**Proyecto de evaluación Actividad 3**

Logotipo

Descripción generada automáticamente

**Estudiante: Juan Carlos González Ibarra**

**Correo:** [**jcarlos0284@gmail.com**](mailto:jcarlos0284@gmail.com)

**Fecha: 17 de mayo, 2024**

**Contenido**

1. Tema
   1. Planteamiento del problema
   2. Objetivos
2. Metodología de Desarrollo
3. Extracción de Datos

3.1 Fuentes de extracción de datos

3.2 Que es la extracción de datos

3.3 Extracción de datos con Python

1. Data Clean

4.1 Descripción de los datos

4.2 Carga de Datos

4.3 Preprocesar los datos

4.3.1 Fusionar data frames

4.3.2 Eliminar variables irrelevantes

4.4 Definición de la(s) Variable(s) Objetivo(s).

4.5 Almacenamiento de data frame en base de datos relacional

1. Modelado de Datos

5.1 Tipos de Modelos

5.2 Métodos de Modelado

5.3 Modelo No Supervisado

5.4 Modelo Supervisado

5.5 Proceso Modelado de Datos

5.6 Modelo de Clasificación

5.6.1 Resultados del Modelo de Clasificación

5.7 Modelo de Regresión

5.7.1 Entrenamiento del modelo

5.7.2 Validación del modelo

5.7.3 Resultados del Modelo de Regresión

5.8 Modelo de Regresión Logística

5.8.1 Entrenamiento del modelo

5.8.2 Validación del modelo

5.8.3 Resultados del Modelo de Regresión Logística

5.9 Predicción y toma de decisiones

5.9.1 Guardar el modelo

5.10 Aplicación Web de Predicción

5.10.1 Desarrollo

**5. Modelado de Datos**

Después de tener una limpieza de datos y organizadas las variables que se utilizaran para crear un ssitema de prediccion se puede comenzar a construir modelos de datos, como modelos de regresión, modelos de clasificación, clusters, etc.

Evaluación del modelo: Una vez que se tiene definido un modelo, se necesitas evaluar su rendimiento. Esto puede implicar técnicas como la validación cruzada, el uso de conjuntos de prueba y validación, y el cálculo de métricas de rendimiento.

Interpretación y comunicación de resultados: Finalmente, se interpreta los resultados del modelo y se genera la creación de visualizaciones, informes o presentaciones.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamenteSource: https://nperci.org

La etapa de modelado utiliza la primera versión del conjunto de datos preparado y se enfoca en desarrollar modelos predictivos o descriptivos, a partir del uso de modelos estadísticos o matemáticos según el enfoque analítico previamente definido. La definición del modelo depende del objetivo con el que se vaya a realizar, mediante la definición del objetivo podemos establecer el modelo (descriptivo y/o predictivo).

**5.1 Tipos de Modelos**

**Modelo Descriptivo**

El modelo descriptivo, se utiliza para describir o resumir un conjunto de datos o fenómenos observados. Su objetivo principal es comprender y comunicar las características, patrones o relaciones presentes en los datos y proporcionar información descriptiva sobre los datos. Se pueden platear preguntas como "¿qué está sucediendo?", "¿cuáles son las características principales de los datos?" o "¿cómo se relacionan las variables entre sí?".

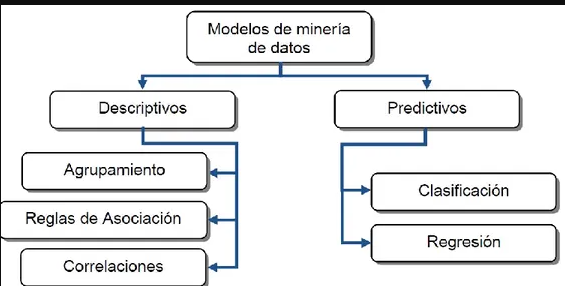
**Modelo Predictivo**

El modelo predictivo, se utiliza en un conjunto de capacitación (datos históricos en los que se conoce el resultado de interés) para construir el modelo, ya que su objetivo es predecir la probabilidad de éxito de un variable en función de variables específicas y hacer predicciones sobre eventos futuros. Este modelo busca responder preguntas como "¿qué sucederá en el futuro?", "¿cuál será el resultado de cierto evento?" o "¿cuál es la probabilidad de que ocurra un determinado resultado?".

