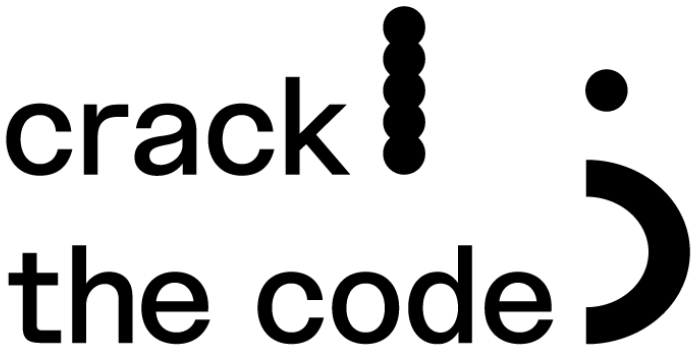


Crack The Code

Docente

Maguiña Diaz, Alyssa Nicole

Curso

Inteligencia Artificial – SIC 5083

Actividad

Proyecto 1

Análisis de Películas y Series de TV en IMDB

Estudiante

- Suclle Luque, Daniel Alex

LIMA- PERÚ
2024

Contenido

1. Introducción 4
   1.1. Objetivos 4
2. Carga y Limpieza de datos 5
   2.1. Columna Año de lanzamiento 6
   2.2. Columna certificación 8
   2.3. Duración 9
   2.4. Puntuación de Metacritic 10
   2.5. Recaudación bruta 11
3. Análisis Exploratorio (EDA) 13
   3.1. Gráfico 1 – Distribución de años de lanzamiento 14
   3.2. Gráfico 2 – Distribución de duración 15
   3.3. Gráfico 3 – Distribución de calificación en IMDB 16
   3.4. Gráfico 4 – Distribución de géneros de las películas 17
   3.5. Gráfico 5 – Top 20 actores más frecuentes 18
   3.6. Gráfico 6 – Películas más taquilleras 19
   3.7. Pregunta 1 - ¿Qué directores tienden a generar mayores ingresos en la taquilla? 20
   3.8. Pregunta 2 - ¿Qué actores están asociados con películas que generan mayores ingresos? 21
   3.9. Pregunta 3 - ¿Qué directores tienden a recibir más votos en IMDB? 22
   3.10. Pregunta 4 - ¿Qué actores están asociados con películas que reciben más votos? 23
   3.11. Pregunta 5 - Analizar la preferencia de actores por diferentes géneros cinematográficos. 24
4. Modelo Matemático: Regresión lineal múltiple 24
   4.1. Regresión línea simple 25
   4.2. Regresión lineal multiple 25
   4.3. Obtención de las derivadas parciales 25
      4.3.1. Función de costo 26
      4.3.2. Derivada Parcial con respeto a ***Wj*** 26
      4.3.3. Derivada Parcial con respecto a ***b*** 27
      4.3.4. Implementación en el algoritmo de descenso de gradiente 27

5. Conclusiones... 33

1. Introducción

En la era digital actual, el análisis de datos se ha convertido en una herramienta fundamental para entender y predecir comportamientos en diversas áreas, incluyendo la industria del entretenimiento. Con la vasta cantidad de información disponible en bases de datos como IMDb (Internet Movie Database), es posible obtener insights valiosos sobre las películas y series de televisión, así como sobre las preferencias del público y las tendencias del mercado.

Este proyecto, titulado **"Análisis de Películas y Series de TV en IMDb"**, tiene como objetivo explorar y analizar datos de películas y series de televisión utilizando técnicas de análisis de datos y visualización. Utilizando Python y sus poderosas librerías como pandas y matplotlib, llevamos a cabo un análisis exhaustivo que incluye la carga, limpieza y visualización de datos, así como la implementación de modelos de regresión para identificar patrones y relaciones significativas en la industria cinematográfica.

A través de este análisis, buscamos responder preguntas clave como: ¿Qué directores y actores tienden a generar mayores ingresos en taquilla? ¿Qué géneros de películas son más populares entre el público? Y, ¿qué factores influyen en las votaciones de IMDb? Las respuestas a estas preguntas pueden ofrecer perspectivas valiosas para productores, directores y profesionales del marketing en la toma de decisiones informadas.

1.1. Objetivos

- **Carga y Limpieza de Datos:**
  - Cargar el dataset de IMDb utilizando la librería panda.
  - Realizar la limpieza de datos, incluyendo el manejo de valores nulos y el ajuste de tipos de datos.
- **Análisis Exploratorio de Datos (EDA):**
  - Explorar las tendencias, distribuciones y relaciones entre diferentes variables del dataset.
  - Identificar patrones y correlaciones significativas entre directores, actores, géneros y el éxito en taquilla.
- **Visualización de Datos:**
  - Crear gráficos y visualizaciones que permitan una comprensión clara y visual de los datos analizados.
  - Utilizar la librería matplotlib para generar visualizaciones atractivas y informativas.
- **Implementación de Modelos de Regresión:**

    - Implementar y entrenar modelos de regresión lineal utilizando técnicas de descenso de gradiente.
    - Evaluar el rendimiento de los modelos y ajustar hiperparámetros para mejorar la precisión de las predicciones.
- **Generación de Insights y Conclusiones:**
    - Interpretar los resultados del análisis y los modelos de regresión para extraer conclusiones relevantes.
    - Proporcionar recomendaciones basadas en los insights obtenidos, dirigidas a profesionales de la industria del entretenimiento.

2. Carga y Limpieza de datos

La carga y limpieza de datos es una fase crucial en cualquier proyecto de análisis de datos, ya que asegura la calidad y la integridad de los datos utilizados. En este proyecto, utilizamos la librería pandas de Python para gestionar estas tareas, dada su eficiencia y facilidad de uso para la manipulación de datos.

En esta etapa, nos aseguraremos de que nuestros datos estén listos para el análisis. Realizaremos las siguientes tareas

Manejo de valores nulos:

Identificaremos columnas con valores faltantes y decidiremos si eliminamos filas o imputamos valores para los datos faltantes.

Ajuste de tipos de datos:

Verificaremos que las columnas tengan los tipos de datos correctos. Por ejemplo, convertiremos fechas a objetos datatetime si es necesario.

Información de cuantos valores nulos existe por columnas y tipo de datos de nuestro dataframe antes de la limpieza de datos.

```
# Obtenemos informacion de cuantos valores nulos existen por columnas y su tipo de dato
df.info()
#17-18-19 imputacion de datos
```

✓ 0.0s

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 15 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Titulo                  1000 non-null   object
 1   Año de lanzamiento      1000 non-null   object
 2   Certificacion           899 non-null    object
 3   Duracion                1000 non-null   object
 4   Genero                  1000 non-null   object
 5   Calificacion en IMDB    1000 non-null   float64
 6   Resumen                 1000 non-null   object
 7   Puntuacion en Metacritic  843 non-null    float64
 8   Director                1000 non-null   object
 9   Actor Principal 1       1000 non-null   object
 10  Actor Principal 2       1000 non-null   object
 11  Actor Principal 3       1000 non-null   object
 12  Actor Principal 4       1000 non-null   object
 13  Numero de votos         1000 non-null   int64
 14  Recaudacion bruta       831 non-null    object
dtypes: float64(2), int64(1), object(12)
memory usage: 117.3+ KB
```

Analizando los resultados podemos inferir que la columna de recaudación bruta es la que posee mayores valores nulos (169).

Una posible solución sería usar el método dropna() en nuestro dataframe pero nos dejaria con tan solo 734 registros de películas en nuestro dataframe, lo cualno es ideal. Para evitar esta perdida significativa de datos, analizaremos cada columna y tomaremos medidas adecuadas para solucionar este problema de manera efectiva.

2.1. Columna Año de lanzamiento

Analizando esta columna llegamos a la conclusión de que se debe cambiar el tipo de dato de **object** a **int,** para visualizar si algún valor causaría problemas al momento de la conversaion del tipo de dato usaremos el método unique().

```
['1994' '1972' '2008' '1974' '1957' '2003' '1993' '2010' '1999' '2001'
 '1966' '2002' '1990' '1980' '1975' '2020' '2019' '2014' '1998' '1997'
 '1995' '1991' '1977' '1962' '1954' '1946' '2011' '2006' '2000' '1988'
 '1985' '1968' '1960' '1942' '1936' '1931' '2018' '2017' '2016' '2012'
 '2009' '2007' '1984' '1981' '1979' '1971' '1963' '1964' '1950' '1940'
 '2013' '2005' '2004' '1992' '1987' '1986' '1983' '1976' '1973' '1965'
 '1959' '1958' '1952' '1948' '1944' '1941' '1927' '1921' '2015' '1996'
 '1989' '1978' '1961' '1955' '1953' '1925' '1924' '1982' '1967' '1951'
 '1949' '1939' '1937' '1934' '1928' '1926' '1920' '1970' '1969' '1956'
 '1947' '1945' '1930' '1938' '1935' '1933' '1932' '1922' '1943' 'PG']
```

Observamos que tenemos un valor “PG” dentro de columna esto traería problemas al cambiar el tipo de dato, analizamos el registro donde el año de lanzamiento es “PG”.

```
#Observamos que existe un valor PG que causaria problemas al momento de convertir los datos a int
#Imprimimos el registro que contiene PG como Año de lanzamiento
registro = df[df['Año de lanzamiento'] == 'PG']
registro
```

	Titulo	Año de lanzamiento	Certificacion	Duracion	Genero	Calificacion en IMDB	Resumen	Puntuacion en Metacritic	Director	Actor Principal 1	Actor Principal 2	Actor Principal 3	Actor Principal 4	Numero de votos	Recaudacion bruta
966	Apollo 13	PG	U	140 min	Adventure, Drama, History	7.6	NASA must devise a strategy to return Apollo 1...	77.0	Ron Howard	Tom Hanks	Bill Paxton	Kevin Bacon	Gary Sinise	269197	173,837,933

Como se trata de un solo registro podemos investigar el año de estreno y reemplazarlo.

