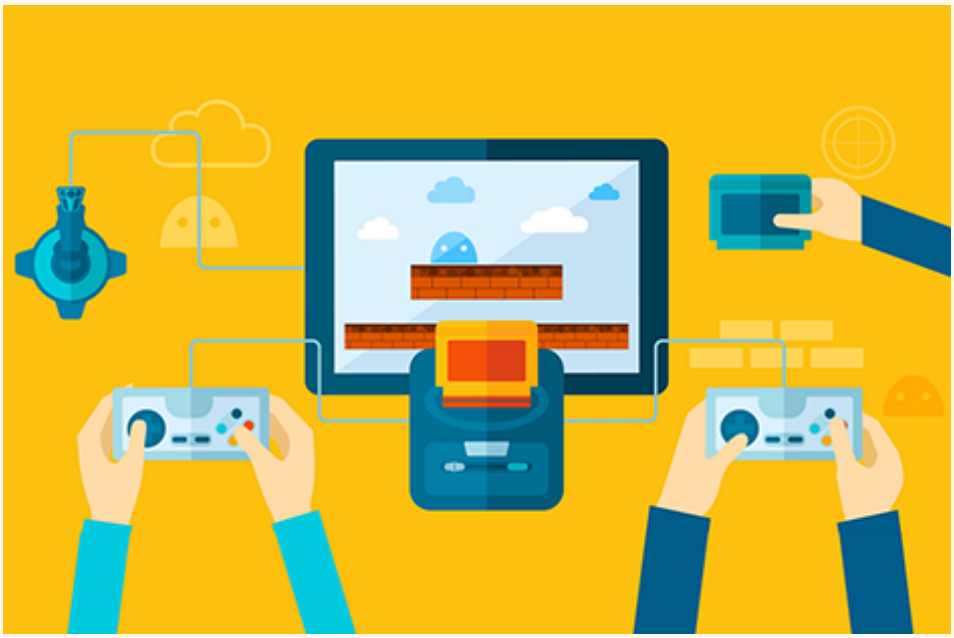
TRABAJO FINAL DATA SCIENCE
CODERHOUSE

STEAM:
ANALISIS Y PREDICCION DE SENTIMIENTO EN RESEÑAS DE VIDEOJUEGOS

—

Ignacio Vallecillo - Francisco Lopez Ballent - Gerardo Heidel



Fecha de entrega: 12 de Octubre de 2022

INTRODUCCIÓN

En el presente trabajo se mostrará todo el proceso de desarrollo de un modelo de Machine Learning capaz de predecir el sentimiento de reviews de videojuegos utilizando el lenguaje de programación Python y todas sus librerías asociadas al manejo y visualización de datos.

Objetivo

El objetivo de éste trabajo es generar un modelo de Machine Learning capaz de hacer un análisis de sentimiento de una review o comentario sobre un videojuego y determinar si el sentimiento del mismo es positivo o negativo.

Metodología

El trabajo se divide de acuerdo a las distintas etapas que tiene la creación de un modelo de Machine Learning estas son:

1. **Data Acquisition**

   Recopilación de los datos que van a ser utilizados para conformar el Dataset.

2. **Preprocesamiento de los datos**

   Preprocesamiento de los datos que conforman el Dataset. Consiste en hacer una limpieza de los datos para mejorar la performance del modelo.

3. **Análisis exploratorio de los datos (EDA)**

   Análisis de los datos, univariado, bivariado y multivariado. Esto nos permite conocer más en profundidad la data que vamos a utilizar.

4. **Creación del modelo de ML**

   En esta etapa se genera el modelo de ML probando distintos modelos que se ajustan a nuestro caso de estudio.

5. **Optimización del modelo**

   Aquí buscamos mejorar la performance de los modelos realizando ajustes en los hiperparámetros de los mismos.

6. Comparación de modelos

Comparar los distintos modelos propuestos para quedarnos con el mejor.

7. Evaluación del modelo

Evaluar la performance del modelo de acuerdo a distintas métricas.

1. Data Acquisition

Para la conformación de nuestro Dataset elegimos utilizar dos Dataset ya armados descargados de la plataforma Kaggle. Uno que contiene reviews con su respectivo sentimiento y otro con información de los juegos. Si bien solo con el Dataset de las reviews ya nos alcanzaba para nuestro objetivo, decidimos sumar información de los juegos para enriquecer el análisis. Estos Datasets tienen las siguientes características:

Steam reviews

Este Dataset contiene 6.4 Millones de reviews de videojuegos extraídas de la plataforma Steam.

URL: https://www.kaggle.com/datasets/andrewmvd/steam-reviews

Features:

- **app_id**: ID único para cada título.
- **app_name**: Nombre del juego.
- **review text**: Contenido de la review.
- **review score**: Valoración de la review (1: Positiva, -1: Negativa)
- **review_votes**: Valoración de otros usuarios a la review

Steam Store Games

Dataset que contiene información general sobre más de 27.000 juegos, extraída de la plataforma de Steam.

URL: https://www.kaggle.com/datasets/nikdavis/steam-store-games

Features:

- **release_date**: Formato Año-Mes-Dia
- **english**: Si el idioma es ingles
- **developer**: Empresa o persona que desarrolló el juego
- **publisher**: Nombre o nombres de los publicantes
- **platforms**: Lista de Sist. Op. soportados por el juego
- **required_age**: Mínimo de edad requerida según PEGI UK standards
- **categories**: lista de categorías del juego.
- **genres**: Lista de géneros del juego.
- **price**: Precio actualizado del juego en libras esterlinas

1.1 Preparación del Dataset

Debido al gran tamaño de los Dataset se decidió tomar una muestra representativa. Para esto, del Dataset "Steam Reviews", que es el que tiene las variables que nos interesan, se tomó una muestra de 50.000 reviews positivas y 50.000 reviews negativas, de esta manera achicamos el tamaño y balanceamos el Dataset.

