Facultad de Ingeniería

Tópicos Avanzados de la Analítica

Caso de Negocio – Primer Semestre 2023

Carlos Andrés Másmela Pinilla
Daniela Isabel Zárate Bonilla
Yudy Tatiana Pedraza Torres
Adrián Ernesto Esquinas Álvarez

Contenido

1. Business Understanding

La clasificación de películas a través de su descripción es una herramienta valiosa que puede ayudar a las empresas especialmente de la industria de streaming a mejorar la experiencia del usuario y aumentar las ventas. La clasificación de películas a través de su descripción puede proporcionar beneficios tanto para los usuarios como para las empresas. Para los usuarios, la clasificación puede ayudar a reducir la sobrecarga de información y facilitar la identificación de películas que sean de su interés. Para las empresas, la clasificación puede ayudar a mejorar la experiencia del usuario, aumentar las ventas y mejorar la satisfacción del cliente.

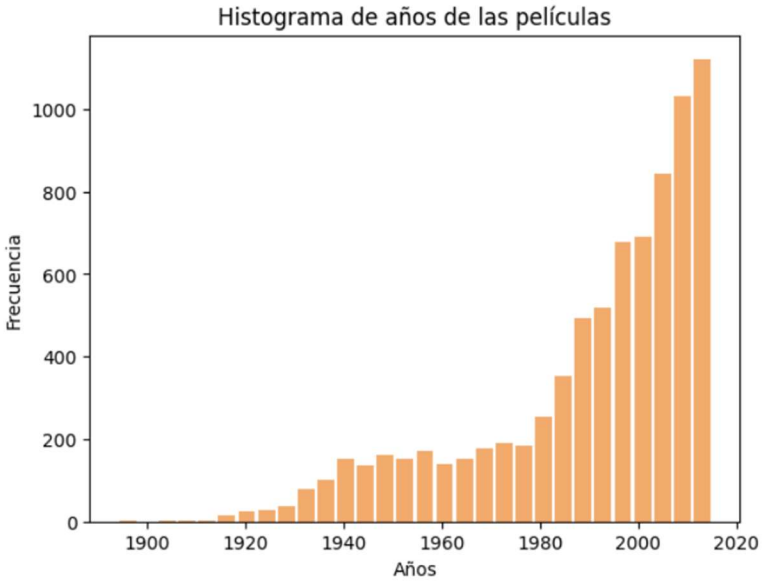
Para realizar una clasificación de películas eficaz, es importante comprender los siguientes factores:

- **Los objetivos del negocio:** ¿Qué espera lograr el negocio con la clasificación de películas?
- **Los usuarios:** ¿Quiénes son los usuarios de la clasificación? ¿Cuáles son sus necesidades y expectativas?
- **Las películas:** ¿Qué características de las películas son relevantes para la clasificación?

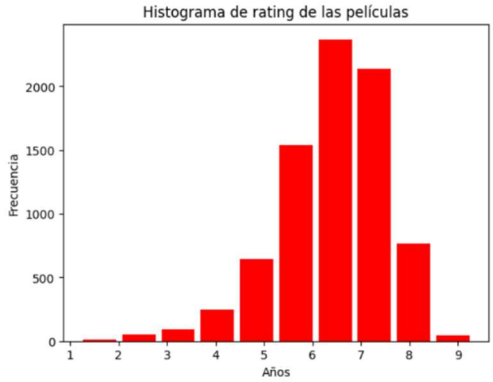
El NLP es una herramienta clave para la clasificación de películas. Este enfoque implica desarrollar un conjunto de reglas que definan cómo las películas deben clasificarse. Estas reglas pueden basarse en características específicas de las películas, como el género, el tema o la calificación de edad. Con el NLP se puede mejorar la precisión y la eficiencia de la clasificación.

2. Entendimiento de los datos

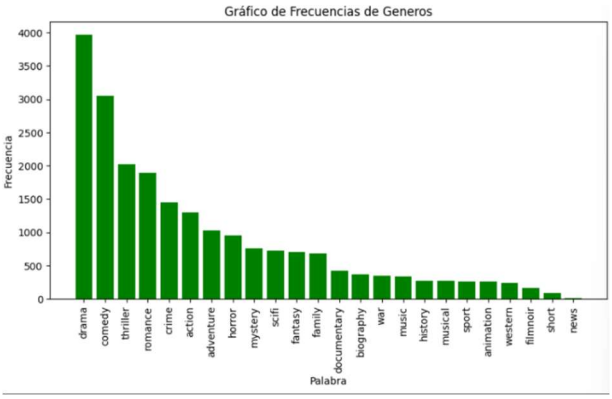
Los datos proporcionados los cuales contienen 7.895 películas comprenden el año, el título, una corta descripción (plot) de la película, el rating y la variable a predecir, el género. Al explorar la variable del año en el que se estrenaron las películas, se puede evidenciar que al menos el 50% de las películas son del año 1980 hasta el año 2015, adicionalmente que las películas se encuentran entre los años 1894 y 2015.



