



## Tópicos avanzados en analítica

### Proyecto No 1

#### Autores:

Ian Nicolas Rincón, Rosemary Ríos Pulido,  
Carlos Felipe Mora, John Sebastián Martínez  
Andrés Parra Rodríguez <sup>a</sup>

#### Docente:

Sergio Mora

<sup>a</sup> Estudiantes de Maestría en Analítica para la Inteligencia de Negocios

Departamento de Ingeniería Industrial

Pontificia Universidad Javeriana

Bogotá, Colombia

## TALLER 1 TÓPICOS AVANZADOS

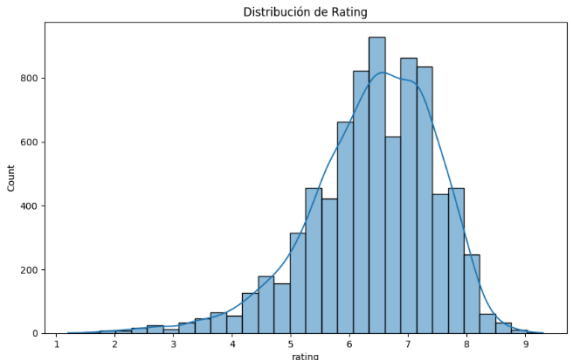
### 1) Objetivo.

Clasificar el género de una película según su descripción, considerando que una película puede pertenecer a varios géneros.

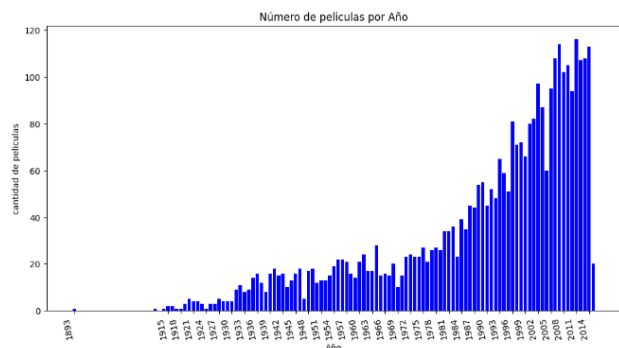
### 2) Adquisición de datos y Análisis descriptivo.

Para la adquisición de los datos se hace indispensable el uso de la biblioteca de 'pandas' en Python, dado que esta es una gran herramienta para la manipulación y el análisis de los datos. Los datos se encuentran en un repositorio de GitHub de manera comprimida. La adquisición se realiza mediante dos fases principales: la carga de los datos de entrenamiento y la carga de los datos de prueba.

Una vez se cargan los datos, se inicia con el proceso de entendimiento de los datos en donde se observa lo siguiente:

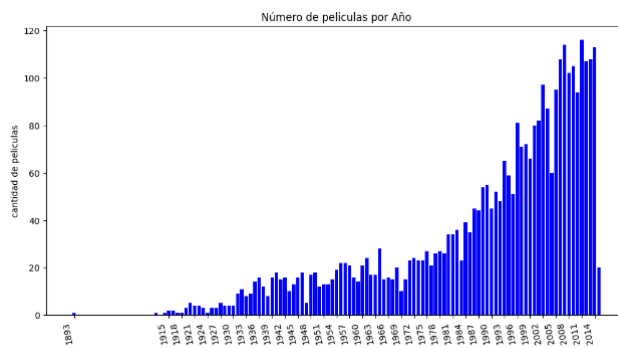
Base de datos de entrenamiento	Base de datos de testeo
Está conformado por 5 campos (year, title, plot, genres y rating) y 7.895 registros.	Está conformado por 3 campos (year, title y plot) y 3.383 registros.
No se presenta nulos en ninguno de los campos	No presenta nulos en ninguno de los campos
El periodo de estudio inicia desde el año 1894 al 2015.	El periodo de estudio inicia desde el año 1894 al 2015.
<p>La calificación del rating de las películas observadas va desde una calificación 1,2 hasta 9,3 donde se evidencia que la mayor cantidad de películas (cerca de 890) tiene una calificación de 6,5; pareciéndose a la calificación media de 6,4. Mediante un gráfico de distribución se observa que tiende a estar sesgada a la derecha:</p>  <p><b>Gráfico 1: Distribución del rating</b></p>	

