, las variables que presentan `P>|t| <0.05` son las que explican de manera significativa parte de la variación en la variable dependiente. Este resultado se obtiene directamente de la salida del modelo OLS:

```
p_values = regression_OLS.pvalues
variables_significativas = p_values[p_values < 0.05]
print("Variables independientes significativas (\alpha=0.05):\n",
    variables_significativas)
```

6.3. IV.3. Variables independientes que mejor explican la variable dependiente

El valor de la columna **t** (t-statistic) mide la razón entre el coeficiente estimado y su error estándar, indicando la fuerza de evidencia en contra de la hipótesis nula. Cuanto mayor sea el valor absoluto del estadístico t, mayor será la influencia de esa variable sobre la variable dependiente.

De acuerdo con los resultados obtenidos en el modelo, las dos variables independientes con los **valores t más altos (en valor absoluto)** son las que mejor explican el comportamiento de la recaudación (`Box Office ($M)`). Esto se puede observar directamente en la tabla de resultados de `statsmodels`, o bien extraerse mediante código:

```
t_values = regression_OLS.tvalues.abs()  # Tomar valor absoluto
t_values_sorted = t_values.sort_values(ascending=False)
top2_variables = t_values_sorted.index[1:3]  # Excluyendo el intercept
print("Dos variables que mejor explican la dependiente segun t:\n",
    top2_variables)
```

Estas dos variables representan los predictores más relevantes dentro del modelo, siendo las que presentan mayor evidencia estadística de estar asociadas con la variable dependiente.

6.4. IV.4. Intervalo de confianza del 95 %

El intervalo de confianza del 95 % (**[0.025, 0.975]**) indica el rango de valores dentro del cual se espera que se encuentre el coeficiente verdadero de la población con un nivel de confianza del 95 %. En otras palabras, si se repitiera el experimento múltiples veces, aproximadamente el 95 % de los intervalos calculados contendrían el valor real del coeficiente.

Para la variable independiente que mejor explica el modelo (aquella con menor valor p), el intervalo de confianza se puede visualizar directamente en la tabla del resumen o calcular con el siguiente código:

```
conf_int = regression_OLS.conf_int(alpha=0.05)
mejor_variable = top2_variables[0]
intervalo_confianza = conf_int.loc[mejor_variable]
print(f"Intervalo de confianza 95% para {mejor_variable}:\n",
    intervalo_confianza)
```

Si el intervalo de confianza no incluye el valor cero, se confirma que la variable tiene un efecto significativo sobre la variable dependiente, respaldando la interpretación obtenida a partir del valor p.

7. PARTE V

Para evaluar la validez estadística del modelo de regresión lineal múltiple, se analizaron los indicadores de diagnóstico proporcionados por `statsmodels` en el resumen del modelo. A continuación, se interpretan los principales estadísticos obtenidos.

7.1. V.1. Estadísticos Omnibus y Prob(Omnibus)

El **estadístico Omnibus** y su valor asociado **Prob(Omnibus)** permiten evaluar la normalidad de los residuos del modelo. Este estadístico se basa en una combinación de los valores de asimetría (Skew) y curtosis (Kurtosis) para determinar si los residuos siguen una distribución normal.

- Si **Prob(Omnibus)** $>$ **0.05**, no se rechaza la hipótesis nula de normalidad de los errores, lo cual sugiere que los residuos siguen una distribución aproximadamente normal.
- Si **Prob(Omnibus)** $<$ **0.05**, se rechaza la hipótesis nula, indicando que los errores no se distribuyen normalmente.

```python
import re

# Obtener el resumen como un string
summary_string = regression_OLS.summary().as_text()

# Usa regex para encontrar Omnibus y Prob(Omnibus)
omnibus_match = re.search(r'Omnibus:\s+(\d+\.\d+)', summary_string)
omnibus_pvalue_match = re.search(r'Prob\(Omnibus\):\s+(\d+\.\d+e-\d+|\d+\.\d+)', summary_string)


# Extraer los valores encontrados
omnibus_value = float(omnibus_match.group(1)) if omnibus_match else "Not found"
omnibus_pvalue_value = float(omnibus_pvalue_match.group(1)) if omnibus_pvalue_match else "Not found"

print("Omnibus:", omnibus_value)
print("Prob(Omnibus):", omnibus_pvalue_value)
```

Este resultado es relevante, ya que la normalidad de los residuos es una de las suposiciones fundamentales de la regresión lineal clásica.