```
df.loc[df['Año de lanzamiento'] == 'PG', 'Año de lanzamiento'] = '1970'
#Comprobamos la moficicacion
registro = df[df['Titulo'] == 'Apollo 13']

registro
```

[143] ✓ 0.0s Python

	Titulo	Año de lanzamiento	Certificacion	Duracion	Genero	Calificacion en IMDB	Resumen	Puntuacion en Metacritic	Director	Actor Principal 1	Actor Principal 2	Actor Principal 3	Actor Principal 4	Numero de votos	Recaudacion bruta
966	Apollo 13	1970	U	140 min	Adventure, Drama, History	7.6	NASA must devise a strategy to return Apollo 1...	77.0	Ron Howard	Tom Hanks	Bill Paxton	Kevin Bacon	Gary Sinise	269197	173,837,933

Luego de esto verificamos de nuevo los datos de la columna Año de lanzamiento con el método unique().

```
#Corroboramos si no existe algun otro valor que cause problemas al convertir el tipo de dato
print(df['Año de lanzamiento'].unique())
```

[144] ✓ 0.0s

```
['1994' '1972' '2008' '1974' '1957' '2003' '1993' '2010' '1999' '2001'
 '1966' '2002' '1990' '1980' '1975' '2020' '2019' '2014' '1998' '1997'
 '1995' '1991' '1977' '1962' '1954' '1946' '2011' '2006' '2000' '1988'
 '1985' '1968' '1960' '1942' '1936' '1931' '2018' '2017' '2016' '2012'
 '2009' '2007' '1984' '1981' '1979' '1971' '1963' '1964' '1950' '1940'
 '2013' '2005' '2004' '1992' '1987' '1986' '1983' '1976' '1973' '1965'
 '1959' '1958' '1952' '1948' '1944' '1941' '1927' '1921' '2015' '1996'
 '1989' '1978' '1961' '1955' '1953' '1925' '1924' '1982' '1967' '1951'
 '1949' '1939' '1937' '1934' '1928' '1926' '1920' '1970' '1969' '1956'
 '1947' '1945' '1930' '1938' '1935' '1933' '1932' '1922' '1943']
```

No existe algún otro dato que de problemas al cambiar el tipo de dato.

```
# Cambiamos el tipo de dato de object a tipo entero

df['Año de lanzamiento'] = df['Año de lanzamiento'].astype(int)

df['Año de lanzamiento']
```

✓ 0.0s

```
0      1994
1      1972
2      2008
3      1974
4      1957
       ...
995    1961
996    1956
997    1953
998    1944
999    1935
Name: Año de lanzamiento, Length: 1000, dtype: int64
```

Luego de ejecutar el código para cambiar el tipo de dato observamos que ahora es del tipo **int**.

2.2. Columna certificación

Analizaremos la columna certificación dentro de esta columna existe valores nulos por cual realizaremos la imputación de datos utilizando la moda, es decir, el valor mas frecuente en esta columna. Esto se hace para evitar la pérdida de información y mantener la consistencia de datos.

```
print("Valores nulos antes de la imputación:")
print(df['Certificacion'].isnull().sum())

moda_certificacion = df['Certificacion'].mode()[0]

print(f"Moda utilizada para la imputación: {moda_certificacion}")

df['Certificacion'] = df['Certificacion'].fillna(moda_certificacion)


print("Valores nulos después de la imputación:")
print(df['Certificacion'].isnull().sum())

df['Certificacion']
```

[146] ✓ 0.0s

```
Valores nulos antes de la imputación:
101
Moda utilizada para la imputación: U
Valores nulos después de la imputación:
0

0          A
1          A
2         UA
3          A
4          U
        ...
995        A
996        G
997    Passed
998        U
999        U
Name: Certificacion, Length: 1000, dtype: object
```

Existe 101 valores nulos.

Moda utilizada para la imputación: U.

2.3. Duración

En esta columna se cambiará el tipo de dato de **Object** a **int.** Además, quitaremos el string “min” que poseen los datos para no tener problemas al cambiar el tipo de dato.

```
print(df['Duracion'].unique())
df['Duracion'] = df['Duracion'].str.replace(' min', '').astype(int)
```

✓ 0.0s

```
['142 min' '175 min' '152 min' '202 min' '96 min' '201 min' '154 min'
 '195 min' '148 min' '139 min' '178 min' '161 min' '179 min' '136 min'
 '146 min' '124 min' '133 min' '160 min' '132 min' '153 min' '169 min'
 '130 min' '125 min' '189 min' '116 min' '127 min' '118 min' '121 min'
 '207 min' '122 min' '106 min' '112 min' '151 min' '150 min' '155 min'
 '119 min' '110 min' '88 min' '137 min' '89 min' '165 min' '109 min'
 '102 min' '87 min' '126 min' '147 min' '117 min' '181 min' '149 min'
 '105 min' '164 min' '170 min' '98 min' '101 min' '113 min' '134 min'
 '229 min' '115 min' '143 min' '95 min' '104 min' '123 min' '131 min'
 '108 min' '81 min' '99 min' '114 min' '129 min' '228 min' '128 min'
 '103 min' '107 min' '68 min' '138 min' '156 min' '167 min' '163 min'
 '186 min' '321 min' '135 min' '140 min' '180 min' '158 min' '210 min'
 '86 min' '162 min' '177 min' '204 min' '91 min' '172 min' '45 min'
 '145 min' '100 min' '196 min' '93 min' '120 min' '92 min' '144 min'
 '80 min' '183 min' '111 min' '141 min' '224 min' '171 min' '188 min'
 '94 min' '185 min' '85 min' '205 min' '212 min' '238 min' '72 min'
 '67 min' '76 min' '159 min' '83 min' '90 min' '84 min' '191 min'
 '197 min' '174 min' '97 min' '75 min' '157 min' '209 min' '82 min'
 '220 min' '64 min' '184 min' '168 min' '166 min' '192 min' '194 min'
 '193 min' '69 min' '70 min' '242 min' '79 min' '71 min' '78 min']
```

Luego de la ejecución del código obtenemos.

```
print(df['Duracion'].unique())
```

✓ 0.0s

```
[142 175 152 202  96 201 154 195 148 139 178 161 179 136 146 124 133 160
 132 153 169 130 125 189 116 127 118 121 207 122 106 112 151 150 155 119
 110  88 137  89 165 109 102  87 126 147 117 181 149 105 164 170  98 101
 113 134 229 115 143  95 104 123 131 108  81  99 114 129 228 128 103 107
  68 138 156 167 163 186 321 135 140 180 158 210  86 162 177 204  91 172
  45 145 100 196  93 120  92 144  80 183 111 141 224 171 188  94 185  85
 205 212 238  72  67  76 159  83  90  84 191 197 174  97  75 157 209  82
 220  64 184 168 166 192 194 193  69  70 242  79  71  78]
```

2.4. Puntuación de Metacritic

Imputaremos los valores nulos en esta columna utilizando la mediana. La mediana es una medida robusta que no se ve afectada por valores atípicos, lo que la hace adecuada para este tipo de datos.

```python
# Verificar los valores nulos antes de la imputación
print("Valores nulos antes de la imputación:")
print(df['Puntuacion en Metacritic'].isnull().sum())

# Imputar valores nulos en la columna 'Puntuación en Metacritic' con la mediana
median_puntuacionM = df['Puntuacion en Metacritic'].median()
df['Puntuacion en Metacritic'] = df['Puntuacion en Metacritic'].fillna(median_puntuacionM)

# Verificar los valores nulos después de la imputación
print("Valores nulos después de la imputación:")
print(df['Puntuacion en Metacritic'].isnull().sum())

# Mostrar la mediana utilizada para la imputación
print(f"Mediana utilizada para la imputación: {median_puntuacionM}")

print(df['Puntuacion en Metacritic'])
```

✓ 0.0s

```
Valores nulos antes de la imputación:
157
Valores nulos después de la imputación:
0
Mediana utilizada para la imputación: 79.0
0       80.0
1      100.0
2       84.0
3       90.0
4       96.0
       ...
995     76.0
996     84.0
997     85.0
998     78.0
999     93.0
Name: Puntuacion en Metacritic, Length: 1000, dtype: float64
```

Verificamos que antes de realizar la imputación existían 157 valores nulos, la mediana que se utilizó para la imputación es 79.

2.5. Recaudación bruta

En esta columna realizaremos un cambio de tipo de dato de **Object** a **Float**. Además, realizaremos imputación de datos utilizando la mediana, esto es esencial para garantizar que los datos sean consistentes.