Se observa un crecimiento de películas estrenadas por año llegando a su máximo pico en el 2013 acercándose a las 280 películas como se presenta en el siguiente gráfico:



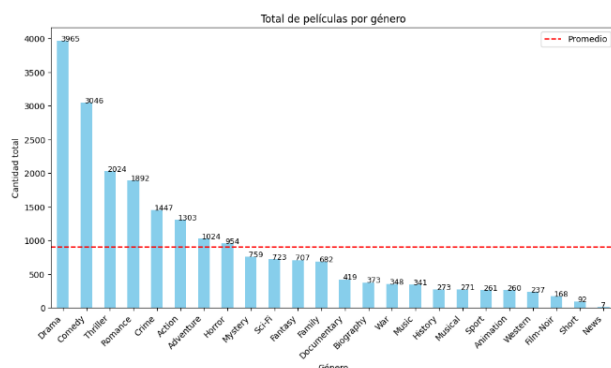
**Gráfico 2: Evolución de las películas por año**

Se observa un crecimiento de películas por año llegando a su máximo pico en el 2011 acercándose a las 115 películas (menos de la mitad que tuvo el máximo histórico en la información de entrenamiento) como se evidencia en el siguiente gráfico:



**Gráfico 3: Evolución de las películas por año**

Se presentan 24 géneros distintos de películas, en donde pueden combinar entre 2 a 4 géneros y se evidencia que el drama, la comedia, thriller, romance, crimen, acción, aventura y horror son las categorías más lanzadas (por encima del promedio) en la muestra de entrenamiento:



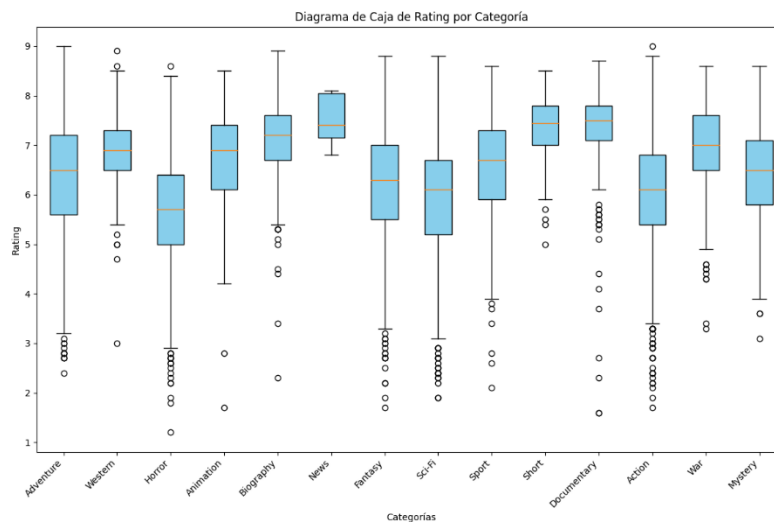
**Gráfico 4: Cantidad de películas por género**

Por otra parte, se realiza un top de las palabras más repetidas en el campo “title” por género y se evidencia que “find”, “one”, “life” están mencionadas en todas las categorías.



**Gráfico 5: Top de palabras repetidas por género en la base de entrenamiento**

Se lleva a cabo un análisis comparativo del rating según el género, revelando que las películas que exhiben el rating promedio más alto (7,8), aunque se identifican 13 documentales como valores atípicos con ratings inferiores a 6. En contraste, las películas de horror muestran los ratings promedio más bajos (5,8), con 10 películas que se encuentran fuera del rango intercuartílico, llegando incluso a un rating mínimo de 1,2.



**Gráfico 6: Caja y bigotes de los ratings por género en la base de entrenamiento**

### 3) Limpieza de datos

En esta etapa se realizó ajustes a los datos de tipo texto la aplicación de minúsculas, la eliminación de los signos de puntuación) y palabras de poco valor (mediante el comando stopwords) o palabras que se repetían varias veces como n, find y life:

Modelo	Minúsculas	Eliminación de stopwords	Eliminación de signos de puntuación	Eliminación de palabras
XGBOOST	Si	Si	Si	Si
CLASIFICACIÓN MULTINOMIAL NAIVE BAYES	Si	Si	Si	Si
REGRESIÓN LOGISTICA MULTINOMIAL	Si	Si	Si	Si
RANDOM FOREST	Si	Si	Si	Si
SUPPORT VECTOR MACHINE	Si	Si	Si	Si
REGRESIÓN LOGISTICA MULTINOMIAL CON VALIDACIÓN CRUZADA	Si	Si	Si	Si