Luego con la muestra ya conformada, realizamos un merge con el otro Dataset uniendolos según el app_id.

1.2 Características del Dataset

Con el Dataset final ya conformado, estas son algunas de las características del mismo:

- Cantidad de filas: 90.062
- Cantidad de columnas: 18
- Cantidad de juegos únicos: 5.303

En cuanto a los missing values esta es la distribución de los mismos en porcentaje para cada una de las columnas del Dataset.

```
app_id          0.000000
app_name        1.445671
review_text     0.089938
review_score    0.000000
review_votes    0.000000
```

```
release_date    0.000000
developer       0.000000
publisher       0.000000
platforms       0.000000
required_age    0.000000
categories      0.000000
genres          0.000000
price           0.000000
```

Vemos que las únicas columnas que tiene missing values es la de app_name y review_text, el tratamiento de estos valores se cubre más adelante en la sección de Preprocesamiento de los datos.

1.3 Data types

Veamos cuáles de las variables son categóricas y cuales son numéricas.

```
app_id           int64
app_name        object
review_text     object
review_score     int64
review_votes     int64
release_date    object
developer       object
publisher       object
platforms       object
required_age     int64
categories      object
genres          object
price          float64
```

2. Preprocesamiento de los datos

En esta sección veremos todo el proceso de limpieza que se le aplicó a la data. En nuestro caso, como nuestra variable principal es el texto que conforma la review, gran parte del procesamiento fue aplicado a dicha variable.

2.1 Missing values

Como vimos anteriormente tenemos missing values en las columnas de app_name y review_text, procedimos a eliminar solo aquellos registros que tenían missing values en la columna de review_text. Esta forma de lidiar con los missing values se llevó a cabo debido a que el porcentaje de registros con valores nulos era muy bajo (0.089%) y a su vez, la variable de review_text para nuestro análisis resulta fundamental por lo tanto no era tolerable tener valores nulos.

2.2 Limpieza del Dataset

Al imprimir algunos valores random de la variable review_text nos encontramos con la particularidad de que se repetía frecuentemente la frase “Early Access Review”.

```
1 -> Just the right amount of creepy.
2 ->  Early Access Review
3 ->  Early Access Review
4 ->  Early Access Review
5 -> this game sucks, but is the best at the same time
6 -> That game gave me a weird impression of a deja-vu.  Meh/10
7 -> its rather good
8 -> Story, Music, Narration. 10/10
9 -> Its a buggy mess, even after all this time.
10 ->  Early Access Review
```

Ante esta situación, nos fijamos en primer lugar, qué porcentaje de reviews tenemos con esta frase. Resultó ser el 17,9%. Como es un porcentaje alto, analizamos si este texto tenía alguna relación con el sentimiento de la review, para entender si nos aporta alguna información. Para esto, calculamos el porcentaje de reviews positivas y negativas con el texto “Early Access Review”.

```
% Positivas -> 46.92
% Negativas -> 53.08
```

Cómo está relativamente balanceado, podemos inferir que no hay una tendencia a ser calificadas como positivas o negativas por ende no aportan información adicional y pueden ser eliminadas.

2.3 Preprocesamiento del texto

En los modelos de NLP es muy importante realizar un correcto preprocesamiento del texto con el que se va a entrenar al modelo, existen muchas técnicas y procesos que se pueden aplicar, en nuestro caso decidimos aplicar lo siguiente:

- Hyperlinks: Eliminamos todos los links que aparecen en las reviews por ejemplo https://youtube.com...
- Puntuación: Se eliminan todos los símbolos de puntuación ya que no aportan información sobre el sentimiento.

Además de estas consideraciones se decidió eliminar todas las reviews que tenían más de 150 palabras por dos razones, en primer lugar porque, como vamos a ver más adelante, la distribución de cantidad de palabras por review da que la gran mayoría de las reviews tienen entre 0 y 50 palabras y en segundo lugar, porque más adelante se va aplicar Sentence Embedding mediante un modelo pre-entrenado y el mismo requiere que los inputs tengan como máximo 200 palabras.

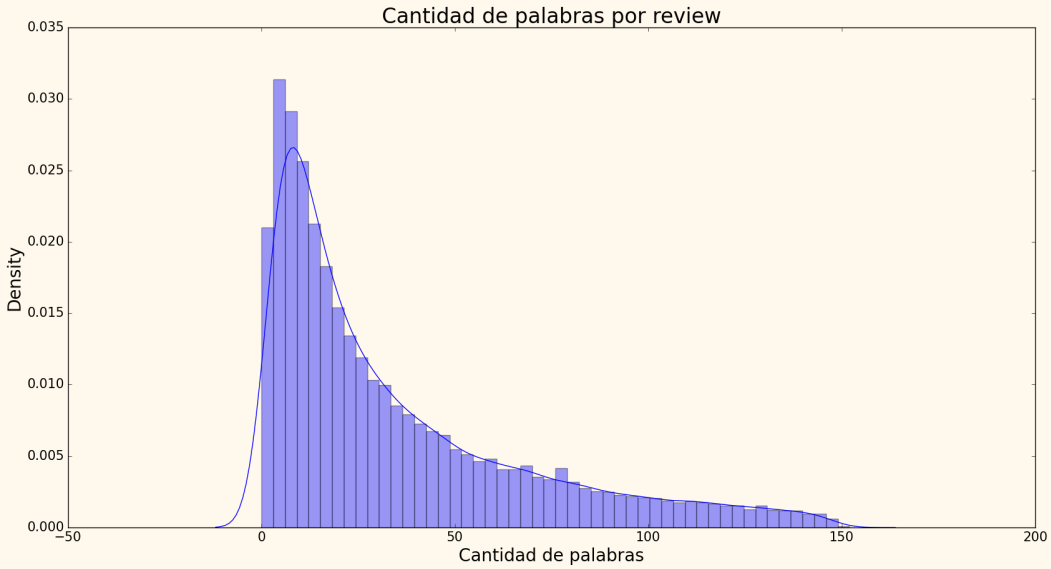
3. Análisis exploratorio de los datos (EDA)

Este análisis decidimos dividirlo en dos grandes grupos, por un lado, el análisis de las variables principales (review_text y review_score) y por el otro lado el análisis de las variables secundarias (price, developer, genre, etc.).