```
print(df['Recaudacion bruta'].unique())
df['Recaudacion bruta'] = pd.to_numeric(df['Recaudacion bruta'].str.replace(',', ''), errors='coerce')
```

✓ 0.0s

```
['28,341,469' '134,966,411' '534,858,444' '57,300,000' '4,360,000'
 '377,845,905' '107,928,762' '96,898,818' '292,576,195' '37,030,102'
 '315,544,750' '330,252,182' '6,100,000' '342,551,365' '171,479,930'
 '46,836,394' '290,475,067' '112,000,000' nan '53,367,844' '188,020,017'
 '7,563,397' '10,055,859' '216,540,909' '136,801,374' '57,598,247'
 '100,125,643' '130,742,922' '322,740,140' '269,061' '335,451,311'
 '13,092,000' '13,182,281' '53,089,891' '132,384,315' '32,572,577'
 '187,705,427' '6,719,864' '23,341,568' '19,501,238' '422,783,777'
 '204,843,350' '11,990,401' '210,609,762' '5,321,508' '32,000,000'
 '1,024,560' '163,245' '19,181' '1,661,096' '5,017,246' '12,391,761'
 '190,241,310' '858,373,000' '678,815,482' '209,726,015' '162,805,434'
 '448,139,099' '6,532,908' '1,223,869' '223,808,164' '11,286,112'
 '707,481' '25,544,867' '2,375,308' '248,159,971' '44,017,374'
 '83,471,511' '78,900,000' '275,902' '8,175,000' '36,764,313' '288,475'
 '159,227,644' '1,373,943' '687,185' '7,098,492' '6,857,096' '120,540,719'
 '34,400,301' '33,225,499' '30,328,156' '3,635,482' '130,096,601'
 '138,433,435' '933,933' '191,796,233' '75,600,000' '2,832,029'
 '46,357,676' '85,160,248' '51,973,029' '45,598,982' '309,125,409'
 '11,487,676' '28,262,574' '159,600,000' '6,207,725' '56,954,992'
 '15,000,000' '44,824,144' '18,600,000' '13,275,000' '3,200,000'
 '8,819,028' '55,240' '332,930' '5,720,000' '1,585,634' '28,877'
 '1,236,166' '5,450,000' '898,575' '4,186,168' '85,080,171' '54,513,740'
 '342,370' '20,186,659' '739,478' '1,429,534' '144,501' '1,626,289'
 '7,461' '39,567' '6,391,436' '13,657,115' '128,012,934' '293,004,164'
 '116,900,694' '1,113,541' '40,222,514' '37,634,615' '415,004,880'
 ...
 '285,761,243' '66,666,062' '92,823,600' '111,543,479' '78,756,177'
 '49,530,288' '65,207,127' '2,150,000' '119,285,432' '12,465,371'
 '22,490,039' '76,657,000' '43,000,000' '35,000,000' '132,088,635'
 '959,000' '696,690' '1,378,435' '141,843,612' '13,780,024' '30,500,000']
```

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

Con este código logramos eliminar los string "," para no perjudicar la conversión. Además, si existiera algún valor que no se pueda cambiar a dato float se reemplazara con nan(nulo).

```
# Imputar valores nulos en la columna 'Recaudación bruta' con la mediana
medianda_recaudacion = df['Recaudacion bruta'].median()
df['Recaudacion bruta'] = df['Recaudacion bruta'].fillna(medianda_recaudacion)

# Verificar los valores nulos después de la imputación
print("Valores nulos después de la imputación:")
print(df['Recaudacion bruta'].isnull().sum())

# Mostrar la mediana utilizada para la imputación
print(f"Mediana utilizada para la imputación: {medianda_recaudacion}")
```

✓ 0.0s

```
Valores nulos después de la imputación:
0
Mediana utilizada para la imputación: 23530892.0
```

Con el código imputamos lo valores nulos utilizando la mediana.

Luego de finalizar de analizar las columnas que nos servirán para nuestro análisis utilizamos el método info().

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 15 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Titulo                  1000 non-null   object
 1   Año de lanzamiento      1000 non-null   int64
 2   Certificacion           1000 non-null   object
 3   Duracion                1000 non-null   int64
 4   Genero                  1000 non-null   object
 5   Calificacion en IMDB    1000 non-null   float64
 6   Resumen                 1000 non-null   object
 7   Puntuacion en Metacritic  1000 non-null   float64
 8   Director                1000 non-null   object
 9   Actor Principal 1       1000 non-null   object
 10  Actor Principal 2       1000 non-null   object
 11  Actor Principal 3       1000 non-null   object
 12  Actor Principal 4       1000 non-null   object
 13  Numero de votos         1000 non-null   int64
 14  Recaudacion bruta       1000 non-null   float64
dtypes: float64(3), int64(3), object(9)
memory usage: 117.3+ KB
```

Logramos mantener las 1000 filas utilizando la imputación de datos. Además, cambiar el tipo de variable de object a numéricas para poder realizar operaciones.

3. Análisis Exploratorio (EDA)

En esta sección se presenta un análisis exploratorio de los datos con varios gráficos y visualizaciones para entender mejor las características del conjunto de datos de películas.

```
df.describe()
```
✓ 0.0s

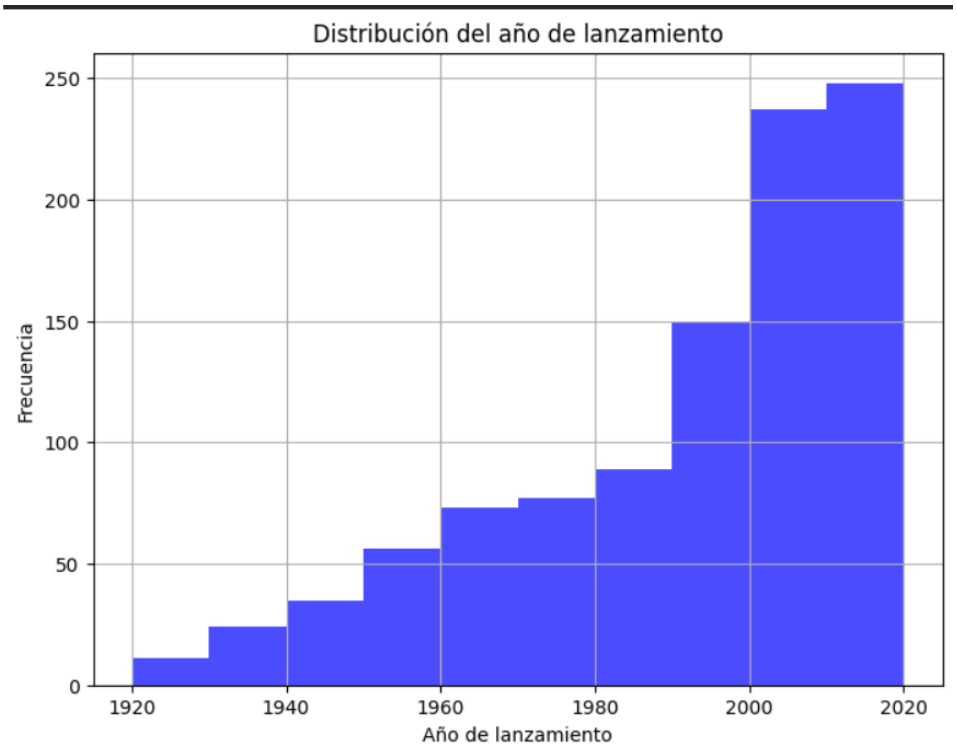
	Año de lanzamiento	Duracion	Calificacion en IMDB	Puntuacion en Metacritic	Numero de votos	Recaudacion bruta
count	1000.000000	1000.000000	1000.000000	1000.000000	1.000000e+03	1.000000e+03
mean	1991.196000	122.891000	7.949300	78.133000	2.736929e+05	6.051360e+07
std	23.295026	28.093671	0.275491	11.368225	3.273727e+05	1.014192e+08
min	1920.000000	45.000000	7.600000	28.000000	2.508800e+04	1.305000e+03
25%	1976.000000	103.000000	7.700000	72.000000	5.552625e+04	5.012919e+06
50%	1999.000000	119.000000	7.900000	79.000000	1.385485e+05	2.353089e+07
75%	2009.000000	137.000000	8.100000	85.250000	3.741612e+05	6.153989e+07
max	2020.000000	321.000000	9.300000	100.000000	2.343110e+06	9.366622e+08

```
df.head()
```
✓ 0.0s Python

	Titulo	Año de lanzamiento	Certificacion	Duracion	Genero	Calificacion en IMDB	Resumen	Puntuacion en Metacritic	Director	Actor Principal 1	Actor Principal 2	Actor Principal 3	Actor Principal 4	Numero de votos	Recaudacion bruta
0	The Shawshank Redemption	1994	A	142	Drama	9.3	Two imprisoned men bond over a number of years...	80.0	Frank Darabont	Tim Robbins	Morgan Freeman	Bob Gunton	William Sadler	2343110	28341469.0
1	The Godfather	1972	A	175	Crime, Drama	9.2	An organized crime dynasty's aging patriarch t...	100.0	Francis Ford Coppola	Marlon Brando	Al Pacino	James Caan	Diane Keaton	1620367	134966411.0
2	The Dark Knight	2008	UA	152	Action, Crime, Drama	9.0	When the menace known as the Joker wreaks havo...	84.0	Christopher Nolan	Christian Bale	Heath Ledger	Aaron Eckhart	Michael Caine	2303232	534858444.0
3	The Godfather: Part II	1974	A	202	Crime, Drama	9.0	The early life and career of Vito Corleone in ...	90.0	Francis Ford Coppola	Al Pacino	Robert De Niro	Robert Duvall	Diane Keaton	1129952	57300000.0
4	12 Angry Men	1957	U	96	Crime, Drama	9.0	A jury holdout attempts to prevent a miscarria...	96.0	Sidney Lumet	Henry Fonda	Lee J. Cobb	Martin Balsam	John Fiedler	689845	4360000.0

3.1. Gráfico 1 – Distribución de años de lanzamiento

Se muestra un gráfico que ilustra la distribución de los años de lanzamiento de las películas en el conjunto de datos.