*Tabla 1: Resumen de limpieza de datos por modelo*

### 4) Preprocesamiento

Modelo	Descripción
<b>XGBOOST CON N-GRAMAS DE 1-8 Y COUNTVECTORIZER</b>	<ol style="list-style-type: none"><li>1. Los datos de entrenamiento fueron divididos en dos conjuntos utilizando la función <code>train_test_split</code>, asignando el 67% de los datos para el entrenamiento del modelo y el restante para pruebas.</li><li>2. Se utilizó <code>TextPreprocessor</code> para realizar el preprocesamiento inicial del texto eliminando puntuación, stopwords y ajustando el texto a minúsculas.</li><li>3. Se realiza vectorización con <code>countvectorizer</code>.</li><li>4. Los datos fueron clasificados utilizando <code>xgb.XGBClassifier()</code> como clasificador.</li></ol>
<b>XGBOOST CON N-GRAMAS DE 1-2, LEMATIZACIÓN Y TFIDFVECTORIZER</b>	<ol style="list-style-type: none"><li>1. Los datos de entrenamiento se dividieron en dos partes mediante la función <code>train_test_split</code>, asignando el 67% de los datos para el entrenamiento del modelo y el resto para pruebas.</li><li>2. Se utilizó <code>TextPreprocessor</code> para realizar el preprocesamiento inicial del texto eliminando puntuación, stopwords y ajustando el texto a minúsculas.</li><li>3. Se proceso el texto utilizando al función <code>token.lemma</code>, lo cual permite la lematización y la conversión de las palabras a su forma base.</li><li>4. Se realiza vectorización con <code>TFIDF</code>.</li><li>5. Se modelaron los datos con el comando <code>xgb.XGBClassifier()</code>.</li></ol>
<b>XGBOOST CON N-GRAMAS DE 1-8, LEMATIZACIÓN Y COUNTVECTORIZER</b>	<ol style="list-style-type: none"><li>1. Los datos de entrenamiento se dividieron en dos partes mediante la función <code>train_test_split</code>, asignando el 67% de los datos para el entrenamiento del modelo y el resto para pruebas.</li></ol>

	<p>2. Se utilizó TextPreprocessor para realizar el preprocesamiento inicial del texto eliminando puntuación y ajustando el texto a minúsculas.</p> <p>3. Se procesó el texto con lematización buscando la conversión de las palabras a su forma base.</p> <p>4. Se realiza vectorización con countvectorizer.</p> <p>5. Se modelaron los datos con el comando xgb.XGBClassifier().</p>
<b>XGBOOST CON N-GRAMAS DE 1-8, LEMATIZACIÓN, ESTEMATIZACIÓN Y COUNTVECTORIZER</b>	<p>1. Los datos de entrenamiento se dividieron en dos partes mediante la función train_test_split, asignando el 67% de los datos para el entrenamiento del modelo y el resto para pruebas.</p> <p>2. Se utilizó TextPreprocessor para realizar el preprocesamiento inicial del texto eliminando puntuación, stopwords y ajustando el texto a minúsculas.</p> <p>3. Se llevó a cabo la lematización utilizando un léxico descargado llamado WordNet para convertir las palabras a su forma base.</p> <p>4. El texto se normalizó mediante la reducción de palabras a su raíz o base utilizando el comando use_stemming.</p> <p>5. Se realiza vectorización con countvectorizer.</p> <p>6. Se modelaron los datos con el comando xgb.XGBClassifier(), asignando los siguientes hiperparámetros:</p> <ul style="list-style-type: none"> <li>• Se estableció una tasa de aprendizaje del 5% para controlar la contribución de cada árbol en el conjunto, lo que generaliza más el ensamble, con 400 árboles.</li> <li>• Se definió una profundidad máxima de 5 para capturar interacciones no lineales en los datos.</li> <li>• Se configuró una instancia mínima de 1 en cada nodo para evitar divisiones que conduzcan a regiones muy pequeñas del espacio de características.</li> <li>• Se utilizó el 80% de las características para entrenar cada árbol.</li> <li>• Se optimizó la pérdida logística.</li> </ul>
<b>CLASIFICACIÓN MULTINOMIAL NAIVE BAYES CON LEMATIZACIÓN Y TFIDFVECTORIZER</b>	<p>1. Los datos de entrenamiento se dividieron en dos partes mediante la función train_test_split, asignando el 67% de los datos para el entrenamiento del modelo y el resto para pruebas.</p> <p>2. Se realizó el preprocesamiento inicial del texto eliminando puntuación, stopwords y ajustando el texto a minúsculas.</p> <p>3. Se llevó a cabo la lematización para convertir las palabras a su forma base.</p> <p>4. Se realiza vectorización con TFIDF.</p> <p>5. Las etiquetas de género fueron transformadas de cadenas de texto a listas y posteriormente se aplicaron técnicas de binarización para convertirlas en un formato adecuado para el modelado de múltiples etiquetas.</p> <p>6. Se modelaron los datos con el comando MultinomialNB().</p>
<b>REGRESIÓN LOGÍSTICA MULTINOMIAL CON LEMATIZACIÓN, TFIDFVECTORIZER Y GRIDSEARCH</b>	<p>1. Los datos de entrenamiento se dividieron en dos partes mediante la función train_test_split, asignando el 67% de los datos para el entrenamiento del modelo y el resto para pruebas.</p> <p>2. Se realizó el preprocesamiento inicial del texto eliminando puntuación, caracteres no alfabéticos, stopwords y ajustando el texto a minúsculas.</p> <p>3. Se llevó a cabo la lematización para convertir las palabras a su forma base.</p> <p>4. Se realiza vectorización con TFIDF.</p>