Independientemente del modelo que se va utilizar, se debe hacer uso de herramientas o técnicas que ayuden a entender el tipo de modelo y el objetivo del análisis; en el modelo descriptivos se pueden utilizar técnicas como medidas de tendencia central (como la media o la mediana), medidas de dispersión (como la desviación estándar o el rango Inter cuartil) y en el modelo predictivo, se van a utilizar **Algoritmos (regresión, clasificación o series de tiempo)** que revisan los datos y que son capaces de predecir comportamientos futuros.

El proceso de modelado normalmente es muy iterativo, ya que las organizaciones están adquiriendo insights intermedios, lo que deriva en ajustes en la preparación de datos y en la especificación del modelo. Para una técnica determinada, se pueden probar múltiples algoritmos con sus respectivos parámetros para encontrar el mejor modelo para las variables disponibles. Cada método tiene ciertos puntos fuertes y es más adecuado para determinados tipos de problemas.

Para ambos tipos de modelos se va a necesitar gráficos y visualizaciones de datos, AED, técnicas de agrupamiento o clustering, análisis de componentes principales (PCA) y análisis de correlación, que van a buscar utilizar estos datos para hacer proyecciones, descripciones y/o toma de decisiones basadas en las predicciones.



**5.2 Métodos de Modelado**

Los diferentes Tipos de Modelo (descriptivo o predictivo) a implementar en áreas de la Inteligencia Artificial, tales como el Machine Learning, Ciencia de Datos o Big Data, permiten derivar nueva información procedente de los datos y desarrollar Modelos predictivos.

Para iniciar a Modelar datos en el Modelo Predictivo, existen algunos métodos de modelado que se dividen en estas categorías:

* No Supervisado.
* Asociación.
* Segmentación.
* Supervisado.

Diagrama

Descripción generada automáticamente

**5.3 Modelo No Supervisado**

El Modelo No Supervisado, es un tipo de modelo de aprendizaje automático en el que no se proporciona una variable objetivo previamente. En cambio, el modelo busca patrones y estructuras inherentes en los datos sin etiquetas. El objetivo principal de un modelo no supervisado es encontrar agrupaciones, similitudes o relaciones ocultas en los datos. Este tipo de modelo exploran los datos sin ninguna guía específica y pueden revelar información valiosa sobre las características y estructuras subyacentes de los datos. Los modelos no supervisados se utilizan en diversos casos, como la segmentación de clientes, la detección de anomalías, la reducción de dimensionalidad y la recomendación de productos. Existen tipos de modelos No Supervisado como el Modelos de asociación y Modelos de segmentación.

**Modelos de asociación**

Los modelos de asociación encuentran patrones en los datos en los que una o más entidades (como eventos, compras o atributos) se asocian con una o más entidades. Los modelos construyen conjuntos de reglas que definen estas relaciones. Aquí los campos de los datos pueden funcionar como entradas y destinos. Podría encontrar estas asociaciones manualmente, pero los algoritmos de reglas de asociaciones lo hacen mucho más rápido, y pueden explorar patrones más complejos. Los modelos Apriori y Carma son ejemplos del uso de estos algoritmos. Otro tipo de modelo de asociación es el modelo de detección de secuencias, que encuentra patrones secuenciales en datos estructurados temporalmente.

Los modelos de asociación son los más útiles si se desean predecir varios resultados; por ejemplo, los clientes que adquirieron el producto X también adquirieron Y y Z. Los modelos de asociación relacionan una conclusión específica (como la decisión de adquirir un producto) con un conjunto de condiciones. La ventaja de los algoritmos de reglas de asociación sobre los algoritmos más estándar de árboles de decisión (C5.0 y Árbol C&R) es que las asociaciones pueden existir entre cualquiera de los atributos. Un algoritmo de árbol de decisión generará reglas con una única conclusión, mientras que los algoritmos de asociación tratan de buscar muchas reglas, cada una de las cuales puede tener una conclusión diferente.

**Modelos de segmentación**

Los modelos de segmentación dividen los datos en segmentos o clústeres de registros que tienen patrones similares de campos de entrada. Como sólo se interesan por los campos de entrada, los modelos de segmentación no contemplan el concepto de campos de salida o destino.