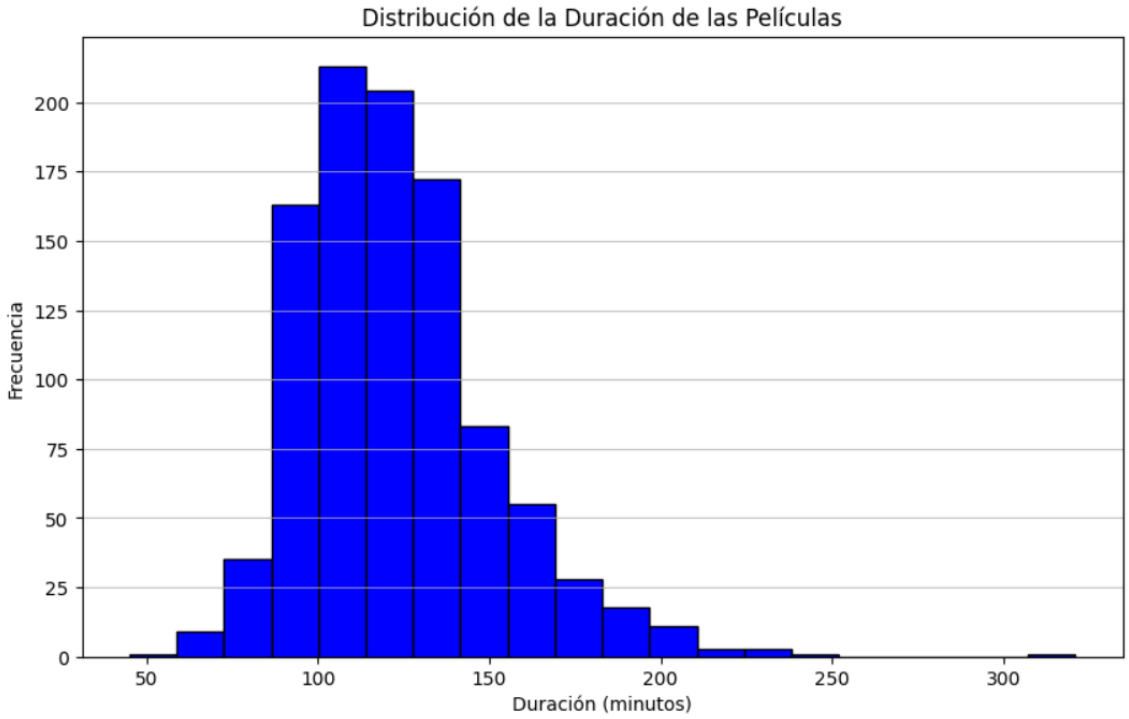


Podemos observar que:

- La mayoría de los peliculas se lanzaron entre el año 2000 y 2020.
- El número de peliculas lanzados aumenta gradualmente a medida que nos acercamos a la actualidad.

El gráfico indica que el conjunto de datos contiene más películas lanzadas recientemente y que hay un crecimiento constante en el número de lanzamientos con el paso del tiempo.

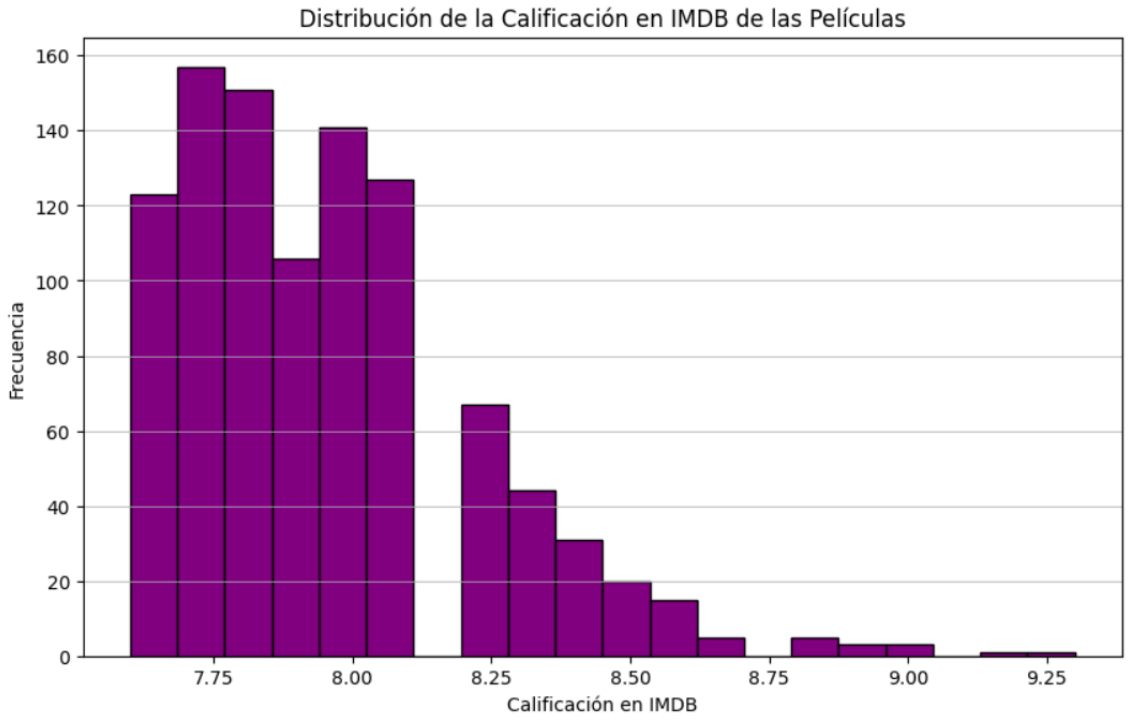
3.2. Gráfico 2 – Distribución de duración



El histograma muestra la distribución de la duración de las películas. Se puede observar que la mayoría de las películas tienen una duración de entre 90 y 150 minutos. Hay una pequeña cantidad de películas con una duración inferior a 90 minutos y una cantidad aún menor de películas con una duración superior a 150 minutos.

La distribución es ligeramente sesgada hacia la derecha, lo que significa que hay más películas con una duración más corta que con una duración más larga. Esto es típico de la distribución de la duración de las películas, ya que las películas más largas suelen ser más costosas de producir y pueden ser más difíciles de comercializar.

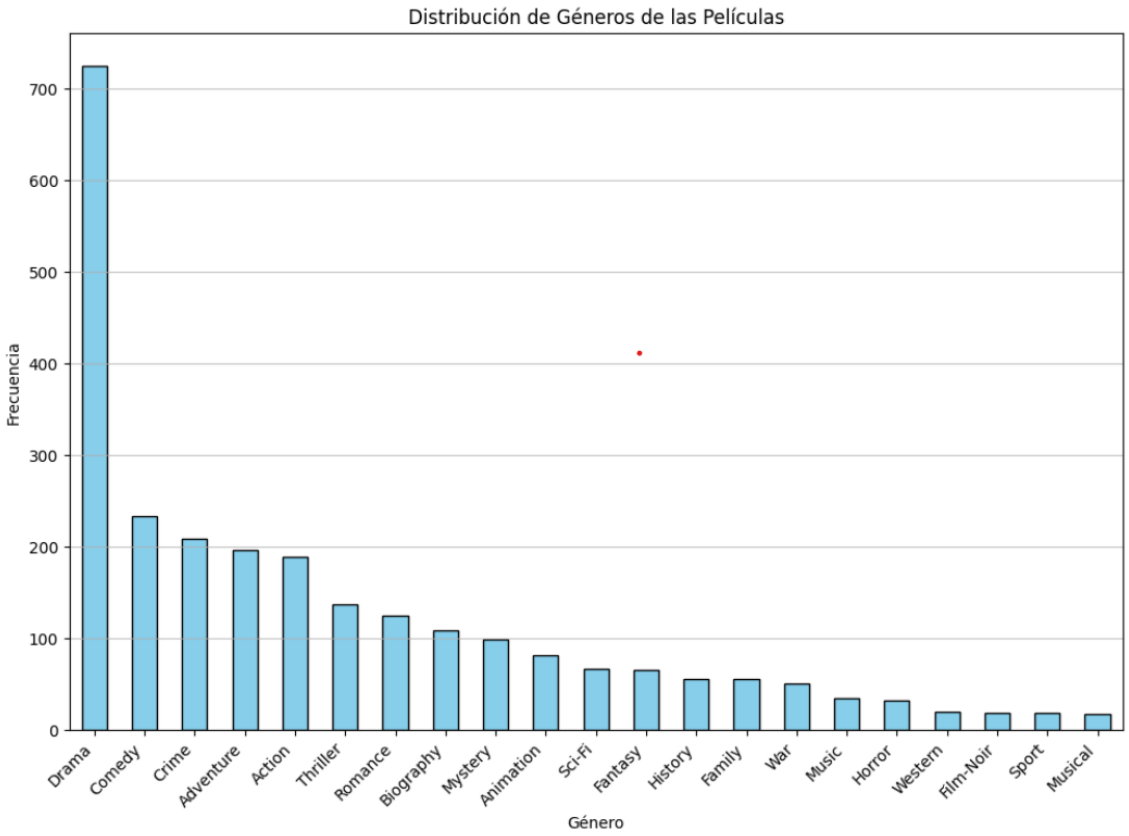
3.3. Gráfico 3 – Distribución de calificación en IMDB



El gráfico muestra la distribución de las calificaciones de las películas en IMDb. El eje horizontal representa la calificación en IMDb, mientras que el eje vertical representa la frecuencia. Como puede ver, la mayoría de las películas tienen una calificación entre 7.5 y 8.5. La frecuencia de las películas con una calificación mayor que 8.5 disminuye rápidamente. Esto significa que es más probable que una película tenga una calificación entre 7.5 y 8.5 que una calificación entre 8.5 y 9.5.