	<p>5. Se modelaron los datos con el comando <code>LogisticRegression()</code>.</p> <p>6. Se utilizó <code>GridSearchCV</code> para realizar una búsqueda exhaustiva de hiperparámetros mediante validación cruzada. Se buscó la combinación óptima de hiperparámetros que maximiza el área bajo la curva ROC (ROC AUC) en el conjunto de entrenamiento.</p>
<b>REGRESIÓN LOGISTICA MULTINOMIAL CON N-GRAMAS DE 1-2, LEMATIZACIÓN Y TfidfVectorizer</b>	<p>1. Los datos de entrenamiento fueron divididos en dos conjuntos utilizando la función <code>train_test_split</code>, asignando el 67% de los datos para el entrenamiento del modelo y el restante para las pruebas.</p> <p>2. Se definió una función <code>clean_text</code> para la limpieza del texto en la columna 'plot'. Esta limpieza incluyó la conversión del texto a minúsculas, eliminación de caracteres no alfabéticos, eliminación de palabras vacías (stop words).</p> <p>3. Se aplicó lematización utilizando la función <code>WordNetLemmatizer</code> para convertir las palabras a su forma base.</p> <p>4. Se hace vectorización con TF-IDF (<code>TfidfVectorizer</code>) para convertir el texto en características numéricas.</p> <p>5. La clasificación multiclase se llevó a cabo utilizando <code>OneVsRestClassifier</code>, el cual utiliza <code>LogisticRegression</code> como clasificador base.</p>
<b>RANDOM FOREST CON LEMATIZACIÓN Y COUNTVECTORIZER</b>	<p>1. Se dividió los datos de entrenamiento en dos partes, con la función la función <code>train_test_split</code> en donde se puso el 67% de los datos para entrenar el modelo y el restante para el testeo</p> <p>2. Se define una función <code>clean_text</code> para limpiar el texto de la columna 'plot'. Esto incluye convertir el texto a minúsculas, eliminar caracteres no alfabéticos, eliminar palabras vacías (stop words).</p> <p>3. Se aplicó lematización con la función <code>WordNetLemmatizer</code> para que las palabras se conviertan a su forma base</p> <p>4. Se hace vectorización con <code>Countvectorizer</code> para convertir el texto en características numéricas.</p> <p>5. Se define construir 100 árboles de decisión con 10 niveles de profundidad</p>
<b>SUPPORT VECTOR MACHINE CON N-GRAMAS DE 1-2, LEMATIZACIÓN Y TfidfVectorizer</b>	<p>1. Los datos de entrenamiento fueron divididos en dos conjuntos utilizando la función <code>train_test_split</code>, asignando el 67% de los datos para el entrenamiento del modelo y el resto para las pruebas.</p> <p>2. Se definió la función <code>clean_text</code> para llevar a cabo la limpieza del texto en la columna 'plot'. Esta limpieza incluyó la conversión del texto a minúsculas, la eliminación de caracteres no alfabéticos, la eliminación de palabras vacías (stop words).</p> <p>3. Se aplicó la lematización utilizando la función <code>WordNetLemmatizer</code> para llevar las palabras a su forma base.</p> <p>4. Se hace vectorización con TF-IDF (<code>TfidfVectorizer</code>) para convertir el texto en características numéricas.</p> <p>5. Se utilizó un clasificador de Máquinas de Vectores de Soporte (SVM) a través de la estrategia uno-contra-el-resto (<code>OneVsRestClassifier</code>). Se aplicó regularización y una función sigmoide con un kernel para hacer linealmente separables las clases de géneros.</p>
<b>SUPPORT VECTOR MACHINE CON N-GRAMAS DE 1-2, ESTEMATIZACIÓN Y TfidfVectorizer</b>	<p>1. Los datos de entrenamiento fueron divididos en dos conjuntos utilizando la función <code>train_test_split</code>, asignando el 67% de los datos para el entrenamiento del modelo y el restante para las pruebas.</p> <p>2. Se definió una función <code>clean_text</code> para realizar la limpieza del texto en la columna 'plot'. Esto implicó la conversión del texto a minúsculas, la eliminación de caracteres no alfabéticos, la supresión de palabras vacías (stop words).</p>