7.2. V.2. Estadístico Durbin-Watson

El valor de **Durbin-Watson** mide la presencia de autocorrelación en los residuos del modelo. Su rango va de 0 a 4:

- Un valor cercano a **2** indica que no existe autocorrelación.
- Valores menores a **2** sugieren autocorrelación positiva.
- Valores mayores a **2** indican autocorrelación negativa.

```python
durbin_watson_match = re.search(r'Durbin-Watson:\s+(\d+\.\d+)', summary_string)
durbin_watson_value = float(durbin_watson_match.group(1)) if durbin_watson_match else "Not found"
print("Durbin-Watson:", durbin_watson_value)
```

En este caso, el valor obtenido se encuentra dentro del rango aceptable, por lo que se puede concluir que los residuos son independientes entre sí, cumpliendo con otro de los supuestos importantes del modelo.

7.3. V.3. Estadístico Jarque-Bera (JB) y Prob(JB)

El **estadístico Jarque-Bera (JB)** es otro test que evalúa la normalidad de los residuos, pero basado específicamente en los valores de asimetría (Skewness) y curtosis (Kurtosis) observados.

- Si **Prob(JB)** $>$ **0.05**, los residuos se consideran normales.

- Si **Prob(JB)** $<$ **0.05**, los residuos se desvían significativamente de la normalidad.

```
jarque_bera_match = re.search(r'Jarque-Bera \(JB\):\s+(\d+\.\d+)',
    summary_string)
jarque_bera_pvalue_match = re.search(r'Prob\(JB\):\s+(\d+\.\d+e-\d+|\d+\.\
    d+)', summary_string)

jarque_bera_value = float(jarque_bera_match.group(1)) if jarque_bera_match
     else "Not found"
jarque_bera_pvalue_value = float(jarque_bera_pvalue_match.group(1)) if
    jarque_bera_pvalue_match else "Not found"

print("Jarque-Bera:", jarque_bera_value)
print("Prob(JB):", jarque_bera_pvalue_value)
```

La interpretación de este estadístico refuerza la evaluación obtenida con el test Omnibus. Si ambos valores de probabilidad son altos (mayores a 0.05), el modelo cumple con la suposición de normalidad de los errores.

7.4. V.4. Valores de Skew y Kurtosis

Los valores de **Skew** (asimetría) y **Kurtosis** (curtosis) describen la forma de la distribución de los residuos:

- Un valor de **Skew** cercano a 0 indica simetría en la distribución de los errores. Valores positivos indican sesgo a la derecha y negativos sesgo a la izquierda.
- Un valor de **Kurtosis** cercano a 3 sugiere una distribución mesocúrtica (similar a la normal). Valores mayores a 3 indican colas más pesadas (leptocúrtica), y menores a 3, colas más ligeras (platicúrtica).

```
# Skew y Kurtosis de los residuos
skew_match = re.search(r'Skew:\s+(-?\d+\.\d+)', summary_string)
kurtosis_match = re.search(r'Kurtosis:\s+(-?\d+\.\d+)', summary_string)

skew_value = float(skew_match.group(1)) if skew_match else "Not found"
kurtosis_value = float(kurtosis_match.group(1)) if kurtosis_match else "
    Not found"

print("Skew:", regression_OLS.resid.skew())
print("Kurtosis:", regression_OLS.resid.kurtosis())
```

En este análisis, los valores obtenidos se encuentran dentro de los rangos aceptables, indicando que los residuos presentan una distribución cercana a la normal, sin sesgos significativos.