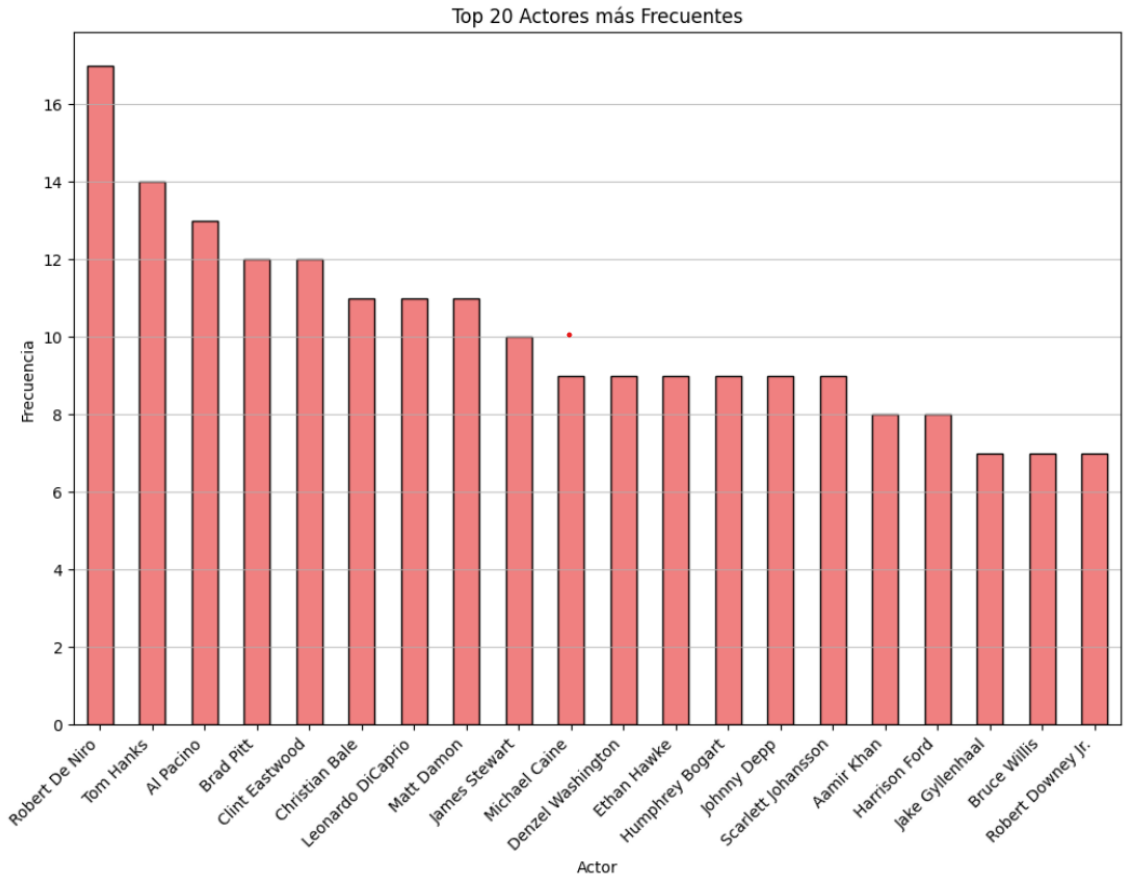
En otras palabras, la distribución de las calificaciones en IMDb es asimétrica, con una cola hacia la derecha. Esto sugiere que la mayoría de las películas tienen una calificación moderada, mientras que hay un número relativamente pequeño de películas con una calificación extremadamente alta.

3.4. Gráfico 4 – Distribución de géneros de las películas



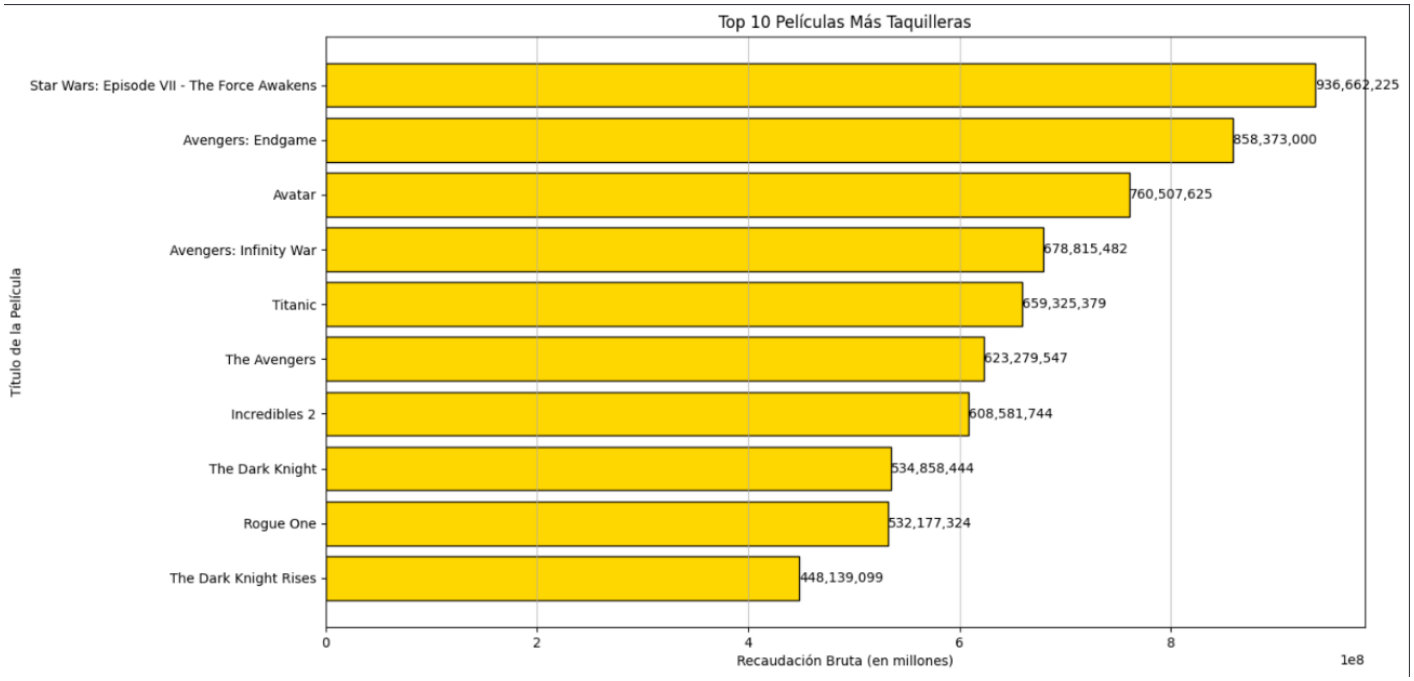
El gráfico muestra la distribución de géneros de películas. Se puede observar que el género "Drama" es el más popular, seguido de "Comedia" y "Crimen". Los géneros menos populares son "Musical" y "Sport". El gráfico también muestra que la mayoría de las películas pertenecen a unos pocos géneros, mientras que los demás géneros tienen una presencia superior.

3.5. Gráfico 5 – Top 20 actores más frecuentes



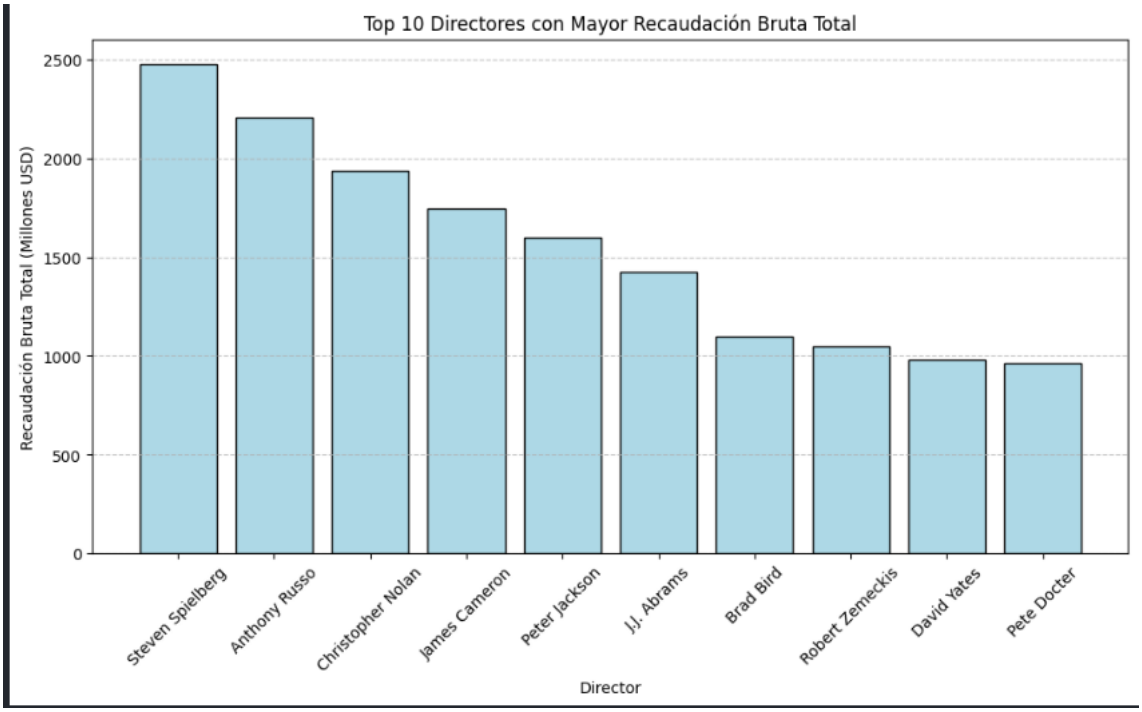
Este gráfico muestra los 20 actores que más aparecen en las películas. El eje vertical muestra la frecuencia con la que aparece cada actor, mientras que el eje horizontal muestra los nombres de los actores. El gráfico muestra que Robert De Niro es el actor que más aparece en las películas, seguido de Tom Hanks y Al Pacino.

3.6. Gráfico 6 – Películas más taquilleras



El gráfico muestra las 10 películas más taquilleras de todos los tiempos. La película en la parte superior, "Star Wars: Episodio VII - El despertar de la fuerza", recaudó 936 662 225 millones de dólares. La película en la parte inferior, "El caballero oscuro: La leyenda renace", recaudó 448 139 099 millones de dólares.