	<p>3. Se aplicó la estematización utilizando la función PorterStemmer() para llevar las palabras a su forma base.</p> <p>4. Se hace vectorización con TF-IDF (TfidfVectorizer) para convertir el texto en características numéricas.</p> <p>5. Se utilizó un clasificador de Máquinas de Vectores de Soporte (SVM) a través de la estrategia uno-contra-el-resto (OneVsRestClassifier). Se aplicó regularización y una función sigmoide con un kernel para hacer linealmente separables las clases de géneros.</p>
<b>REGRESIÓN LOGISTICA MULTINOMIAL CON N-GRAMAS DE 1-3, ESTEMATIZACIÓN Y TFIDFVECTORIZER</b>	<p>1. Se dividió los datos de entrenamiento en dos partes, con la función la función train_test_split en donde se puso el 67% de los datos para entrenar el modelo y el restante para el testeo</p> <p>2. Se define una función clean_text para limpiar el texto de la columna 'plot'. Esto incluye convertir el texto a minúsculas, eliminar caracteres no alfabéticos, eliminar palabras vacías (stop words).</p> <p>3. Se aplicó la estematización utilizando la función PorterStemmer() para llevar las palabras a su forma base.</p> <p>4. Se hace vectorización con TF-IDF (TfidfVectorizer) para convertir el texto en características numéricas</p> <p>5. Se utiliza un clasificador de regresión logística con la estrategia One-vs-Rest para predecir las probabilidades de pertenencia a cada género para las instancias de prueba. Estas probabilidades se utilizan luego para calcular el área bajo la curva (AUC) como medida de rendimiento del modelo.</p>
<b>REGRESIÓN LOGISTICA MULTINOMIAL CON N-GRAMAS DE 1-3, ESTEMATIZACIÓN, TFIDFVECTORIZER Y VALIDACIÓN CRUZADA (K-FOLD)</b>	<p>1. Los datos de entrenamiento fueron divididos en dos conjuntos utilizando la función train_test_split, asignando el 67% de los datos para entrenar el modelo y el resto para realizar pruebas.</p> <p>2. Se definió una función llamada clean_text para llevar a cabo la limpieza del texto en la columna 'plot'. Esto incluyó la conversión del texto a minúsculas, la eliminación de caracteres no alfabéticos, la supresión de palabras vacías (stop words).</p> <p>3. Se aplicó la estematización utilizando la función PorterStemmer() para llevar las palabras a su forma base.</p> <p>4. Se realiza vectorización con TfidfVectorizer.</p> <p>5. Se utilizó el comando MultiOutputClassifier con LogisticRegression como clasificador base para predecir múltiples etiquetas de salida.</p> <p>Se implementó una función logística con validación cruzada utilizando 10 divisiones para evaluar su rendimiento utilizando la curva ROC.</p>