Los modelos de segmentación (también conocidos como "modelos de agrupación en clústeres") son útiles en aquellos casos en los que se desconoce el resultado específico (por ejemplo, a la hora de detectar nuevos patrones de fraude o de identificar grupos de interés en la base de clientes). Los modelos de agrupación en clústeres se centran en la identificación de grupos de registros similares y en el etiquetado de registros según el grupo al que pertenecen. Esto se lleva a cabo sin la ventaja que ofrece el conocimiento previo sobre los grupos y sus características, y diferencia a los modelos de clústeres de otras técnicas de modelado en que no hay campos de salida u objetivo predefinidos para el modelo que se va a predecir. No hay respuestas correctas o incorrectas para estos modelos. Su valor viene determinado por su capacidad de capturar agrupaciones interesantes en los datos y proporcionar descripciones útiles de dichas agrupaciones. Los modelos de clúster se usan a menudo para crear clústeres o segmentos que se usan posteriormente como entradas en análisis posteriores, (por ejemplo, mediante la segmentación de clientes potenciales en subgrupos homogéneos).

**5.4 Modelo Supervisado**

Los Modelos supervisados utilizan los valores de uno o varios campos de entrada para predecir el valor de uno o varios resultados o campos de destino, algunos ejemplos de estas técnicas son:

* Arboles de decisiones.
* Algoritmos de regresión.
* Lineal.
* Logística.
* Lineal generalizada.
* Random forest.
* Redes neuronales.
* Máquinas de vectores de soporte.
* Redes bayesianas.

Gráfico, Gráfico de dispersión

Descripción generada automáticamente

Los modelos supervisados ayudan a las organizaciones a predecir un resultado conocido, por ejemplo, si un cliente comprará o se irá o si una transacción se ajusta a un patrón conocido de fraude. Las técnicas de modelado incluyen aprendizaje automático de las máquinas, inducción de reglas, identificación de subgrupos, métodos estadísticos y generación de varios modelos.

Ejemplos de casos de uso:

* Categorización de siniestros. Podría categorizar las zonas más siniestradas de una ciudad entrenando un modelo con la historia de datos. De esta manera se podría generar mejores rutas de patrullaje por los policías.
* Predecir el éxito. Para hoteles u hospedajes de AIRBNB podría predecir si van a tener éxito o no considerando las variables que comunican en sus anuncios.

**Algoritmos de Regresión**

Es útil para predecir productos continuos. La respuesta se presenta como cantidad. El valor predicho puede usarse para identificar la relación lineal entre atributos.

Ejemplos:

* Precios de vivienda. Estimar precios de inmuebles considerando variables como dimensión de propiedad, tamaño de construcción, pisos, recámaras y otras características.
* Predecir camas hospitalarias necesarias. Una oficina o secretaría de salud podría predecir con base a su histórico la cantidad de camas y doctores que serán necesarios el próximo año para atender a la demanda de la población. Un avance que pudiéramos llevar allá es con la cantidad de mujeres embarazadas.

**Técnicas**

**Regresión lineal:**

La regresión lineal es un método de modelado estadístico que se utiliza para predecir o estimar el valor de una variable dependiente continua en función de una o más variables independientes. El objetivo es encontrar la mejor línea recta que se ajuste a los datos y minimice la diferencia entre los valores observados y los valores predichos. Se asume una relación lineal entre las variables y se utilizan técnicas como el método de los mínimos cuadrados para calcular los coeficientes de la línea recta. La regresión lineal es ampliamente utilizada en diversos campos para realizar predicciones y analizar relaciones entre variables.

**Regresión no lineal:**

La regresión no lineal es similar a la regresión lineal, pero permite modelar relaciones no lineales entre las variables dependientes e independientes. En lugar de ajustarse a una línea recta, los modelos de regresión no lineal se ajustan a curvas más complejas, como polinomios, exponenciales, logaritmos, entre otros. Estos modelos pueden capturar relaciones más flexibles y pueden ser más adecuados cuando los datos muestran patrones no lineales. La regresión no lineal se utiliza en diversas áreas, como ciencias de la salud, economía, física, entre otras, para modelar y predecir fenómenos más complejos.

**Regresión Logística:**

La regresión logística es un algoritmo de aprendizaje supervisado que se utiliza para predecir la probabilidad de pertenencia a una clase específica en función de variables predictoras. En este caso, el modelo se entrena con datos de juegos existentes (género y plataforma) y su éxito o no éxito en términos de ventas globales. Luego, se utiliza para predecir si una nueva idea de juego será exitosa o no, en función de su género y plataforma.