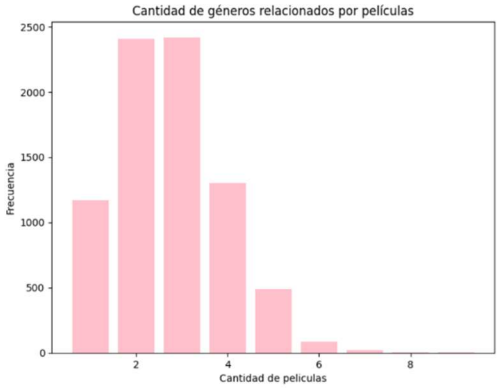
Por otro lado, la variable rating, la cual da información sobre el puntaje que tiene la película, se puede observar que en promedio las películas tienen un puntaje de 6.4, siendo 1.2 el puntaje mínimo y 9.3 el puntaje máximo.



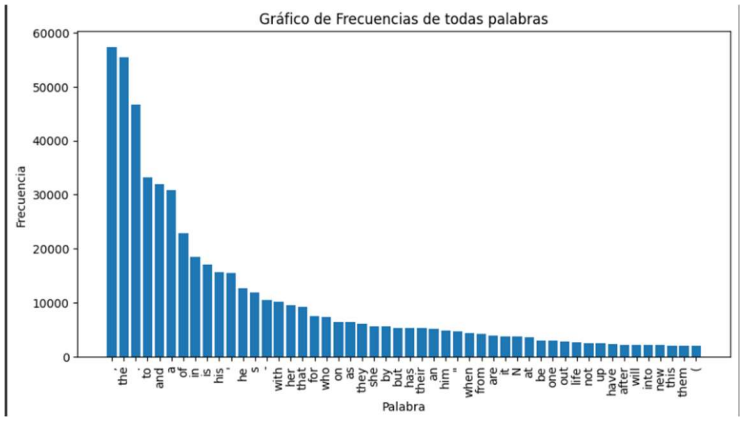
Para la variable a predecir, los géneros, en la data proporcionada se puede observar que el género más frecuente es drama. Mientras los géneros menos frecuentes son news y short.



Además de eso, se encontró que en promedio las películas están relacionadas con 3 géneros. Mínimo las películas están relacionadas con un solo género y máximo hay 9 géneros relacionados por película.



Adicionalmente, se conocieron las palabras más frecuentes en la descripción de la película los cuales se visualizan en la siguiente gráfica. La mayoría de las palabras mas frecuentes no son muy dicientes, siendo conectores que no dan información para predecir el género.



3. Preparación de los datos

Con el fin de poder procesar la data primero fue necesario verificar si se presentaban valores nulos en la misma, los cuales después de una revisión se pudo observar que no hay valores nulos en la data.

Posteriormente, se procede a limpiar la misma y generar los monogramas para ello fue necesario crear la función "limpeza_y_generacion_monograma_y_bigrama" donde se debe ingresar como parámetro si se quiere generar la data a procesar como monograma o bigrama. Para explicar la función se mencionará cada una de sus características que fue aplicado a cada uno de los documentos:

- **Convertir a minúsculas:** con el fin de manejar la data siempre en un mismo formato y que el computador reconozca como mismo ítem a todas las palabras iguales sin importar si están en mayúscula o minúscula.
- **Tokenización:** mediante la función Word_tokenize se procede a tokenizar las palabras.
- **Puntuación:** Se procede a eliminar toda la puntuación que se encuentra dentro del documento con el fin de sólo procesar texto.
- **Stop words:** Se procede a eliminar los stopwords con la función stopwords.words en opción en inglés de NLTK.
- **Lematizar:** Se procede a lematizar las palabras con el fin de presentarlas en su forma canónica para ser analizadas.