3.7. Pregunta 1 - ¿Qué directores tienden a generar mayores ingresos en la taquilla?



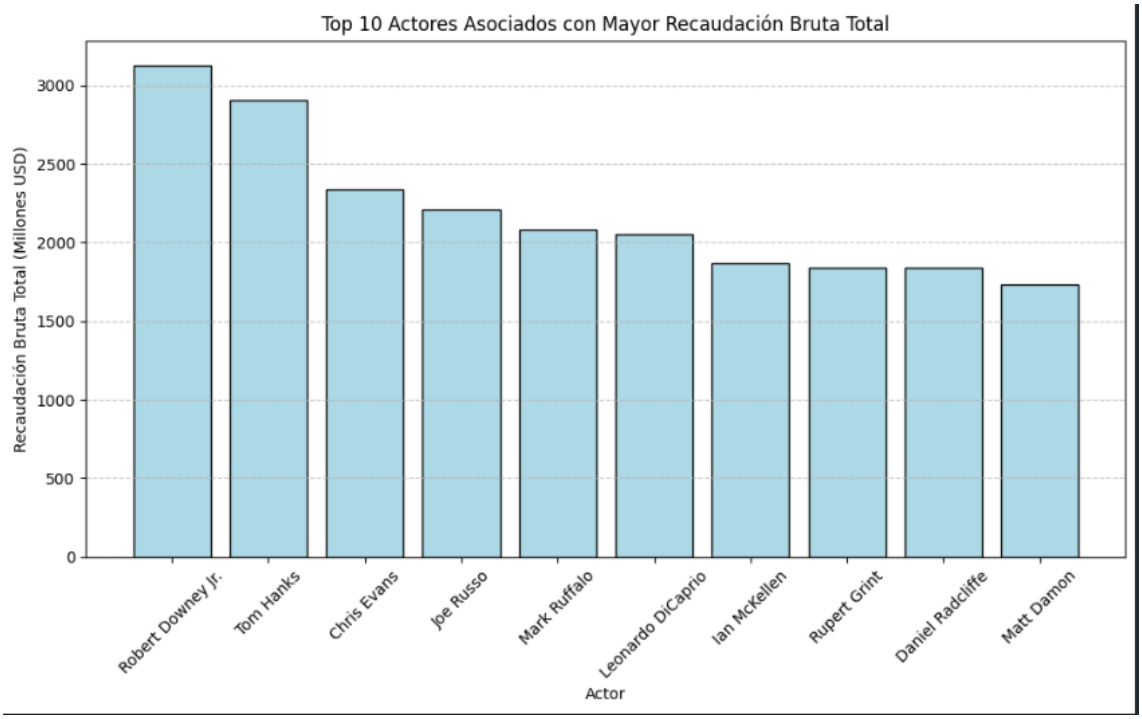
El gráfico muestra los 10 directores con la mayor recaudación bruta total en dólares.

En la parte superior de la lista está Steven Spielberg, con una recaudación bruta total de más de 2.500 millones de dólares.

El resto de los directores en la lista son, en orden de recaudación bruta total: Anthony Russo, Christopher Nolan, James Cameron, Peter Jackson, J.J. Abrams, Brad Bird, Robert Zemeckis, David Yates, y Pete Docter.

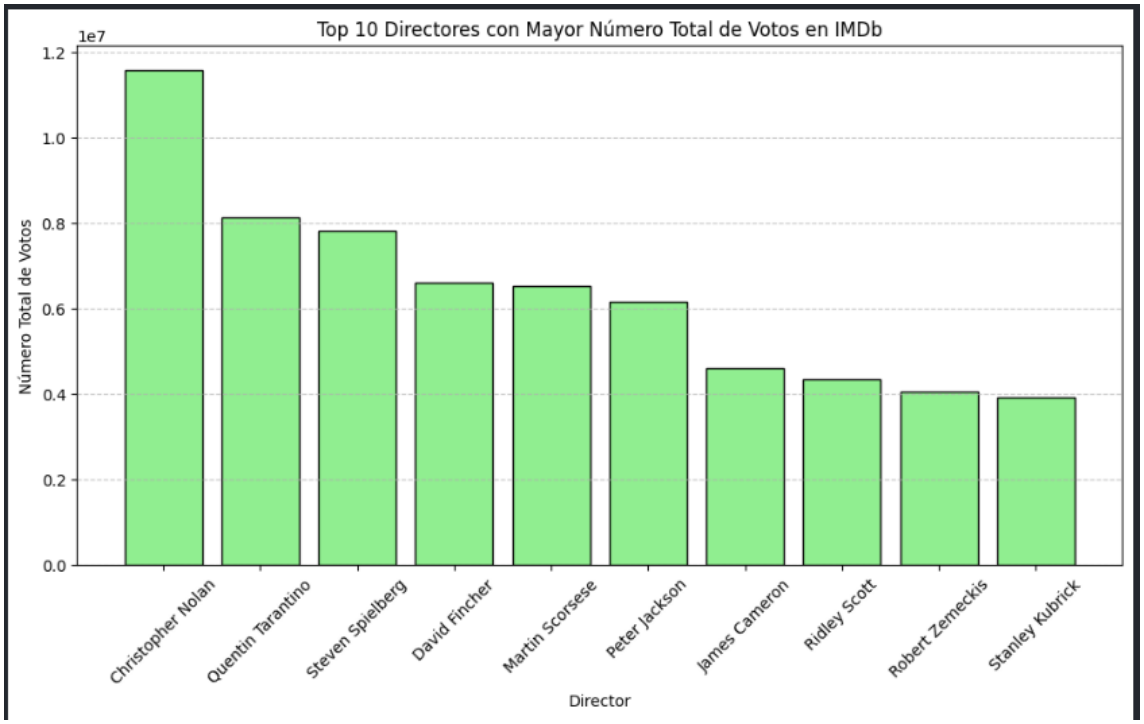
Es importante tener en cuenta que la recaudación bruta total es solo una medida de éxito, y no necesariamente refleja la calidad o el impacto de un director.

3.8. Pregunta 2 - ¿Qué actores están asociados con películas que generan mayores ingresos?



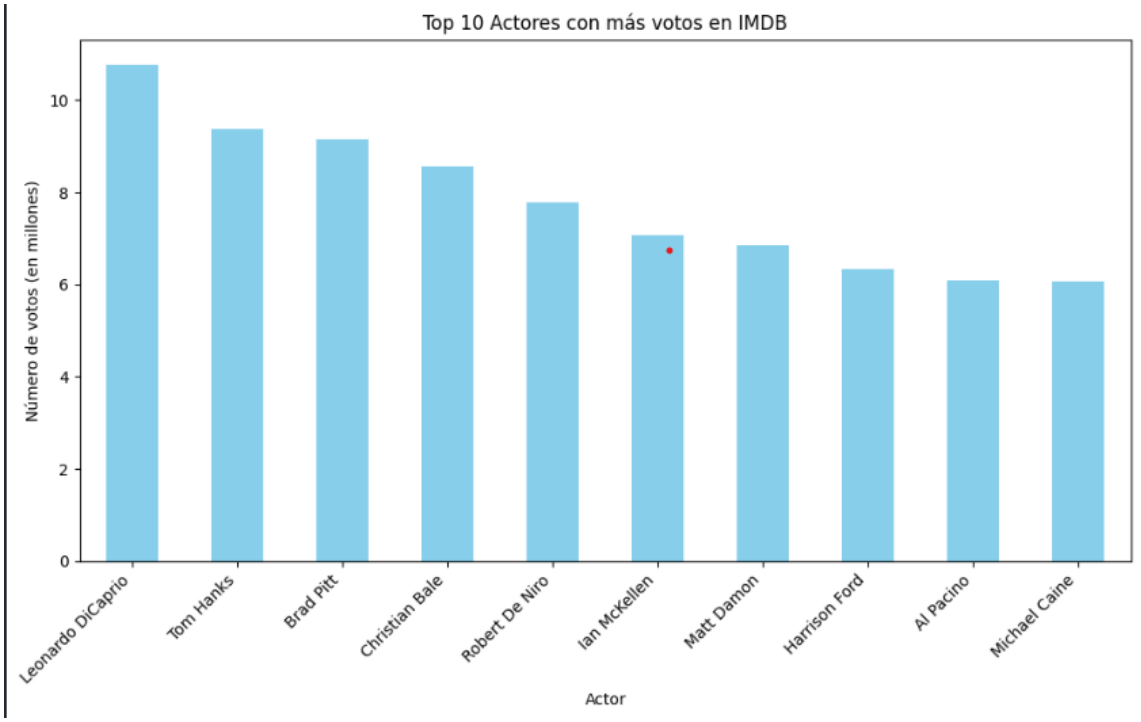
El gráfico muestra los 10 actores asociados con la mayor recaudación bruta total en la taquilla de las películas en la que han participado. El actor Robert Downey Jr. encabeza la lista con una recaudación total de más de 3000 millones de dólares. Le sigue Tom Hanks con una recaudación de casi 3000 millones de dólares. Chris Evans y Joe Russo siguen a continuación con recaudaciones de más de 2000 millones de dólares. Los siguientes cinco actores en la lista tienen recaudaciones de entre 1800 y 2000 millones de dólares, mientras que los dos últimos actores en la lista tienen recaudaciones de alrededor de 1700 millones de dólares.