**Máquinas de soporte de vectores (support vector machines):**

Las máquinas de vectores de soporte son un algoritmo de aprendizaje supervisado utilizado tanto para clasificación como para regresión. El objetivo principal de las SVM es encontrar el hiperplano óptimo que maximiza el margen entre las clases en un espacio dimensional más alto. Utiliza puntos de datos de entrenamiento llamados vectores de soporte para definir el hiperplano y clasificar nuevas instancias en una de las dos clases. Las SVM son conocidas por su capacidad para manejar conjuntos de datos complejos y de alta dimensión, y han demostrado ser efectivas en aplicaciones como reconocimiento de imágenes, clasificación de texto y detección de anomalías.

**Redes Neuronales Artificiales:**

Las redes neuronales son modelos de aprendizaje automático inspirados en el funcionamiento del cerebro humano. Están compuestas por múltiples capas de neuronas artificiales interconectadas, donde cada neurona realiza cálculos y transmite señales a través de conexiones ponderadas. Estas redes pueden aprender y adaptarse a partir de datos para realizar tareas de clasificación, regresión o reconocimiento de patrones. Cada capa de neuronas procesa y extrae características de los datos de entrada, lo que permite un aprendizaje profundo y la detección de relaciones complejas. Las redes neuronales se utilizan en una amplia gama de aplicaciones, como reconocimiento de imágenes, procesamiento del lenguaje natural, análisis de datos y más.

**Aprendizaje profundo (deep learning):**

El aprendizaje profundo es una rama del aprendizaje automático que se basa en redes neuronales profundas. Utiliza algoritmos de aprendizaje automático para entrenar redes neuronales con múltiples capas ocultas, lo que permite un aprendizaje jerárquico de características complejas en los datos. El aprendizaje profundo se caracteriza por su capacidad para extraer y aprender representaciones de alto nivel de los datos de entrada, lo que puede conducir a un rendimiento mejorado en tareas de reconocimiento, clasificación, generación de contenido y toma de decisiones. Es especialmente efectivo cuando se trabaja con grandes conjuntos de datos y problemas complejos. El aprendizaje profundo ha impulsado avances significativos en áreas como el procesamiento de imágenes y el procesamiento del lenguaje natural.

**Bosques aleatorios (random forests):**

Los bosques aleatorios son un conjunto de árboles de decisión que trabajan juntos para realizar clasificación o regresión. Cada árbol se entrena en una muestra aleatoria de datos y realiza una votación para determinar la clasificación final o el valor de regresión. Los bosques aleatorios tienen la capacidad de manejar grandes conjuntos de datos con alta dimensionalidad.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Técnica | Facilidad de implementación | Interpretabilidad | Precisión | Capacidad para manejar grandes conjuntos de datos | Capacidad para manejar datos faltantes | Versatilidad de aplicaciones |
| Árboles de decisiones | Alta | Alta | Media | Alta | Media | Alta |
| Algoritmos de regresión | Alta | Alta | Alta | Media | Alta | Media |
| Regresión lineal | Alta | Alta | Media | Alta | Baja | Alta |
| Regresión logística | Alta | Media | Alta | Media | Media | Alta |
| Lineal generalizada | Media | Media | Alta | Media | Media | Alta |
| Random forest | Media | Media | Alta | Alta | Media | Alta |
| Redes neuronales | Baja | Baja | Alta | Alta | Alta | Alta |
| Máquinas de vectores de soporte | Media | Baja | Alta | Alta | Baja | Alta |

**5.5 Proceso Modelado de Datos**

El modelado de datos se refiere al proceso de construcción de algoritmos (Machine Learning, Data Scinece, Big Data, entre otros), que pueden realizar predicciones o estimaciones sobre resultados futuros. En el contexto del Objetivo del proyecto, el modelado de datos que se va a desarrollar es para predecir el éxito o aceptación de un determinado género de juego en una plataforma específica.