Posteriormente se procede a generar los embeddings con los cuales los modelos podrán procesar los textos ya que solo con los tokens los modelos no serán capaces de realizar ninguna clasificación, para ello se procede a emplear distintas formas de embeddings de la data:

- **Bag of Words (BoW):** El código crea una lista de palabras en un documento y cuenta cuántas veces aparece cada palabra. Luego, convierte estos conteos en un conjunto de números que representan el documento.
- **Term Frequency-Inverse Document Frequency (TF-IDF):** Similar a BoW, pero en lugar de contar, evalúa la importancia de una palabra en un documento en

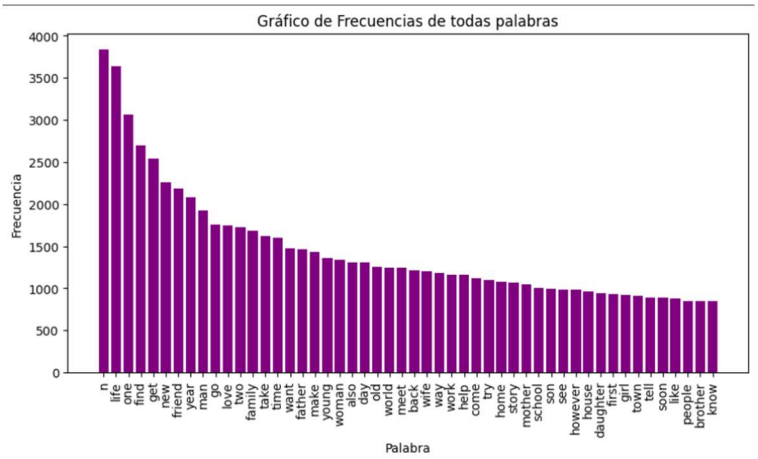
comparación con su importancia en todos los documentos. Esto ayuda a destacar palabras importantes.

- **Word2Vec:** Este método asigna palabras a vectores numéricos. Estos vectores capturan significados y relaciones entre palabras.

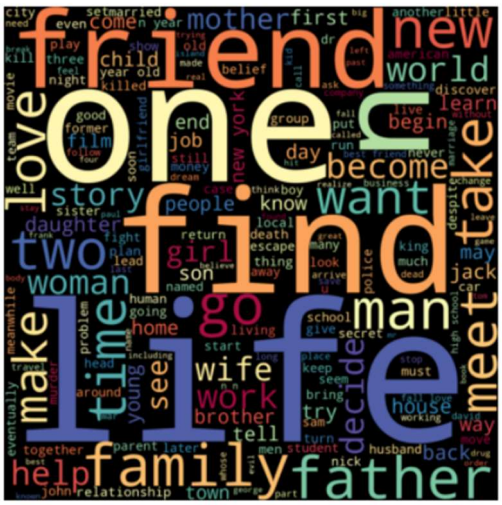
Estas codificaciones se utilizaron para evaluar los modelos, encontrando la mejor codificación.

4. Análisis de la data posterior a la preparación de los datos

Posterior a la limpieza de los datos, se genera nuevamente el gráfico que visualiza la frecuencia de las palabras de la descripción de las películas.



Por último, a continuación, se visualizan las palabras más frecuentes en la descripción de películas de una forma más intuitiva, siendo n, life, one y find las cuatro palabras más frecuentes.



5. Modelos

A partir de todo lo mencionado anteriormente, se procedió a realizar cuatro (4) modelos de Machine Learning con el fin de darle solución al problema en cuestión, los algoritmos empleados fueron los siguientes: Regresión Logística, XGBoost, Redes LSTM y GRU.

4.1. Regresión Logística

Se realiza una regresión logística optimizando los hiperparámetros con el valor de regularización, el algoritmo de optimización (solver), y el número máximo de iteraciones para el entrenamiento del modelo. Se realiza el entrenamiento del modelo utilizando los hiperparámetros definidos en el diccionario params, esto permite configurar la regresión logística con los hiperparámetros especificados.

Hiper-parámetros	Opciones del hiper-parámetro
Penalty	l2
Regularización	1
Optimización	Solver
Iteraciones	1000

```python
from sklearn.linear_model import LogisticRegression

# Hiperparámetros de la regresión logística
params = {
    'penalty': 'l2',
    'C': 10,
    'solver': 'lbfgs',
    'max_iter': 2000,
}

# Entrenar el modelo
clf = OneVsRestClassifier(LogisticRegression(**params))

clf.fit(X_train_dtm, y_train_genres)

# Realizar las predicciones
y_pred_genres = clf.predict_proba(X_test_dtm)

# Calcular la métrica ROC AUC
roc_auc = roc_auc_score(y_test_genres, y_pred_genres, average='macro')
print('ROC AUC Score:', roc_auc)
```

4.2. XGBoost

Para poder dar solución al problema se empleó un algoritmo de ensambles Extreme Gradient Boosting (XGBoost) el es una mejora al modelo de árboles y funciona bien con datasets grandes y complejos al utilizar varios métodos de optimización.