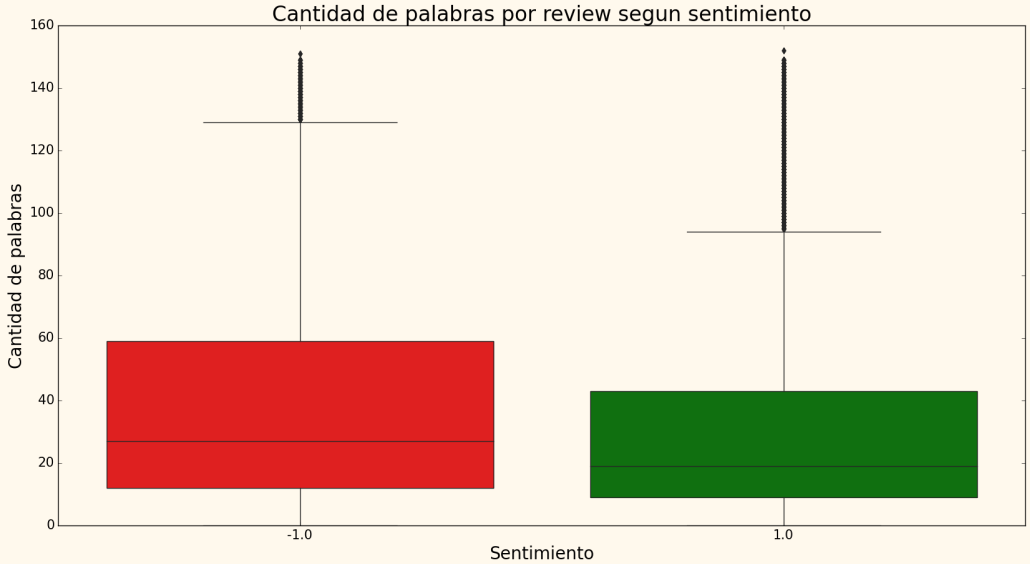
3.1 Analisis de variables principales

3.1.1 Cantidad de palabras por review

Veamos cómo se distribuye la cantidad de palabras por review, se hizo un conteo de la cantidad de palabras en cada review y se graficó la distribución.



Si ahora dividimos en reviews positivas y negativas veamos como es la distribución para cada caso en un boxplot.



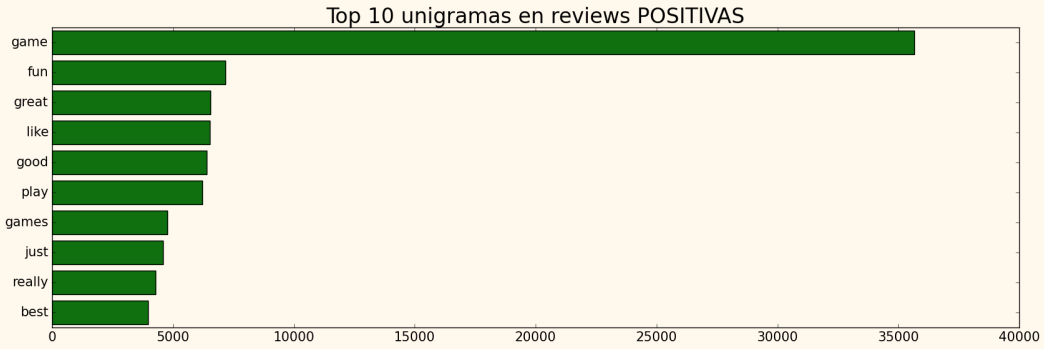
3.1.2 N-Grams

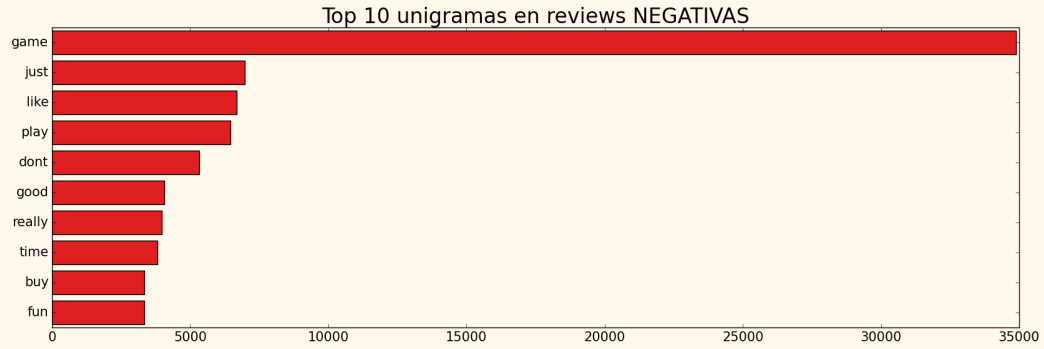
Analicemos ahora cuales son las palabras y frases más utilizadas en las reviews dividiéndolas en positivas y negativas. N-Gram hace referencia a la cantidad de palabras que se toman en

consideración para el análisis de frecuencia, Unigramas muestra las palabras individuales más usadas, Bigramas muestra las frases compuestas de 2 palabras más utilizadas, y así.