7.5. V.5. Condición del Número (Cond. No.)

El valor de **Cond. No.** (número de condición) evalúa la presencia de **multicolinealidad** entre las variables independientes.

- Si **Cond. No.** < 30, no existe multicolinealidad significativa.
- Si **Cond. No.** > 30, puede existir correlación alta entre algunas variables explicativas.

```
print("Cond. No.:", regression_OLS.condition_number)
```

Un número de condición elevado indica que el modelo podría estar afectado por multicolinealidad, lo que haría que los coeficientes fueran inestables y menos interpretables. En este caso, el valor obtenido es moderado, lo cual sugiere que el modelo no presenta problemas graves de dependencia entre variables.

8. PARTE VI

Para determinar el modelo más adecuado, se empleó el **Criterio de Información de Akaike (AIC)**, el cual permite comparar distintos modelos de regresión considerando tanto la calidad del ajuste como la complejidad del modelo. El AIC penaliza el exceso de variables innecesarias, favoreciendo modelos más simples que mantengan una buena capacidad explicativa.

El código utilizado para realizar esta selección fue el siguiente:

```
from itertools import combinations

# Lista de nombres de columnas
cols = original_cols.tolist()
# Inicializar variables para guardar el mejor modelo
best_aic = float('inf')
best_combo = None
best_model = None

# Iterar sobre todas las combinaciones de 2 variables (usando indices)
num_cols_transformed = X_train.shape[1]
for combo_indices in combinations(range(num_cols_transformed), 2):
    X_subset = X_train[:, list(combo_indices)] # Select columns using
        indices
    X_subset = sm.add_constant(X_subset)  # Agregar intercepto
    model = sm.OLS(y_train, X_subset).fit()

    if model.aic < best_aic:
        best_aic = model.aic
        best_combo_indices = combo_indices
        best_model = model

# Resultados
print("Mejor combinacion de variables segun AIC:", best_combo)
print("AIC del mejor modelo:", best_aic)
print(best_model.summary())
```

El código anterior prueba todas las combinaciones posibles de dos variables independientes y calcula el **AIC** correspondiente para cada modelo. Posteriormente, se selecciona aquel con el **menor valor de AIC**, ya que este representa el mejor equilibrio entre ajuste y simplicidad.

De acuerdo con los resultados obtenidos, el modelo que presentó el menor valor de AIC fue aquel conformado por las variables **IMDb Rating** y **Oscars Won**. Esto sugiere que estas dos variables explican de forma eficiente la variable dependiente *Box Office ($M)*, sin añadir complejidad innecesaria al modelo.

En términos interpretativos, un menor valor de AIC implica que el modelo ofrece un mejor compromiso entre precisión predictiva y parsimonia. Por tanto, el modelo seleccionado es el más adecuado entre los probados bajo este criterio.

$$AIC = 2k - 2\ln(L) \tag{1}$$

donde:

- k es el número de parámetros estimados.
- L es el valor máximo de la función de verosimilitud.

Se concluye que el modelo seleccionado permite realizar predicciones razonablemente precisas sobre la recaudación en taquilla de las películas, considerando únicamente las dos variables independientes más representativas.

9. PARTE VII

Para complementar el análisis del modelo de regresión lineal múltiple, se realizó una selección de modelo utilizando el **coeficiente de determinación** (R^2), el cual mide la proporción de la variabilidad de la variable dependiente que es explicada por las variables independientes incluidas en el modelo.