**Tabla 2: Resumen de procesamiento de datos por modelo**

## 5) Representación

Modelo	Detalle
<b>XGBOOST</b>	<p>Se vectoriza con la condición de tener máximo 8.000 palabras con un conjunto de palabras de 1-8 por n-gramas</p> <p>Se vectoriza con la condición de tener máximo 4.000 palabras con un conjunto de palabras de 1-2 por n-gramas</p>



	Se vectoriza con la condición de tener máximo 10.000 palabras con un conjunto de palabras de 1-2 por n-gramas.
<b>CLASIFICACIÓN MULTINOMIAL NAIVE BAYES</b>	Se vectoriza con la condición de tener máximo 2.000 palabras.
<b>REGRESIÓN LOGISTICA MULTINOMIAL</b>	Se vectoriza con la condición de tener máximo 10.000 palabras con un conjunto de palabras de 1-2 por n-gramas. Se vectoriza con la condición de tener máximo 10.000 palabras con un conjunto de palabras de 1-3 por n-gramas.
<b>RANDOM FOREST</b>	Se vectoriza con la condición de tener máximo 10.000 palabras con un conjunto de palabras de 1-2 por n-gramas.
<b>SUPPORT VECTOR MACHINE</b>	Se vectoriza con la condición de tener máximo 10.000 palabras con un conjunto de palabras de 1-2 por n-gramas
<b>REGRESIÓN LOGISTICA MULTINOMIAL CON VALIDACIÓN CRUZADA (K-FOLD)</b>	Se vectoriza con la condición de tener máximo 8.000 palabras con un conjunto de palabras de 1-2 por n-gramas

*Tabla 3: Resumen de la selección de palabras y numero de n-gramas por modelo*

## 6) Evaluación

A continuación, se presenta un resumen de cada modelo generado con su AUC:

Modelo	AUC
<b>XGBOOST CON N-GRAMAS DE 1-8 Y COUNTVECTORIZER</b>	0.8197
<b>XGBOOST CON N-GRAMAS DE 1-2, LEMATIZACIÓN Y TFIDFVECTORIZER</b>	0.8268
<b>XGBOOST CON N-GRAMAS DE 1-8, LEMATIZACIÓN Y COUNTVECTORIZER</b>	0.8148
<b>XGBOOST CON N-GRAMAS DE 1-8, LEMATIZACIÓN, ESTEMATIZACIÓN Y COUNTVECTORIZER</b>	0.8482
<b>CLASIFICACIÓN MULTINOMIAL NAIVE BAYES CON LEMATIZACIÓN Y TFIDFVECTORIZER</b>	0.8264
<b>REGRESIÓN LOGÍSTICA MULTINOMIAL CON LEMATIZACIÓN, TFIDFVECTORIZER Y GRIDSEARCH</b>	0.8791
<b>REGRESIÓN LOGISTICA MULTINOMIAL CON N-GRAMAS DE 1-2, LEMATIZACIÓN Y TFIDFVECTORIZER</b>	0.8813
<b>RANDOM FOREST CON LEMATIZACIÓN Y COUNTVECTORIZER</b>	0.8206
<b>SUPPORT VECTOR MACHINE CON N-GRAMAS DE 1-2, LEMATIZACIÓN Y TFIDFVECTORIZER</b>	0.8463
<b>SUPPORT VECTOR MACHINE CON N-GRAMAS DE 1-2, ESTEMATIZACIÓN Y TFIDFVECTORIZER</b>	0.8508
<b>REGRESIÓN LOGISTICA MULTINOMIAL CON N-GRAMAS DE 1-3, ESTEMATIZACIÓN Y TFIDFVECTORIZER</b>	0.8820

<b>REGRESIÓN LOGISTICA MULTINOMIAL CON N-GRAMAS DE 1-3, ESTEMATIZACIÓN, TFIDFVECTORIZER Y VALIDACIÓN CRUZADA (K-FOLD)</b>	0.8933
---	--------

*Tabla 4: Resumen de rendimiento por modelo*

## 7) Elección del Modelo

Se determina que el modelo de Regresión Logística Multinomial, utilizando n-gramas de 1 a 3, junto con la estematización, el TfidfVectorizer y validación cruzada (K-fold), resulta ser el más efectivo en la tarea de predecir los géneros de una película basándose en su descripción. Este hallazgo se fundamenta en su alto valor de Área Bajo la Curva (AUC), indicando que el modelo predice mejor que clasificar al azar.