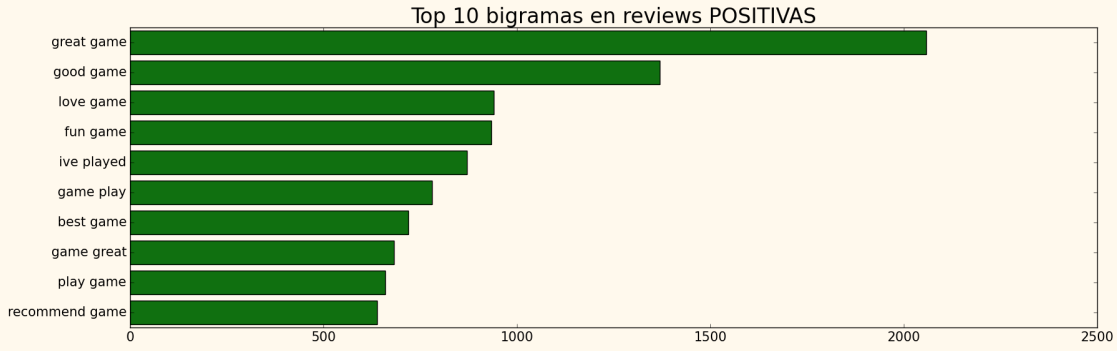
A continuación se muestran los Unigramas. Bigramas y Trigramas más utilizados.

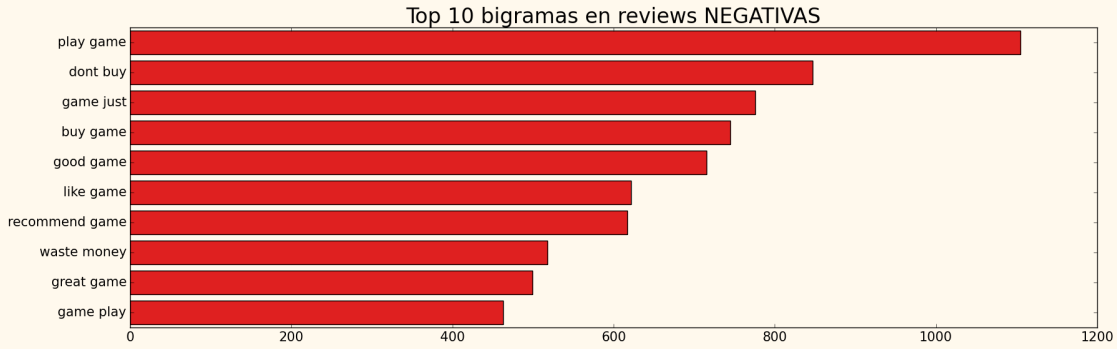
Unigramas



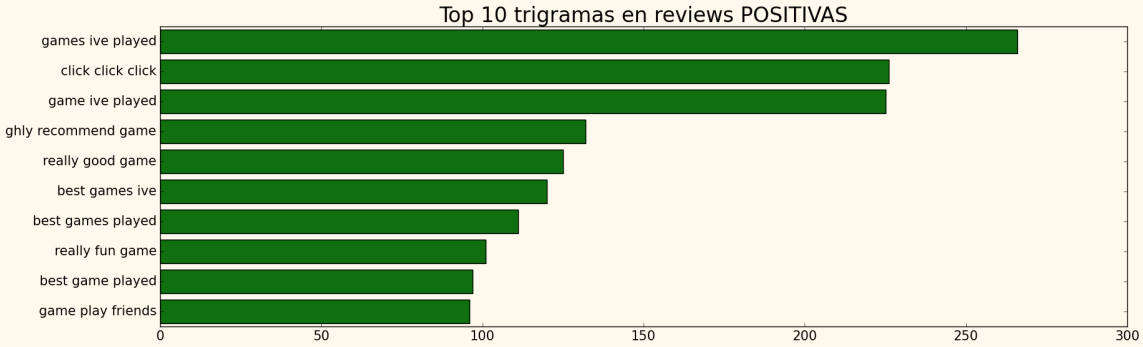


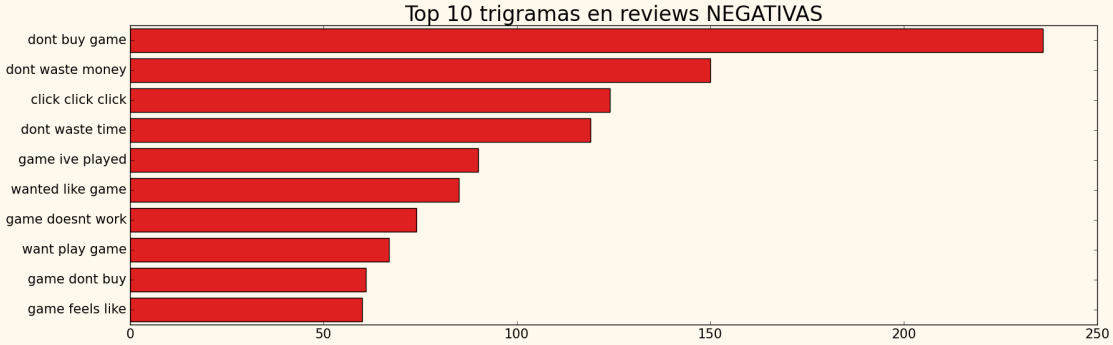
Bigramas




Top 10 bigramas en reviews NEGATIVAS
play game
dont buy
game just
buy game
good game
like game
recommend game
waste money
great game
game play
0
200
400
600
800
1000
1200

Trigramas


Top 10 trigramas en reviews POSITIVAS
games ive played
click click click
game ive played
ghly recommend game
really good game
best games ive
best games played
really fun game
best game played
game play friends
0
50
100
150
200
250
300


Top 10 trigramas en reviews NEGATIVAS
dont buy game
dont waste money
click click click
dont waste time
game ive played
wanted like game
game doesnt work
want play game
game dont buy
game feels like
0
50
100
150
200
250

3.1.3 WordClouds

Los Word Clouds son una forma más gráfica y "bonita" de ver cuales son las palabras más utilizadas en nuestro conjunto. Decidimos generar un Word Cloud para las reviews positivas y otro para las negativas

3.2 Análisis de variables secundarias

Con estas variables se decidió hacer simplemente un análisis básico y descriptivo de las mismas ya que no aportan al objetivo del trabajo.