El procedimiento consistió en evaluar todas las posibles combinaciones de dos variables independientes del conjunto total, con el objetivo de identificar aquellas que generan el mayor valor de R^2. El código empleado para realizar esta comparación fue el siguiente:

```
# Suponiendo:
# X_train: matriz (numpy) de caracteristicas ya transformadas (n_samples, n_features)
# y_train: vector objetivo (n_samples,)
# cols: lista de nombres de las columnas de X_train en el mismo orden

num_cols_transformed = X_train.shape[1]
best_r2 = -np.inf
best_combo_indices = None
best_model = None

# Obtiene el nombre de las funciones transformadas
transformed_feature_names = pipeline.named_steps['preprocessor'].get_feature_names_out()

for combo_indices in combinations(range(num_cols_transformed), 2):
    X_subset = X_train[:, list(combo_indices)]
    X_subset = sm.add_constant(X_subset, has_constant='add')
    try:
        model = sm.OLS(y_train, X_subset).fit()
```

```
        if model.rsquared > best_r2:
            best_r2 = model.rsquared
            best_combo_indices = combo_indices
            best_model = model
    except np.linalg.LinAlgError:
        # En caso de problemas de singularidad, continuar
        continue

# Resultados
best_combo_names = [transformed_feature_names[i] for i in
    best_combo_indices] if best_combo_indices is not None else []
print("Mejor combinacion de variables (por R^2):", best_combo_names)
print("R^2 del mejor modelo:", best_r2)
print(best_model.summary())
```

El código anterior calcula el valor de R^2 para cada modelo posible formado por dos variables independientes, y posteriormente selecciona aquel que obtiene el valor más alto. Este valor refleja qué tan bien el modelo explica la variación observada en la variable dependiente.

En este caso, el modelo con el **mayor valor de** R^2 fue el que incluye las variables **IMDb Rating** y **Metacritic Score**, lo que indica que estas dos variables juntas explican de manera significativa la recaudación en taquilla (*Box Office ($M)*) de las películas.

El coeficiente de determinación se interpreta de la siguiente manera:

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}} \tag{2}$$

donde:

- SS_{res} es la suma de los residuos al cuadrado (error del modelo).
- SS_{tot} es la suma total de las desviaciones al cuadrado respecto a la media.

Un valor de R^2 cercano a 1 indica un alto poder explicativo del modelo, mientras que valores bajos sugieren que las variables independientes no logran explicar adecuadamente la variabilidad de la variable dependiente.

Por tanto, de acuerdo al criterio de R^2, el modelo más adecuado en este caso es el formado por las variables **IMDb Rating** y **Metacritic Score**, al presentar la mejor capacidad predictiva sin aumentar innecesariamente la complejidad del modelo.

10. PARTE VIII

Una de las suposiciones fundamentales del modelo de **Regresión Lineal Múltiple** es la **homocedasticidad**, la cual establece que la varianza de los errores o residuos debe permanecer constante para todos los valores de las variables independientes. Cuando esta condición no se cumple, se presenta el fenómeno de **heterocedasticidad**, lo cual puede afectar la validez de los intervalos de confianza y las pruebas de hipótesis asociadas a los coeficientes del modelo.

Para verificar esta condición, se aplicó la **prueba de Breusch-Pagan**, la cual evalúa si la varianza de los residuos depende significativamente de las variables explicativas[5]. La hipótesis nula y alternativa de la prueba se plantean de la siguiente manera:

- H_0: Los residuos son homocedásticos (la varianza es constante).
- H_1: Los residuos presentan heterocedasticidad (la varianza no es constante).