3.9. Pregunta 3 - ¿Qué directores tienden a recibir más votos en IMDB?



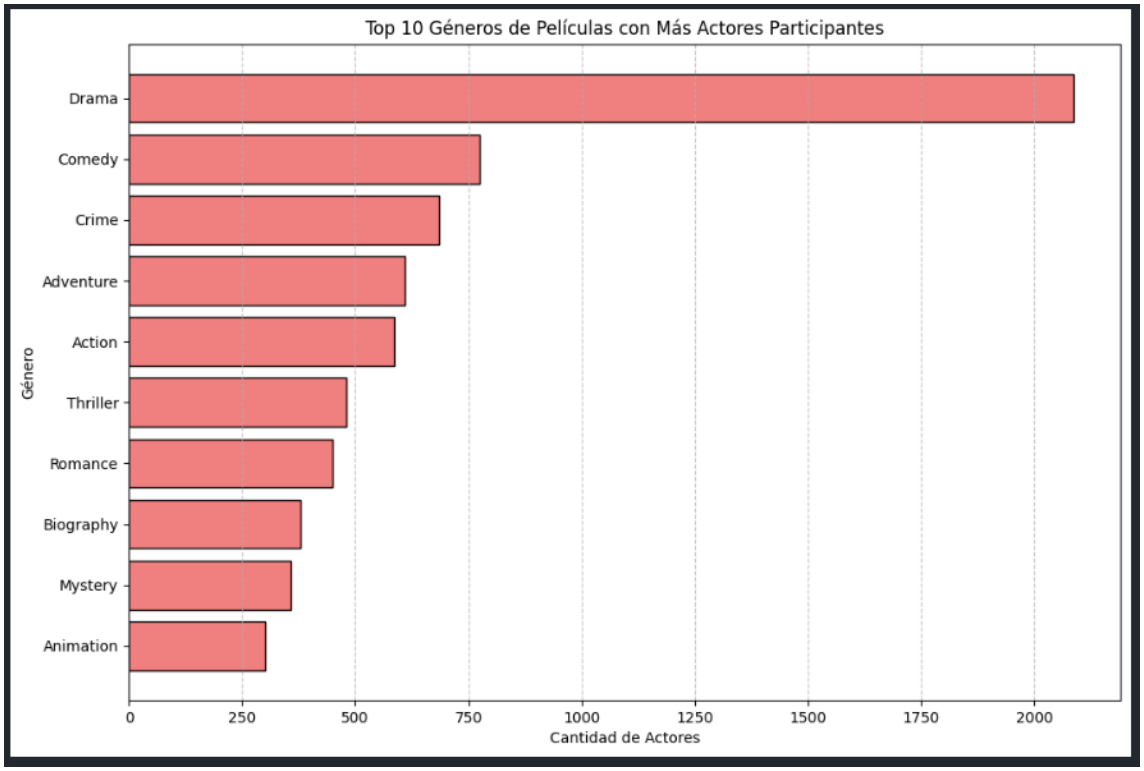
El gráfico muestra los 10 directores con mayor número de votos en IMDb. El director con más votos es Christopher Nolan, seguido de Quentin Tarantino y Steven Spielberg.

3.10. Pregunta 4 - ¿Qué actores están asociados con películas que reciben más votos?



El gráfico muestra los 10 actores con más votos en IMDB. Leonardo DiCaprio tiene el mayor número de votos, con más de 10 millones. Los demás actores tienen entre 6 y 9 millones de votos.

3.11. Pregunta 5 - Analizar la preferencia de actores por diferentes géneros cinematográficos.



El gráfico muestra los 10 géneros de películas con más actores participantes. El género con más actores participantes es Drama, seguido de Comedy, Crime, Adventure, Action, Thriller, Romance, Biography, Mystery y Ani mation.

4. Modelo Matemático: Regresión lineal múltiple

se describe la implementación de un modelo de regresión lineal múltiple utilizando Python y Numpy. La regresión lineal múltiple se utiliza para predecir una variable dependiente a partir de múltiples variables independientes. Compararemos este enfoque con la regresión lineal simple para comprender mejor el proceso.

4.1. Regresión línea simple

La regresión lineal simple se basa en el modelo:

$$Y = W \cdot X + b$$

Donde:

- Y es la variable dependiente que queremos predecir.
- X es la variable independiente.
- W es el coeficiente de la regresión (pendiente de la recta).
- b es el término independiente (intersección con el eje Y).

El objetivo es encontrar los valores óptimos para W y b que minimicen la suma de los errores cuadráticos entre los valores observados y los valores predichos. La función de costo es:

$$J(W,b) = \frac{1}{2m}\sum_{i=1}^{m}(Y_i - \hat{Y}_i)^2$$

Imágenes realizadas con LaTex

4.2. Regresión lineal multiple

La regresión lineal múltiple extiende el modelo de regresión lineal simple para incluir múltiples características. El modelo se representa como:

$$h(X) = b + W_1x_1 + W_2x_2 + \ldots + W_nx_n$$

Donde:

- $h(X)$ es la hipótesis que predice Y.
- x_i son las características de entrada.
- W_i son los coeficientes correspondientes a cada característica.
- b es el término independiente.

La función de costo para la regresión lineal múltiple se calcula de manera similar a la regresión simple:

$$J(W,b) = \frac{1}{2m}\sum_{i=1}^{m}(\hat{Y}_i - Y_i)^2$$

4.3. Obtención de las derivadas parciales

Para ajustar los parámetros del modelo, utilizamos el algoritmo de descenso de gradiente. Este algoritmo requiere el cálculo de las derivadas parciales de la función de costo con respecto a los parámetros ***W*** y ***b***. A continuación, se muestra el procedimiento para obtener estas derivadas.

4.3.1. Función de costo

La función de costo para la regresión lineal múltiple es:

$$J(W,b) = \frac{1}{2m}\sum_{i=1}^{m}(\hat{Y}_i - Y_i)^2$$

Donde:

- $\hat{Y}_i$ es la predicción del modelo para la i-ésima observación.
- Y_i es el valor real para la i-ésima observación.
- m es el número de observaciones.

La predicción del modelo $\hat{Y}_i$ se calcula como:

$$\hat{Y}_i = b + W_1 x_{i1} + W_2 x_{i2} + \ldots + W_n x_{in}$$

```
# def de  la funcion de costo
def compute_cost(X, Y, W, b):
    (variable) predictions: Any
    m = len(Y)
    predictions = np.dot(X, W) + b
    cost = np.sum((predictions - Y) ** 2) / (2 * m)
    return cost

✓ 0.0s
```

4.3.2. Derivada Parcial con respeto a *Wj*

Para encontrar la derivada parcial de la función de costo con respecto a un coeficiente W_j, utilizamos la fórmula de la derivada de la función de costo:

$$\frac{\partial J(W,b)}{\partial W_j} = \frac{1}{m}\sum_{i=1}^{m}(\hat{Y}_i - Y_i)\cdot x_{ij}$$

Explicación:

- La derivada parcial mide el cambio en la función de costo con respecto a un cambio en el coeficiente W_j.
- La fórmula suma el error de predicción ponderado por el valor de la característica x_{ij} correspondiente.
- La multiplicación por $\frac{1}{m}$ es un promedio para normalizar el cambio en el costo.

4.3.3. Derivada Parcial con respecto a ***b***

La derivada parcial de la función de costo con respecto al término independiente b es:

$$\frac{\partial J(W,b)}{\partial b} = \frac{1}{m}\sum_{i=1}^{m}(\hat{Y}_i - Y_i)$$

Explicación:

- La derivada parcial mide el cambio en la función de costo con respecto a un cambio en el término independiente b.
- La fórmula suma el error de predicción para todas las observaciones.
- La multiplicación por $\frac{1}{m}$ es un promedio para normalizar el cambio en el costo.

4.3.4. Implementación en el algoritmo de descenso de gradiente

En el algoritmo de descenso de gradiente, actualizamos los parámetros W y b utilizando las derivadas parciales calculadas:

$$W_j := W_j - \alpha\frac{\partial J(W,b)}{\partial W_j}$$

$$b := b - \alpha\frac{\partial J(W,b)}{\partial b}$$

Donde α es la tasa de aprendizaje. Esta fórmula ajusta los parámetros en la dirección que minimiza la función de costo.

```python
# funcion para  el descenso de gradiente
def gradient_descent(X, Y, W, b, learning_rate, iterations):
    m = len(Y)
    # Lista para guardar el costo en cada iteración
    costs = []
    for i in range(iterations):
        # Calcular las predicciones
        predictions = np.dot(X, W) + b

        # Calcular las derivadas
        dW = (1/m) * np.dot(X.T, (predictions - Y))
        db = (1/m) * np.sum(predictions - Y)

        # Actualizar los parámetros
        W = W - learning_rate * dW
        b = b - learning_rate * db

        # Calcular el costo
        cost = compute_cost(X, Y, W, b)
        costs.append(cost)

        # Imprimir el costo cada 100 iteraciones
        if i % 100 == 0:
            print(f"Iteración {i}: Costo {cost}")

    return W, b, costs
```

4.3.5. Entrenamiento y evaluación del modelo de regresión lineal

Primero definimos los hiperparámetos necesario para el entrenamiento del modelo. Estos incluyen la tasa de aprendizaje y el número de iteración para el algoritmo de descenso gradiente.

```
learning_rate = 0.01
iterations = 1000
```

```
    # Definir los hiperparámetros
    learning_rate = 0.01
    iterations = 1000

    # Entrenar el modelo
    W, b, costs = gradient_descent(X, Y, W, b, learning_rate, iterations)
    print(f"W: {W}")
    print(f"b: {b}")

    # Calcular el costo final
    final_cost = compute_cost(X, Y, W, b)
    print(f"Costo final: {final_cost}")

    # Calcular las predicciones
    y_pred = np.dot(X, W) + b

    # Deshacer la normalización para la variable objetivo
    y_pred = y_pred * Y_std + Y_mean

    # Calcular el RMSE
    def rmse(y_true, y_pred):
        return np.sqrt(np.mean((y_true - y_pred) ** 2))

    # Calcular el RMSE en el conjunto de datos original
    error = rmse(data['Calificacion en IMDB'], y_pred)
    print(f"RMSE del modelo: {error}")
```

```
Iteración 0: Costo 0.29543015478278695
Iteración 100: Costo 0.29543015478278695
Iteración 200: Costo 0.29543015478278695
Iteración 300: Costo 0.29543015478278695
Iteración 400: Costo 0.29543015478278695
Iteración 500: Costo 0.29543015478278695
Iteración 600: Costo 0.29543015478278695
Iteración 700: Costo 0.29543015478278695
Iteración 800: Costo 0.29543015478278695
Iteración 900: Costo 0.29543015478278695
W: [ 0.65344844  0.17976521  0.2649012  -0.3101147 ]
b: 2.853539626812564e-15
Costo final: 0.29543015478278695
RMSE del modelo: 0.211657126385364 46
```