3.2.1 Género

Cuales son los géneros que aparecen con más frecuencia en nuestro conjunto de reviews.

```
TOP 10 GENEROS QUE MAS SE REPITEN EN EL DATASET

Indie           2689
Action          2093
Adventure       1568
Casual          1052
Strategy        1047
RPG              845
Simulation       783
Free to Play     365
Racing           189
Sports           173
```

3.2.2 Publisher

Cuales son las empresas o personas con más publicaciones de juegos en nuestro conjunto.

```
TOP 10 COMPAÑÍAS CON MÁS PUBLICACIONES EN EL DATASET

Ubisoft                75
Square Enix            65
THQ Nordic             64
Strategy First         54
SEGA                   52
1C Entertainment       45
Paradox Interactive    42
Disney Interactive     41
Devolver Digital       39
2K                     38
```

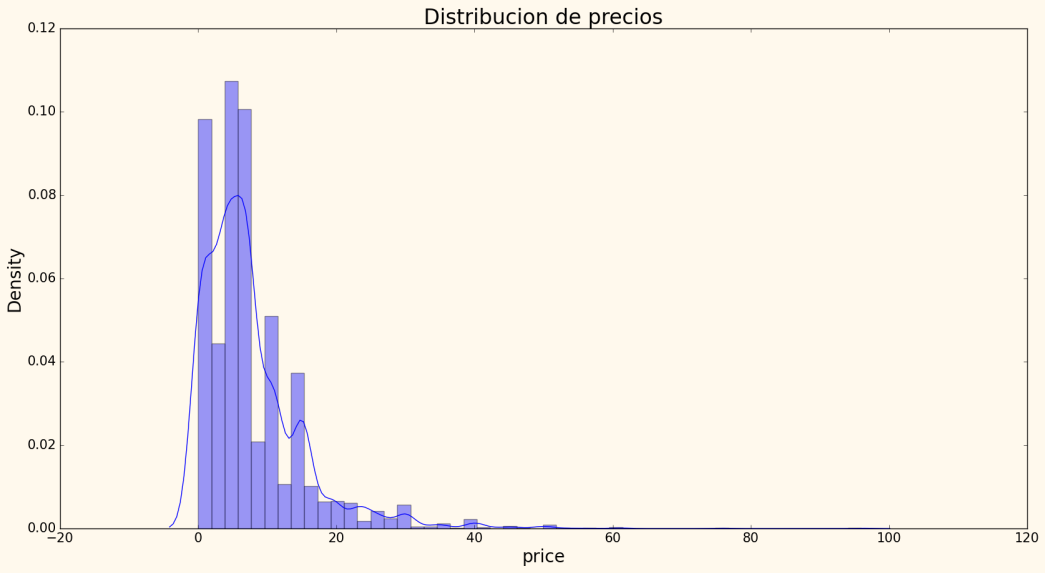
3.2.3 Price

Veamos como es la distribución de los precios de los juegos que aparecen en nuestro Dataset.

Comencemos viendo una descripción de la variable.

```
count    4619.00
mean        7.62
std         7.47
min         0.00
25%         2.79
50%         5.99
75%        10.99
max        95.99
```

Y ahora la distribución de los precios.



4. Creación del modelo de Machine Learning

A continuación buscaremos crear nuestro modelo de Machine Learning evaluando distintos modelos que se ajustan a nuestro caso. El problema que enfrentamos se clasifica como supervisado de clasificación, para estos casos existen múltiples modelos que se pueden utilizar, nosotros elegimos avanzar con los modelos: Random Forest, LightGBM y XGBoost.

Al enfrentarnos a un caso de NLP es necesario tokenizar nuestra variable independiente, es decir, transformar al texto en vectores numéricos para que pueda ser interpretado por los modelos de ML. Una técnica que se utiliza para esto es el proceso de Sentence Embedding, este proceso transforma las sentencias, en nuestro caso las reviews, en vectores de N elementos que guardan información sobre la semántica de las palabras a diferencia de los procesos de tokenización como Bag of Words que solo contabilizan la cantidad de veces que aparece una palabra en una sentencia.

4.1 Sentence Embedding

Si bien este proceso puede hacerse *manualmente* con la data que tenemos, existen modelos pre-entrenados que nos facilita considerablemente el trabajo, mejorando los tiempos de procesamiento y los resultados.

Para este caso decidimos utilizar el modelo BERT (Bidirectional Encoder Representations from Transformers), es un modelo de redes neuronales entrenado con millones de archivos de texto que logra captar la semántica de los inputs y devuelve como resultado vectores que nos permite realizar cálculos y operaciones sobre los mismos. Por ejemplo, si queremos saber qué tan similares son dos frases, al tenerlas en forma vectorial, podemos calcular la distancia entre un vector y otro (una frase y otra) y de esta manera, cuanto más chica sea esa distancia, más parecidas semánticamente son esas frases. *Este juego me pareció increíble* va a estar en un lugar del espacio vectorial cercano a *Este juego me pareció buenisimo*.

El resultado de este proceso es una matriz con tantas filas como reviews y, en este caso, por el modelo que utilizamos, 767 columnas. Esta matriz pasa a ser nuestro Dataset.

4.2 Entrenamiento de modelos

Como mencionamos anteriormente, vamos a evaluar los modelos de Random Forest, LightGBM y XGBoost, para esto, dividimos a nuestro dataset en dos conjuntos, uno de training y otro de testing, el conjunto de training lo usamos para entrenar a los modelos y el de testing para evaluar la performance de los modelos. A continuación, se muestran los resultados de accuracy obtenidos tanto en el conjunto testing y el conjunto training para los 3 casos utilizando los modelos con los hiperparametros que vienen por default. Esto de evaluar con los dos conjuntos lo hacemos para detectar posible overfitting.