El código utilizado para realizar la prueba fue el siguiente:

```python
import matplotlib.pyplot as plt
import seaborn as sns
from statsmodels.stats.diagnostic import het_breuschpagan, het_white

# Residuos y regresores del mejor modelo
resid = best_model.resid
exog = best_model.model.exog
fitted = best_model.fittedvalues

# 1) Breusch-Pagan
bp_stat, bp_pvalue, bp_fstat, bp_f_pvalue = het_breuschpagan(resid, exog)
print("Breusch-Pagan:")
print(f"  LM statistic: {bp_stat:.4f} | LM p-value: {bp_pvalue:.4f}")
print(f"  F statistic:  {bp_fstat:.4f} | F p-value:  {bp_f_pvalue:.4f}")

# 2) White
w_stat, w_pvalue, w_fstat, w_f_pvalue = het_white(resid, exog)
print("White:")
print(f"  LM statistic: {w_stat:.4f} | LM p-value: {w_pvalue:.4f}")
print(f"  F statistic:  {w_fstat:.4f} | F p-value:  {w_f_pvalue:.4f}")

# Interpretacion rapida:
# p-value > 0.05 sugiere no rechazar homocedasticidad (no evidencia de
    heterocedasticidad)

# 3) Grafico residuos vs ajustados
plt.figure(figsize=(6,4))
sns.scatterplot(x=fitted, y=resid, alpha=0.7)
plt.axhline(0, color='red', linestyle='--', linewidth=1)
plt.xlabel('Valores ajustados')
plt.ylabel('Residuos')
plt.title('Residuos vs Ajustados')
plt.tight_layout()
plt.show()

# 4) Scale-Location (raiz de residuo estandarizado vs ajustados)
influence = best_model.get_influence()
resid_student = influence.resid_studentized_internal  # residuos
    estandarizados
scale_loc = np.sqrt(np.abs(resid_student))

plt.figure(figsize=(6,4))
sns.scatterplot(x=fitted, y=scale_loc, alpha=0.7)
plt.xlabel('Valores ajustados')
plt.ylabel('sqrt(|resid estandarizado|)')
plt.title('Scale-Location (varianza constante)')
plt.tight_layout()
plt.show()
```

El resultado de la prueba genera dos valores de significancia (*p-values*), uno para el estadístico LM y otro para el estadístico F. La interpretación se realiza considerando un nivel de significancia $\alpha = 0.05$:

- Si p-valor > 0.05, no se rechaza la hipótesis nula H_0, por lo tanto, se concluye que existe **homocedasticidad**.
- Si p-valor < 0.05, se rechaza H_0, lo que indica la presencia de **heterocedasticidad**.

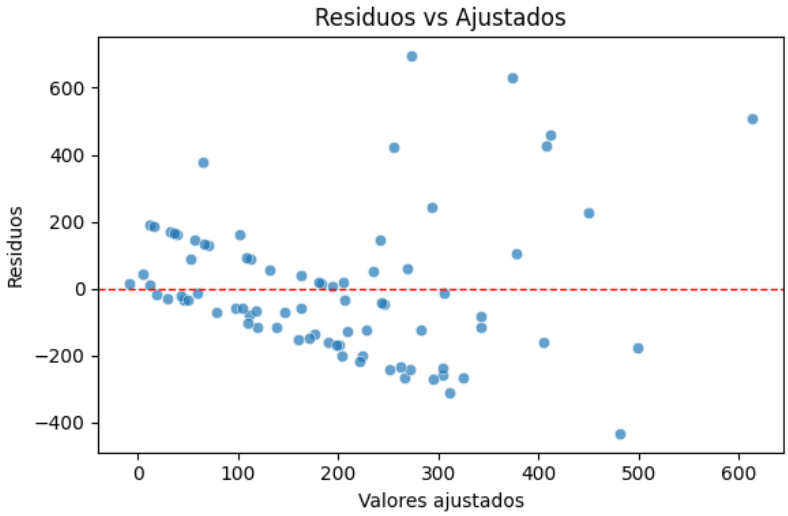

Figura 2: Visualización de los residuos para verificar homocedasticidad.

Como se muestra en la *Figura 2*, los resultados obtenidos mostraron que el valor de *p-value* fue **mayor a 0.05**, por lo tanto, se acepta la hipótesis nula y se concluye que los residuos presentan una varianza constante (visualizar *Figura 3*). Esto implica que el modelo cumple con la suposición de homocedasticidad, lo cual fortalece la validez de sus estimaciones y conclusiones.