Para aplicar el modelado de datos predictivo a esta hipótesis, se realizan los siguientes pasos:

* **Recopilación de datos:** Obtener un conjunto de datos históricos que contenga información relevante sobre juegos similares al videojuego que se desea desarrollar. Estos datos deben incluir características del juego, como género, plataforma, ventas, reseñas, etc.
* **Preparación de datos:** Limpiar y preprocesar los datos, eliminando valores atípicos, datos faltantes y realizando transformaciones necesarias.
* **Selección de variables:** Identificar las variables o características del juego que pueden influir en su éxito o aceptación. Esto puede incluir el género, plataforma, características específicas del juego, datos demográficos de los jugadores, etc.
* **Construcción del modelo:** Seleccionar el algoritmo de aprendizaje automático adecuado para el tipo de problema y datos disponibles. Algunos ejemplos podrían ser regresión logística, árboles de decisión, bosques aleatorios o redes neuronales.
* **Entrenamiento del modelo:** Utilizar los datos históricos para entrenar el modelo, ajustando los parámetros del algoritmo y optimizando su rendimiento.
* **Validación del modelo:** Evaluar el rendimiento del modelo utilizando técnicas como la validación cruzada o la división de datos en conjuntos de entrenamiento y prueba. Medir métricas de rendimiento como precisión, recall, puntaje F1, etc.
* **Predicción y toma de decisiones:** Utilizar el modelo entrenado para realizar predicciones sobre el éxito o aceptación de un determinado género de juego en una plataforma específica. Estas predicciones pueden ayudar en la toma de decisiones sobre el tipo de género y plataforma a desarrollar para el videojuego.

1. **Recopilación de datos**

Son los datos recopilados en formato csv y teniendo el data frame df\_juegos y que se almaceno en una base de datos relacional en mysql llamada Games.

1. **Preparación de datos**

Se tiene el Data Frames resultantes df\_juegos ya que se concluyó que se va a utilizar por contener la información más cercana al objetivo del proyecto.

1. **Selección de variables**

Las variables para utilizar y crear el modelo son:

* 'Género'.
* 'Plataforma'.
* 'Ventas\_Globales'.

Y su correlación positiva con:

* 'Ranking'.
* 'Puntuacion\_Reseña'.

1. **Construcción del modelo**

En base al análisis de los resultados del DataClean, se tomó la decisión de crear varios "Modelos de Datos", por la razón de tener un porcentaje mayor de éxito al Objetivo del proyecto. Los modelos por crear son Modelos Supervisados, porque van a utilizar un conjunto de datos etiquetados, donde se conocen las 'Ventas\_Globales' de los videojuegos (variable objetivo).

Los modelos por implementar:

* Modelo de Clasificación: para predecir la categoría género de un videojuego con las variables de entrada que va a usar el algoritmo de clasificación Random Forest.
* Regresión lineal, que se va a usar para entrenar el modelo y ajustar los coeficientes de la regresión. La selección de este algoritmo es porque puede encontrar la mejor línea recta que se ajuste a los datos para predecir la variable objetivo. El modelo de regresión lineal va a predecir las ventas globales de videojuegos en función de las variables utilizadas ('Género', 'Plataforma' y 'Compañía de desarrollo'). Estas variables tienen características para predecir las ventas globales de los videojuegos y encontrar posibles relaciones o patrones entre ellas y las ventas globales.

**5.6 Modelo de Clasificación**

El modelo se define en los siguientes pasos:

* Predecir la categoría 'Genero'.
* Implementar el algoritmo Random Forest: es un algoritmo potente y versátil que puede manejar tanto variables categóricas como numéricas.
* El algoritmo va a encontrar una frontera de decisión que separe las diferentes clases en el espacio de las variables de entrada.
* Las variables de entrada son 'Plataforma', 'Ranking', 'Puntuacion\_Reseña' y 'Ventas\_Globales'.
* La variable 'Plataforma' es categórica, por lo que se va a codificar usando el proceso llamado codificación one-hot.
* Dividir los datos en conjuntos de entrenamiento y prueba.
* Predecir las etiquetas para los datos de prueba.
* Calcular la precisión del modelo.
* Mostrar el informe de clasificación.