Modelo	Accuracy TEST	Accuracy TRAIN
Random Forest	0.7847	0.9947
LightGBM	0.8017	0.8665
XGBoost	0.7839	0.8117

Del análisis de estos resultados se desprenden algunas conclusiones:

- El modelo de Random Forest tiene una gran diferencia entre los dos conjuntos, esto nos dice que el modelo está haciendo overfitting.
- Considerando el accuracy en test el que obtuvo mejores resultados fue LightGBM pero si ponemos en juego el problema de overfitting XGBoost, si bien tuvo un accuracy en test más bajo, la diferencia con el de train es levemente menor que en el de LightGBM.

Veamos ahora qué pasa cuando optimizamos a los modelos realizando ajustes en sus hiperparámetros.

5. Optimización del modelo

En esta sección, vamos a desarrollar el método que utilizamos para optimizar los modelos. Existen varios métodos para este propósito: Grid Search CV, Randomized Search CV, Optimizacion Bayesiana, entre otros, para este caso decidimos utilizar Randomized Search CV debido a la rapidez y efectividad del mismo.

El método elegido consiste en generar un espacio de hiperparámetros del cual el algoritmo va a seleccionar de manera aleatoria distintos valores para entrenar al modelo. Es necesario especificar la cantidad de iteraciones que queremos que haga el algoritmo y el número de Cross Validations. Por cada iteración el modelo va a seleccionar un conjunto de hiperparametros del espacio, y va a entrenar al modelo y validarlo N veces, donde N es el número de Cross Validations, así hasta completar el número de iteraciones dado.

Por una cuestión de tiempos de procesamiento decidimos optimizar solo los modelos de XGBoost y LightGBM ya que el de Random Forest demoraba muchas horas en hacer la optimización.

Por lo tanto en la comparativa final de modelos se considerarán 5 modelos:

- Random Forest
- LightGBM
- XGBoost
- LightGBM Optimizado
- XGBoost Optimizado

6. Comparación de modelos

6.1 Accuracy, precision y recall

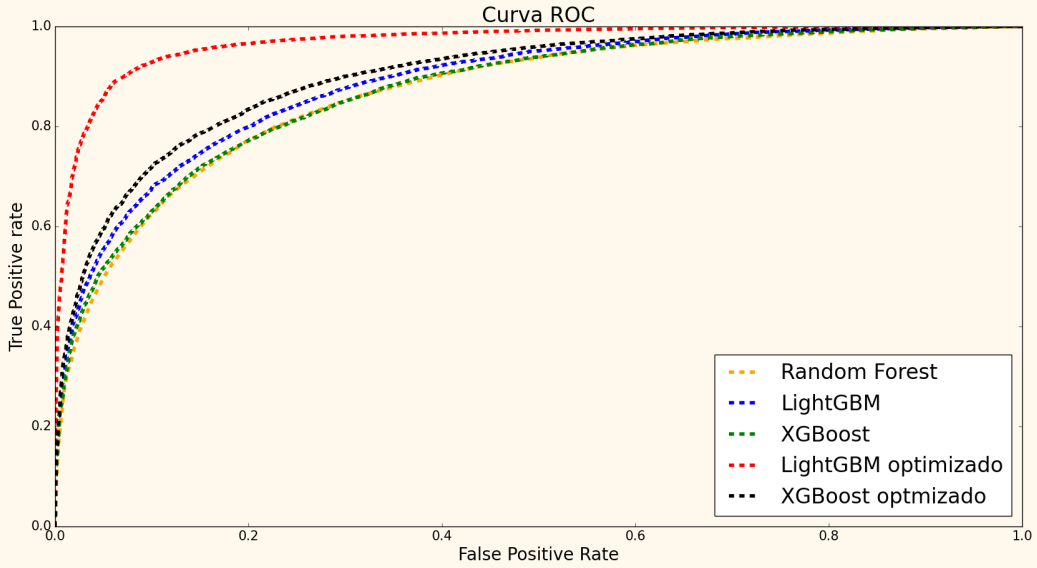
Veamos los resultados obtenidos luego de la optimización, se mostrarán los resultados de los 5 modelos.

Se agrega a su vez, además del accuracy de train y test ya utilizados anteriormente, métricas como la precisión y el recall que nos ayudan a evaluar la performance de los modelos.

Modelo	Accuracy TEST	Accuracy TRAIN	Precisión	Recall
Random Forest	0.784734	**0.994774**	0.783862	0.828479
LightGBM	0.801742	0.866506	0.810151	0.824957
XGBoost	0.783965	0.811769	0.790003	0.815344
LightGBM Opt	**0.915574**	0.916983	**0.919651**	**0.923853**
XGBoost Opt	0.818443	0.878164	0.826609	0.838568

En la tabla se observa los excelentes resultados obtenidos por el modelo de LightGBM optimizado arrojando no solamente valores de accuracy muy altos sino que muy parecidos entre el conjunto train y el test por lo que no hay riesgo de overfitting.

6.2 Curva ROC



Éste gráfico nos ayuda a identificar cual es el método que está devolviendo mejores resultados, En la curva ROC, un modelo es mejor cuanto mayor sea el área bajo la curva correspondiente al modelo, este valor se conoce como AUC. Se observa claramente como la curva roja (LightGBM optimizado) es la que tiene mayor AUC.