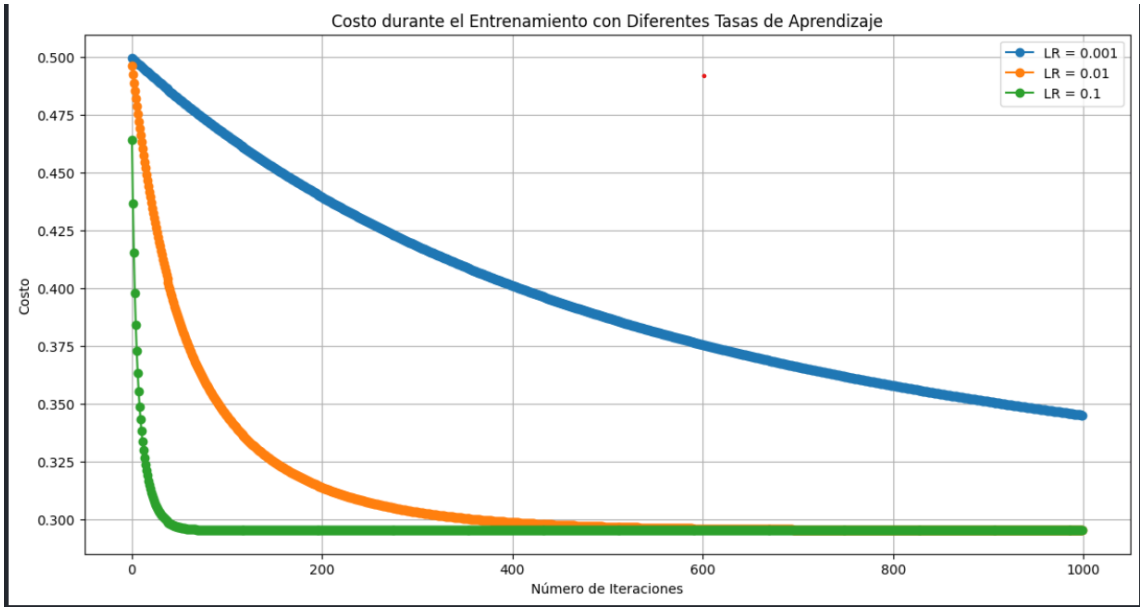
Probamos diferentes tasas de aprendizaje:

```python
# Probar Diferentes Tasas de Aprendizaje
learning_rates = [0.001, 0.01, 0.1]
iterations = 1000

results = []
# Lista para guardar los costos para cada tasa de aprendizaje

costs = []
for lr in learning_rates:
    print(f"\nProbando tasa de aprendizaje: {lr}")

    # Inicializar parámetros
    W = np.zeros(X.shape[1])
    b = 0

    # Entrenar el modelo
    W, b, cost_list = gradient_descent(X, Y, W, b, lr, iterations)

    # Guardar los costos
    costs.append(cost_list)

    # Calcular el costo final
    final_cost = cost_list[-1]

    # Calcular las predicciones
    y_pred = np.dot(X, W) + b

    # quitar la normalización para la variable objetivo
    y_pred = y_pred * Y_std + Y_mean
```

```python
    # Calcular el RMSE
    def rmse(y_true, y_pred):
        return np.sqrt(np.mean((y_true - y_pred) ** 2))

    error = rmse(data['Calificacion en IMDB'], y_pred)

    # Guardar los resultados
    results.append({
        'Learning Rate': lr,
        'Final Cost': final_cost,
        'RMSE': error
    })

# Crear un DataFrame con los resultados
results_df = pd.DataFrame(results)
print("\nResultados de las pruebas con diferentes tasas de aprendizaje:")
print(results_df)

# Graficar el costo a lo largo de las iteraciones para cada tasa de aprendizaje
plt.figure(figsize=(14, 7))

for idx, lr in enumerate(learning_rates):
    plt.plot(costs[idx], label=f'LR = {lr}', linestyle='-', marker='o')

plt.xlabel('Número de Iteraciones')
plt.ylabel('Costo')
plt.title('Costo durante el Entrenamiento con Diferentes Tasas de Aprendizaje')
plt.legend()
plt.grid(True)
plt.show()
```

✓ 0.2s

```
Probando tasa de aprendizaje: 0.001
Iteración 0: Costo 0.4996219185041025
Iteración 100: Costo 0.46607561678884486
Iteración 200: Costo 0.43955117022265083
Iteración 300: Costo 0.418355525094976
Iteración 400: Costo 0.4012341148086281
Iteración 500: Costo 0.3872530134140151
Iteración 600: Costo 0.3757134261139
Iteración 700: Costo 0.36608953215211626
Iteración 800: Costo 0.3579832112456708
Iteración 900: Costo 0.35109099257976134

Probando tasa de aprendizaje: 0.01
Iteración 0: Costo 0.49624002318105864
Iteración 100: Costo 0.34451131253718888
Iteración 200: Costo 0.3139607447586195
Iteración 300: Costo 0.3032372338370151
Iteración 400: Costo 0.29881861755096534
Iteración 500: Costo 0.29691414560469764
Iteración 600: Costo 0.2960819245988033
Iteración 700: Costo 0.29571667247084804
Iteración 800: Costo 0.295556144527716
Iteración 900: Costo 0.295485561162827
...
   Learning Rate  Final Cost      RMSE
0          0.001    0.345235  0.228804
1          0.010    0.295455  0.211666
2          0.100    0.295430  0.211657
```



El gráfico muestra el costo de entrenamiento de un modelo de aprendizaje automático para diferentes tasas de aprendizaje. La tasa de aprendizaje es un

parámetro que determina cuánto cambia el modelo en cada paso de entrenamiento.

- Tasa de aprendizaje baja (LR = 0.001): El modelo converge lentamente hacia el mínimo del costo. Toma muchas iteraciones para llegar a un punto donde el costo se estabiliza.
- Tasa de aprendizaje media (LR = 0.01): El modelo converge más rápido que con la tasa de aprendizaje baja, pero aún tarda un número considerable de iteraciones.
- Tasa de aprendizaje alta (LR = 0.1): El modelo converge rápidamente, pero puede sobrepasar el mínimo del costo y oscilar a su alrededor. En este caso, el modelo encuentra un mínimo local en vez del mínimo global, lo que puede ser un problema.

5. Conclusiones

Directores y Actores Asociados con Mayores Ingresos en Taquilla:

- El análisis reveló que ciertos directores y actores tienen un impacto significativo en los ingresos de taquilla. Directores como Christopher Nolan y Steven Spielberg, y actores como Leonardo DiCaprio y Tom Hanks, están asociados con películas que generan altos ingresos, lo que indica su popularidad y capacidad para atraer audiencias.

Géneros Populares:

- Los géneros de películas más populares, como la acción, el drama y la comedia, dominan la industria en términos de producción y preferencias del público. Estos géneros no solo son los más producidos, sino que también reciben altas puntuaciones y votos en IMDb, reflejando su aceptación generalizada.

Tendencias Temporales:

- La distribución de películas y series a lo largo de los años mostró un aumento constante en la producción cinematográfica, especialmente a partir de la década de 2000. Este crecimiento puede atribuirse a la expansión de la industria del entretenimiento, el aumento de plataformas de streaming y el desarrollo tecnológico que ha facilitado la producción y distribución de contenido.

Influencia en las Votaciones de IMDb:

- Los análisis indicaron que las películas dirigidas por ciertos directores y protagonizadas por actores reconocidos tienden a recibir más votos y puntuaciones más altas en IMDb. Esto sugiere que la reputación y la popularidad de los directores y actores pueden influir significativamente en la percepción y el rendimiento de las películas.

Eficiencia de los Modelos de Regresión Lineal:

- La implementación de modelos de regresión lineal permitió identificar factores clave que afectan los ingresos de taquilla y las puntuaciones de IMDb. Aunque los modelos proporcionaron una buena base para el análisis, se observó que factores adicionales, no capturados en el dataset, también influyen en estos resultados, lo que sugiere la necesidad de un análisis más profundo y la inclusión de más variables en futuros estudios.

Importancia de la Limpieza de Datos:

- La fase de carga y limpieza de datos fue crucial para asegurar la integridad y la calidad del análisis. El manejo adecuado de valores nulos y la conversión correcta de tipos de datos fueron esenciales para obtener resultados precisos y confiables. Este proceso destacó la importancia de una preparación de datos meticulosa para cualquier proyecto de análisis.

Visualización de Datos:

- Las visualizaciones generadas, como gráficos de barras y dispersión, facilitaron la comprensión de las tendencias y patrones en los datos. Estas visualizaciones no solo hicieron el análisis más accesible, sino que también proporcionaron una forma clara y efectiva de comunicar los hallazgos clave.

Recomendaciones para la Industria del Entretenimiento:

- Basado en los insights obtenidos, se pueden hacer varias recomendaciones a los profesionales de la industria del entretenimiento. Por ejemplo, invertir en talentos reconocidos y populares puede aumentar significativamente las probabilidades de éxito comercial de una película. Además, prestar atención a las tendencias de género y las preferencias del público puede ayudar a orientar la producción hacia contenido más demandado.