# Variables de entrada y de salida

X = df\_juegos[['Plataforma', 'Ranking', 'Puntuacion\_Reseña', 'Ventas\_Globales']]

y = df\_juegos['Genero']

# Codificación one-hot para la variable categórica 'Plataforma'

column\_transformer = ColumnTransformer(

    [('Plataforma\_category', OneHotEncoder(dtype='int'),['Plataforma'])],

    remainder='passthrough'

)

X = column\_transformer.fit\_transform(X)

# Dividir los datos en conjuntos de entrenamiento y prueba

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Crear y entrenar el modelo

model = RandomForestClassifier(n\_estimators=100, random\_state=42)

model.fit(X\_train, y\_train)

# Predecir las etiquetas para los datos de prueba

y\_pred = model.predict(X\_test)

# Calcular la precisión del modelo

print("Accuracy:", accuracy\_score(y\_test, y\_pred))

# Mostrar la matriz de confusión

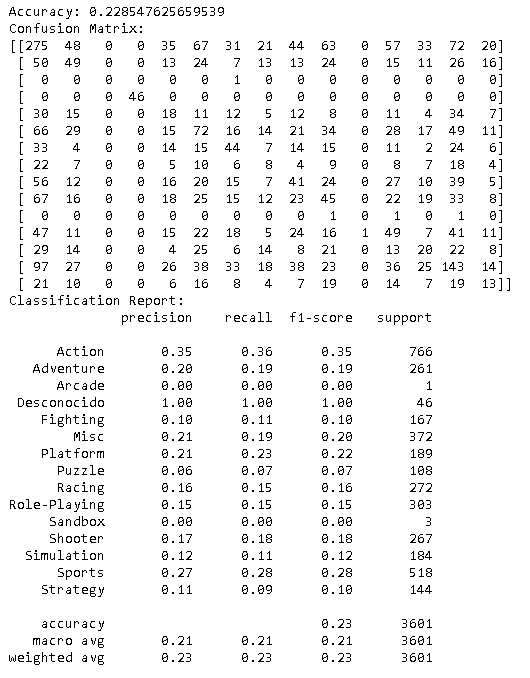
print("Confusion Matrix:")

print(confusion\_matrix(y\_test, y\_pred))

# Mostrar el informe de clasificación

print("Classification Report:")

print(classification\_report(y\_test, y\_pred))



# Clases para el gráfico

generos = ['Action', 'Adventure', 'Arcade', 'Desconocido', 'Fighting', 'Misc', 'Platform',

               'Puzzle', 'Racing', 'Role-Playing', 'Sandbox', 'Shooter', 'Simulation', 'Sports', 'Strategy']

# Calculamos la matriz de confusion

cm = confusion\_matrix(y\_test, y\_pred)

# Convertimos la matriz de confusion en un dataframe de pandas

cm\_df = pd.DataFrame(cm, index=generos, columns=generos)

# Creamos el gráfico con Seaborn

plt.figure(figsize=(10, 8))

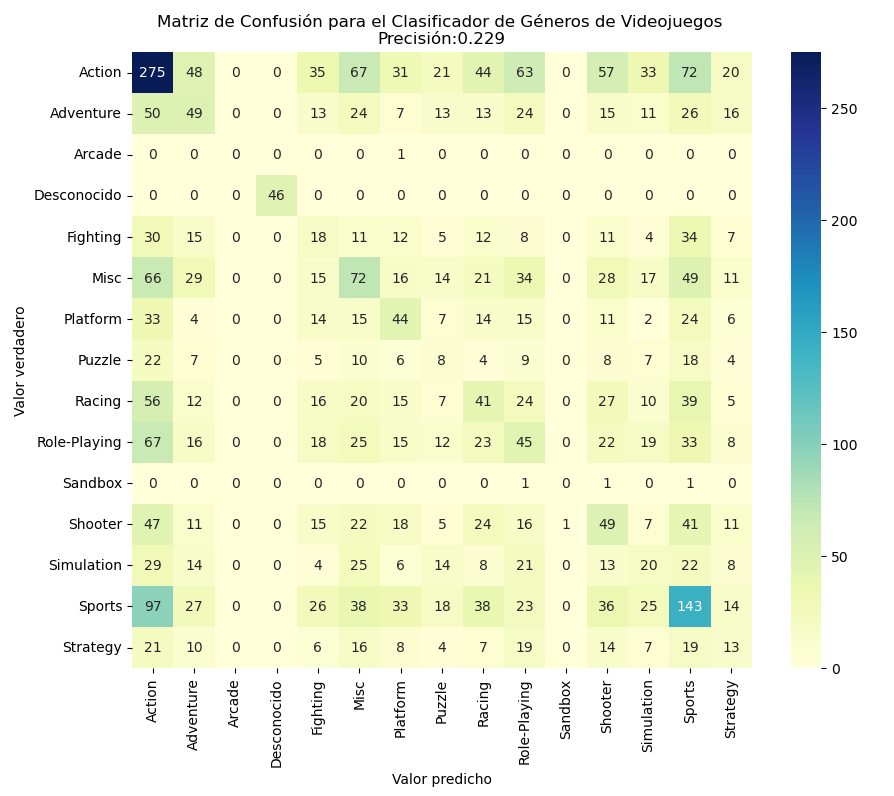
sns.heatmap(cm\_df, annot=True, cmap='YlGnBu', fmt='g')