Para el caso en cuestión, se empleó el algoritmo en varias situaciones: mejorándole hiperparámetroS y dejándolo como modelo estándar. Para la mejora de hiperparámetros se utilizaron los siguientes:

Hiper-parámetros	Opciones del hiper-parámetro
Learning Rate	0.1, 0.2
Profundidad	5, None
Árboles	100, 200

Adicionalmente se empleó el algoritmo OneVSRest, debido a que se trataba de un problema multiclass y multilabel para que pudiera generar la solución al problema. A continuación, se presenta el código empleado.

```
def XGBoost_(params, X_train, y_train, X_test, y_test):
  xgb = OneVsRestClassifier(XGBClassifier(
      n_jobs=-1,
      random_state = 42
      ** params
      ))
  xgb.fit(X_train, y_train)

  y_pred_xgb = xgb.predict_proba(X_test)
  return y_pred_xgb, roc_auc_score(y_test, y_pred_xgb, average='macro')
```

Se aclara que para correr el modelo se empleó al algoritmo Bag of Words con 5000 palabras y todas las películas y se realizó una partición de test en un 33%.

4.3. LSTM

Una red neuronal LSTM es un tipo de red recurrente que mejoran la memoria a largo plazo de las mismas. Para la solución del problema fue empleada este tipo de red con dos soluciones: la primera intentaba calcular la una red por cada y que se tenía a clasificar, por su parte la segunda se empleó una única LSTM para dar solución al problema.

4.3.1 LSMT Versión 1

Para poder realizar esta versión de la LSTM inicialmente se realizó un balanceo de la data, para que el modelo aprendiera de igual manera los 1 y 0. Posteriormente, se creó una red que contenía una LSTM bi-direccional con 64 neuronas, seguida de una capa profunda de 32 neuronas, activación ReLu y regularizaciones dropout entre capas, finalmente una capa de salida de una única capa y activación sigmoide. Esto debido a que se va a iterar sobre cada columna y clasificar binariamente si cierta película clasifica o no para dicho género y se realiza un proceso iterativo sobre todos los géneros para poder dar la solución. La función de pérdida fue binary crossentropy, optimizador Adam y métrica de seguimiento accuracy. A continuación, se presenta el código de la red LSTM Versión 1.

```
embedding_layer = Embedding(len(vocabulary) + 1,
                            128,
    input_length=max_url_len,
    trainable=False)

descriptions = Input(shape=(max_url_len,), dtype='int32')

embedded_sequences= embedding_layer(descriptions)
embedded_sequences = BatchNormalization()(embedded_sequences)
embedded_sequences = SpatialDropout1D(0.3)(embedded_sequences)

x = Bidirectional(LSTM(64, return_sequences=True,kernel_initializer='glorot_uniform'))(embedded_sequences
x = GlobalMaxPool1D()(x)
x = Dropout(0.3)(x)

x = Dense(32, activation='relu')(x)
x = Dropout(0.3)(x)

preds = Dense(1,activation='sigmoid')(x)

model = Model(inputs=descriptions,outputs=preds)
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
model.summary()
```

Hiper-parámetros	Opciones del hiper-parámetro
Neuronas	LSTM 64 – FC 32
Activaciones	relu
Optimizador	adam

4.3.2 LSTM Versión 2

Para poder realizar esta versión no se realizó balanceo a la data. Esta LSTM contiene 2 redes LSTM bidireccionales de 128 neuronas cada una, posteriormente una capa oculta con 64 neuronas y otra capa oculta d 32 neuronas, estas capas ocultas con activación ReLu. Finalmente se tiene la capa de salida con 24 neuronas y activación softmax para poder evaluar entre las posibles soluciones que se tienen. Por último, este modelo tiene función de pérdida categorical crossentropy, activación Adam y evalúa el accuracy. A continuación, se presenta el código de esta versión de LSTM.

```
model = Sequential()
model.add(Embedding(len(vocabulary) + 1, 128, input_length=max_url_len))
model.add(SpatialDropout1D(0.2))
model.add(Bidirectional(LSTM(128,return_sequences=True)))
model.add(Dropout(0.3))
model.add(BatchNormalization())
model.add(Bidirectional(LSTM(128)))
model.add(Dropout(0.3))
model.add(BatchNormalization())

model.add(Dense(64,activation='relu'))
model.add(Dense(32,activation='relu'))
model.add(Dense(24, activation = "softmax"))
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

model.summary()
```

Hiper-parámetros	Opciones del hiper-parámetro
Neuronas	LSTM 128 – FC 64, 32
Activaciones	relu
Optimizador	adam

Se aclara que para ambas redes neuronales se les empleó el mismo embeding para poder inicializar la data.