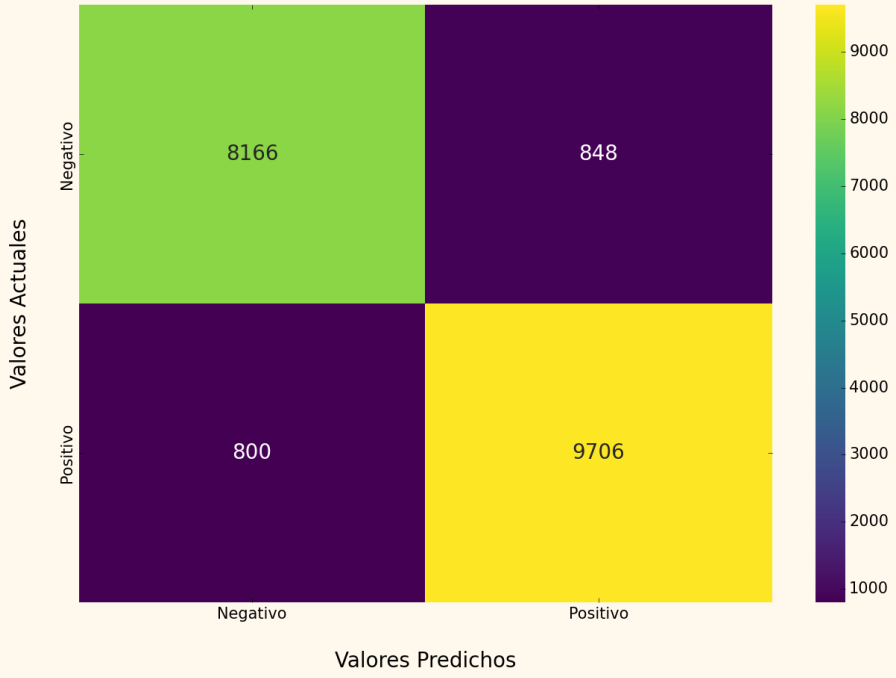
A partir de todo lo dicho anteriormente, podemos determinar que el modelo con la mejor performance y por ende el que vamos utilizar como nuestro modelo es el de **LightGBM optimizado.**

7. Evaluación del modelo

Para terminar el análisis, realizamos una evaluación del modelo seleccionado agregando algunas métricas además de las ya utilizadas anteriormente.

7.1 Matriz de confusión

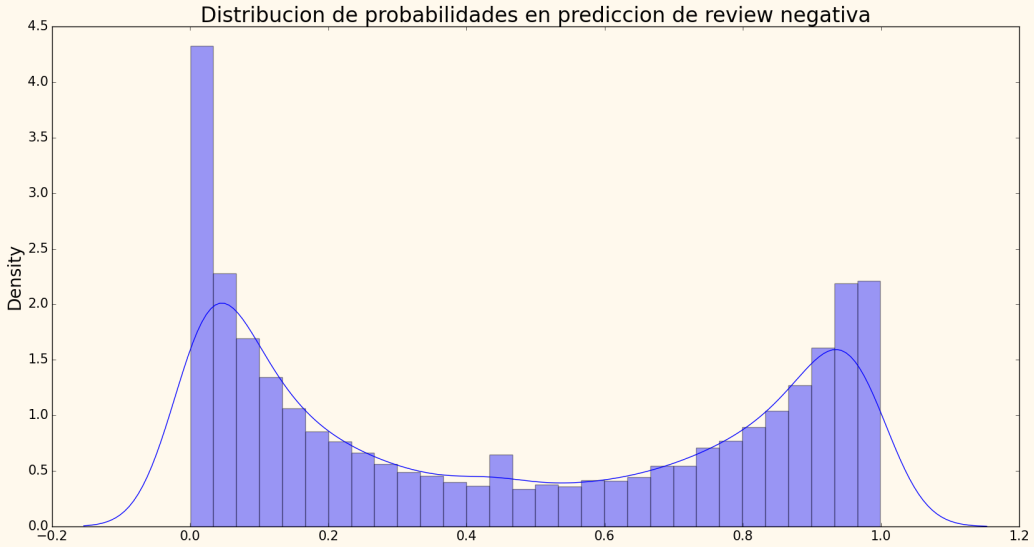
La matriz de confusión nos permite observar gráficamente la performance del modelo. Podemos detectar fácilmente si tiene una tendencia a arrojar falsos positivos (FP) o falsos negativos (FN).



Podemos observar que los valores que obtuvimos son muy buenos, menos de un 10% de FP y FN y no solo eso sino que ambos valores son muy similares por lo que no hay una tendencia a arrojar uno sobre el otro. En resumen lo que obtuvimos es lo siguiente:

- Verdaderos positivos: 9706 (49,7%)
- Verdaderos negativos: 8166 (41,8%)
- Falsos positivos: 848 (4,3%)
- Falsos negativos: 800 (4,1%)

7.2 Distribución de probabilidades de predicción



El modelo internamente establece una probabilidad de que la review analizada sea positiva o negativa, la probabilidad más alta entre las dos es la que va a determinar el Score predicho, es decir, si la predicción de una review devuelve que tiene un 80% de probabilidad de ser negativa y un 20% de probabilidad de ser positiva, la predicción va a decir que la review es negativa.

Este gráfico nos muestra cómo es la distribución de la probabilidad de que una review sea negativa. Se muestra solo la negativa porque la positiva es idéntica pero espejada (P_Pos = 1 - P_neg).

Considerando esto, podemos determinar que el modelo aprendió a detectar con mayor claridad a las review positivas, esto se lee del pico que tenemos en 0 que son todos los casos donde habia 0% de probabilidad de que la review sea negativa y por ende un 100% de probabilidad de que sea positiva cosa que no pasa con tanta frecuencia con las reviews negativas.