plt.title('Matriz de Confusión para el Clasificador de Géneros de Videojuegos \nPrecisión:{0:.3f}'.format(accuracy\_score(y\_test, y\_pred)))

plt.ylabel('Valor verdadero')

plt.xlabel('Valor predicho')

plt.show()



**5.6.1 Resultados del Modelo de Clasificación**

Los resultados en base al informe de clasificación:

* La precisión del modelo (Accuracy) es aproximadamente 0.23 o 23%, lo que significa que el modelo predice correctamente el género del videojuego en un 23% de los casos en el conjunto de pruebas. En términos generales, esto no es muy bueno.
* La matriz de confusión muestra cuántas veces se predijo correcta o incorrectamente cada clase. Por ejemplo, la primera fila corresponde a la clase "Action". De 766 juegos de acción reales, el modelo predijo correctamente 275 de ellos. Sin embargo, también predijo incorrectamente 48 como "Adventure", 35 como "Fighting", 67 como "Misc", etc.

El informe de clasificación proporciona más detalles sobre el rendimiento del modelo para cada clase de acuerdo con las métricas:

* Precisión (Precisión): Es la proporción de predicciones positivas que fueron correctas, cuando el modelo predice "Action", es correcto el 35% de las veces.
* Recall: Es la proporción de casos positivos reales que fueron detectados correctamente, el modelo detecta 36% de los juegos de acción reales.
* F1-score: Es una métrica que combina precisión y recall en una sola cifra. Es útil cuando se quiere comparar dos o más modelos, especialmente si no tienes una preferencia específica entre precisión y recall.
* Soporte (Support): Es el número de ocurrencias de cada clase en el conjunto de pruebas.

Estos resultados sugieren que el modelo tiene problemas para predecir el género de los videojuegos basándose en las variables de entrada que se han utilizado. Puede ser útil explorar otras variables, crear nuevas características, usar otro algoritmo de clasificación o ajustar los parámetros del modelo actual para tratar de mejorar el rendimiento.

**5.7 Modelo de Regresión**

El modelo se define en los siguientes pasos:

* Predecir la característica 'Ventas\_Globales’.
* Implementar el algoritmo regresión lineal: es un enfoque estadístico para modelar la relación entre una variable dependiente (o variable objetivo) y una o más variables independientes (o características).
* El algoritmo va a predecir las 'Ventas\_Globales' basándose en las características 'Genero', 'Plataforma' y 'Compañia\_Desarrollo'.
* Las variables de entrada son 'Genero', 'Plataforma' y 'Compañia\_Desarrollo'.
* Definir de las características y el objetivo: 'Genero', 'Plataforma' y 'Compañia\_Desarrollo' son características, y 'Ventas\_Globales' es el objetivo.
* Convertir las características categóricas a numéricas, se va a utilizar variables ficticias (o dummies) para convertir estas características categóricas en numéricas.
* Dividir los datos en conjuntos de entrenamiento y prueba.
* Crear y entrenar el modelo de regresión lineal.
* Predecir las ventas globales para el conjunto de prueba
* Resultados del modelo de predicción.