4.4. GRU

Una red GRU (Gated Recurrent Unit) utiliza compuertas internas para aprender y recordar información importante en secuencias largas de datos, superando problemas de desaparición del gradiente, lo que la hace eficaz en tareas de procesamiento de lenguaje natural y series temporales.

Para este problema en particular, se empleó una red neuronal GRU bidireccional, con los siguientes hiperparámetros:

Hiper-parámetros	Opciones del hiper-parámetro
Neuronas	64
Activaciones	Softmax
Optimizador	Adam

```
model = Sequential()
model.add(Embedding(len(vocabulary) + 1, 128, input_length=max_url_len))
model.add(Bidirectional(GRU(64)))
model.add(Dense(24, activation='softmax'))

model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

model.summary()
```

Para darle los datos de entrada al modelo se lleva a cabo un procesamiento previo de estos, se identificaron los caracteres únicos en las secuencias de datos y se creó un diccionario de vocabulario para asignar un índice único a cada palabra. Además, se estableció una longitud máxima de secuencia de 300 palabras y se convirtieron las secuencias de palabras en representaciones numéricas utilizando el vocabulario definido. Posteriormente, se realizó una división de los datos en conjuntos de entrenamiento y prueba, asignando el 67% de los datos al conjunto de entrenamiento (X_train_GRU e y_train) y el 33% al conjunto de prueba (X_test_GRU e y_test).

6. Evaluación

Posterior a entrenar todos los modelos y validar su solución en el conjunto de entrenamiento, se realizó un ordenamiento de la data a partir del “AUC”, donde se logró encontrar que el mejor algoritmo fue la regresión logística con un valor de 0.877. A continuación, se presenta tabla con resumen de las mejores corridas de cada algoritmo.

Algoritmo	AUC
Regresión Logística	0.882
XGBoost	0.833
LSTM V1	0.500
LSTM V2	0.502
GRU	0.504

7. Conclusiones

Se realiza un modelo de procesamiento de lenguaje natural aplicado para predecir el género de películas que utiliza técnicas como lematización, eliminación de stopwords, eliminación de puntuación y una representación TF-IDF (Term Frequency-Inverse Document Frequency). Como resultado se obtiene un valor de AUC (Área bajo la Curva ROC) de 0.877, es un resultado bastante sólido con un buen desempeño en la clasificación de géneros de películas. A continuación, se detallan las conclusiones sobre el resultado del modelo:

Métrica del Modelo: EL resultado de AUC del 0.877 indica que el modelo es capaz de distinguir efectivamente entre géneros de películas a partir de la aplicación de una regresión logística optimizando los hiperparámetros.

Calidad de las Características: Las técnicas de preprocesamiento de texto aplicadas como la lematización, eliminación de stopwords, eliminación de puntuación, dejar palabras en minúsculas y la representación TF-IDF, son efectivas en la generación de características para la clasificación de géneros de películas. Estas técnicas han ayudado a reducir el ruido y a resaltar las características relevantes en el texto.

Word2Vec se utiliza para capturar relaciones semánticas entre palabras y es útil en tareas de procesamiento de texto más complejas, mientras que TF-IDF se utiliza para resaltar palabras clave y es más adecuado para clasificación de documentos, en este caso TF-IDF tiene mejor rendimiento para predecir el género de las películas.

Oportunidad de Mejora: Aunque un AUC de 0.877 es un buen resultado ya que los modelos tienen ajuste de hiperparámetros, siempre es posible seguir mejorando el modelo aplicando otras técnicas de preprocesamiento de texto probando otros algoritmos de clasificación o incorporar información adicional como metadatos de películas, para mejorar aún más la precisión del modelo.

Aplicaciones Empresariales: Se considera la implementación en una aplicación práctica, como un sistema de recomendación de películas o una herramienta de etiquetado automático de géneros. Como se mencionó en la sección de business understanding, aplicar este tipo de soluciones en la industria del entretenimiento trae ventajas tanto para los usuarios como para las empresas, ya que para los primeros se genera una reducción de tiempo al facilitar la búsqueda de películas y para los segundos, es una oportunidad para entregar al cliente una mejor experiencia, lo que puede ser un factor para aumentar las ventas.