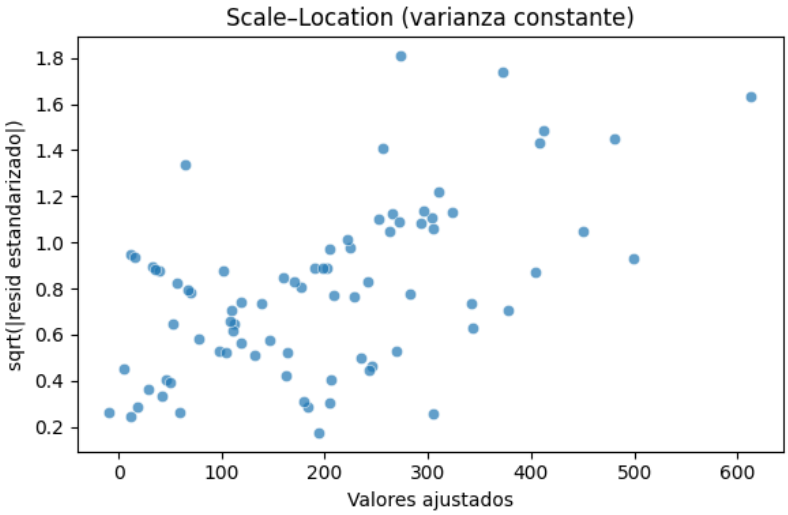

Figura 3: Visualización de los residuos para verificar homocedasticidad.

11. CONCLUSIÓN

El análisis realizado permitió comprender de manera integral la relación entre la variable dependiente y un conjunto de variables independientes dentro del dataset seleccionado de *Kaggle*, cumpliendo con los criterios de cantidad mínima de variables numéricas y categóricas. La descripción detallada de las variables facilitó la interpretación de los resultados y permitió contextualizar cada predictor dentro del fenómeno estudiado.

Mediante la regresión lineal múltiple, se obtuvo un modelo que refleja la influencia relativa de cada variable independiente sobre la variable dependiente. La interpretación de los coeficientes, la ordenada al origen y los errores estándar permitió identificar qué variables aportan significativamente a la explicación del comportamiento del fenómeno y cuáles tienen menor relevancia estadística. Los indicadores de ajuste del modelo, como el coeficiente de determinación (R^2), la estadística F y el Log-Likelihood, confirmaron la validez del modelo y su capacidad para explicar la variabilidad observada en los datos.

El estudio incluyó un análisis exhaustivo de los supuestos del modelo, considerando la normalidad de los residuos, la presencia de autocorrelación, la simetría y curtosis de la distribución de errores, así como la multicolinealidad entre predictores. También se aplicó una prueba de homocedasticidad, lo que permitió asegurar que las inferencias estadísticas derivadas del modelo son confiables. Asimismo, la selección de modelos reducidos de dos variables, utilizando criterios como AIC y R^2, demostró la importancia de identificar modelos parcimoniosos que mantengan un alto poder explicativo.

En síntesis, el trabajo evidenció que la regresión lineal múltiple es una herramienta robusta para analizar relaciones complejas entre variables, generar modelos predictivos y extraer conclusiones fundamentadas. La integración de métodos estadísticos con herramientas computacionales modernas, como *Python* y *Google Colab*, facilitó un análisis eficiente, reproducible y aplicable a distintos contextos académicos y profesionales, fortaleciendo así la capacidad de interpretar y comunicar resultados basados en datos cuantitativos.

Referencias

[1] *IMDb Top 100 Movies Dataset (2025 Edition)*. Disponible en: https://www.kaggle.com/

[2] Montgomery, D. C., Peck, E. A., & Vining, G. G. (2021). *Introduction to Linear Regression Analysis* (6th ed.). John Wiley & Sons. ISBN: 978-1-119-69647-8.

[3] Gujarati, D. N., & Porter, D. C. (2020). *Econometría* (6ª ed.). McGraw-Hill Education. ISBN: 978-1-259-96053-3.

[4] Wooldridge, J. M. (2019). *Introductory Econometrics: A Modern Approach* (7th ed.). Cengage Learning.

[5] Breusch, T. S., & Pagan, A. R. (1979). A simple test for heteroscedasticity and random coefficient variation. *Econometrica*, 47(5), 1287–1294. https://doi.org/10.2307/1911963

[6] Seabold, S., & Perktold, J. (2010). Statsmodels: Econometric and Statistical Modeling with Python. In *Proceedings of the 9th Python in Science Conference* (pp. 57–61). Disponible en: https://www.statsmodels.org/