7.3 Testing del modelo

A continuación mostraremos algunos ejemplos de predicciones con una selección de reviews random del conjunto de testing. Se muestra el texto de la review, el Score que el usuario le dio a su review (1 para positivas y -1 para negativas) y la predicción del modelo mostrada como

probabilidad. Va a mostrar que probabilidad hay de que la review sea positiva y que probabilidad hay de que la review sea negativa, el mayor de los dos va a determinar el Score predicho.

```
still feels like it has a lot of bugs
Score: -1
Probabilidad Negativa: 0.867944674681083
Probabilidad Positiva: 0.13205532531891698

pretty big game like mandy lol ign would like bully too
Score: 1
Probabilidad Negativa: 0.16196138081546818
Probabilidad Positiva: 0.8380386191845318

not worth your time its one battle worthless voice acting no real plot one of the
most pathetic things ive ever seen
Score: -1
Probabilidad Negativa: 0.9067791851970665
Probabilidad Positiva: 0.09322081480293354

hilariously glitchy and just plain fun
Score: 1
Probabilidad Negativa: 0.3844595676605228
Probabilidad Positiva: 0.6155404323394772

this is awesome but sometimes we got server problem but at least this greatest game
i ever played
Score: 1
Probabilidad Negativa: 0.32067260462911584
Probabilidad Positiva: 0.6793273953708842
```

8. Conclusiones

A lo largo del desarrollo del trabajo fuimos aprendiendo y llegando a algunas conclusiones que las compartimos a continuación.

Sobre la conformación del dataset

Cuando arrancamos el trabajo, nos enfrentamos con el gran volumen de datos que habíamos encontrado y con el tiempo y costo computacional que implica procesar semejante cantidad de datos con el agravante de que nuestra variable principal son cadenas de texto de más de 50 palabras en muchos casos. Comenzamos tomando una muestra de 200.000 reviews sobre las 6.400.000 originales. Por más que el porcentaje elegido no llegaba al 5% nos fue necesario incorporar procesamiento de GPU mediante Cudas para lograr procesar los datos.

Con el tiempo y la ayuda de los tutores descubrimos que tomando muestras aún más chicas, de 100.000 reviews aproximadamente, los resultados se mantenían sin la necesidad de sumar procesamiento por GPU.

Por lo tanto, resulta muy importante tomarse un tiempo al iniciar un proyecto como estos para analizar el tamaño del Dataset que vamos a utilizar, no es necesario siempre usar el total de los datos, a veces con un correcto muestreo podemos ahorrarnos mucho tiempo y recursos.

Sobre el uso de BERT para Sentence Embedding

Como explicamos en el desarrollo del trabajo, en NLP es importante preprocesar el texto que se va a usar para entrenar el modelo, parte del preprocesamiento que habíamos utilizado en un principio, incluía el proceso de Stemming/Lemmatization que en pocas palabras buscan llevar a las palabras a la raíz de las mismas, por ejemplo: *studies*, *studying, study* son simplificadas a *studi* en el caso que apliquemos Stemming. Este proceso, como vemos en el ejemplo, simplifica a las palabras en palabras que no tienen sentido como *studi* lo cual descubrimos que era un problema al momento de aplicar el modelo BERT.

El modelo BERT es un modelo de redes neuronales pre-entrenado con toda la Wikipedia y cientos de libros, por lo tanto, fue entrenado con texto natural, sin preprocesamiento, esto quiere decir que si nosotros le pasamos texto preprocesado con palabras que no tienen sentido como *studi* la performance del modelo no es buena. Esto lo comprobamos comparando el

accuracy de los modelos de ML cuando usábamos data no preprocesada y cuando usábamos data preprocesada, la diferencia en algunos modelos llegaba casi al 10% a favor del caso con data sin preprocesar. Por lo tanto, terminamos decidiendo usar un preprocesamiento tal que no afecte a la performance de BERT.

Sobre los resultados obtenidos y el recorrido hasta conseguirlos

Desde que comenzamos con este proyecto, una idea fija teníamos en la mente, conseguir un modelo capaz de predecir con la mayor exactitud posible el sentimiento de las reviews que le entregabamos. En los primeros intentos, trabajando con 200.000 reviews, aplicando métodos de stemming, tokenización y embedding, y saliendo de las enseñanzas del curso, nos propusimos a incurrir en el Deep Learning, nos llevó bastante tiempo entender un poco este mundo pero logramos un modelo capaz de predecir en un 79%, lo que no nos convencía del todo. Pero sin desistir de la meta y con guía de nuestro tutor Manuel Sosa, nos ofreció la idea de simplificar las cosas, reducir la muestra del dataset a la mitad y aplicar modelos de Machine Learning, lo que nos llevó a mejor puerto. Primero obteniendo resultados cada vez más alentadores, con modelos de XGBoost y LigthGBM que predecían entre un 81 a 82%, hasta que aplicando optimizaciones a los mismos a través de la implementación de Randomized Search CV, se logró conseguir un modelo de LigthGBM optimizado capaz de predecir con un Accuracy del 91% el sentimiento de las reviews. Superando ampliamente nuestras expectativas y cumpliendo así nuestra meta inicial.

9. Futuras Iniciativas

Desde un principio, la visión del producto fue la de crear un bot de Twitter que pueda generar recomendaciones / analizar reseñas de películas basándose en el sitio IMDB. Si bien durante el proceso de desarrollo de los modelos se cambió el objetivo a reseñas de juegos dentro de Steam, la visión original se mantiene.

Posibles Aplicaciones:

- **Twitter bot:** Un bot de Twitter que haga crawling sobre twitter y prediga el sentimiento de una opinión o reseña de un usuario sobre un juego o grupo de juegos, utilizando Hashtags populares que hagan referencia a estos juegos (#Destiny2, #GenshinImpact, etc)
- **API Pública:** Una API de acceso público que permita a los usuarios (Developers) ingresar reseñas y devolver un análisis de sentimiento sobre las mismas.
- **Monitoreo de Bugs en juegos:** Si bien el alcance actual del proyecto no cubre la identificación de palabras claves para comprender el uso de frases o palabras claves para saber si una reseña incluye mención de bugs, es algo que hemos detectado al observar los datos. Reseñas negativas tienden a incluir referencias a bugs que afectan a la experiencia del usuario.