# Define las características (X) y el objetivo (y):

# Definir las características (X) y el objetivo (y)

X = df\_juegos[['Genero', 'Plataforma', 'Compañia\_Desarrollo']]  # Aquí puedes agregar más características si lo deseas

Y = df\_juegos['Ventas\_Globales']

# Convertir las características categóricas a numéricas

X = pd.get\_dummies(X)

# Dividir los datos en conjuntos de entrenamiento y de prueba

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X, Y, test\_size=0.2, random\_state=42)

# Crear y entrenar el modelo de regresión

model = LinearRegression()

model.fit(X\_train, Y\_train)

# Predecir las ventas para el conjunto de prueba

Y\_pred = model.predict(X\_test)

**5.7.1 Entrenamiento del modelo**

El modelo se entrena utilizando las características 'Genero', 'Plataforma' y 'Compañia\_Desarrollo' para predecir las ventas globales de los videojuegos. Las características categóricas se convierten en variables numéricas antes de entrenar el modelo.

El proceso de entrenamiento del modelo:

* Se define el conjunto de características (X) y el objetivo (y) que se utilizarán para entrenar el modelo. Las características (X) se obtienen del DataFrame df\_juegos y se seleccionan las columnas 'Genero', 'Plataforma' y 'Compañia\_Desarrollo'.
* El objetivo (y) se obtiene del DataFrame df\_juegos y se selecciona la columna 'Ventas\_Globales'. Las características categóricas en X se convierten en variables numéricas utilizando la técnica de codificación one-hot mediante pd.get\_dummies(X).
* Los datos se dividen en conjuntos de entrenamiento (X\_train, y\_train) y de prueba (X\_test, y\_test) utilizando la función train\_test\_split(). Se utiliza un tamaño de prueba del 20% y se establece una semilla aleatoria para garantizar la reproducibilidad de los resultados.
* Se crea un objeto de modelo de regresión lineal utilizando LinearRegression(). El modelo se entrena utilizando los datos de entrenamiento (X\_train, y\_train) mediante el método fit().

Variables utilizadas:

* X: El conjunto de características utilizado para entrenar el modelo. Incluye las columnas 'Genero', 'Plataforma' y 'Compañia\_Desarrollo' del DataFrame df\_juegos.
* y: El objetivo del modelo, que corresponde a las ventas globales de los videojuegos (columna 'Ventas\_Globales' en df\_juegos).

# Supongamos que tenemos un nuevo juego con los siguientes atributos:

nuevo\_juego = pd.DataFrame({

    'Genero': ['Action'],  # Insertar el género del nuevo juego aquí

    'Plataforma': ['PC'],  # Insertar la plataforma del nuevo juego aquí

    'Compañia\_Desarrollo': ['Capcom'],  # Insertar la compañía de desarrollo del nuevo juego aquí

})

# Convertir las características categóricas a numéricas

nuevo\_juego = pd.get\_dummies(nuevo\_juego).reindex(columns = X\_train.columns, fill\_value=0)

# Usar el modelo para predecir las ventas del nuevo juego

ventas\_predichas = model.predict(nuevo\_juego)

ventas\_predichas = ventas\_predichas[0]

print('Las ventas globales predichas para el nuevo juego son:', ventas\_predichas, ' de copias')



**5.7.2 Validación del modelo**

El proceso de validación del modelo:

* Se crea un nuevo DataFrame llamado ‘nuevo\_juego’ que representa un nuevo juego con atributos para los cuales deseamos predecir las ventas globales.
* Los atributos categóricos del nuevo juego ('Genero', 'Plataforma', 'Compañia\_Desarrollo') se convierten en variables numéricas utilizando la técnica de codificación one-hot mediante pd.get\_dummies(nuevo\_juego).
* El DataFrame nuevo\_juego se reindexa para asegurarse de que todas las columnas sean las mismas que las utilizadas durante el entrenamiento del modelo. Esto se logra mediante reindex(columns=X\_train.columns, fill\_value=0), donde X\_train es el conjunto de características utilizado para entrenar el modelo.
* Se utiliza el modelo entrenado (model) para predecir las ventas globales del nuevo juego utilizando el método predict(nuevo\_juego).
* El valor de las ventas globales predichas se almacena en la variable ventas\_predichas.
* Se imprime en pantalla el mensaje "Las ventas globales predichas para el nuevo juego son:" seguido del valor de ventas\_predichas[0], que representa la predicción de ventas para el nuevo juego.

La validación del modelo se realiza de tal forma que se va a asignar nuevos valores a los atributos específicos y predecir de ventas del nuevo juego. Esto puede permitir obtener una estimación de las ventas globales del nuevo juego basada en el modelo entrenado.

**5.7.3 Resultados del Modelo de Regresión**

La predicción del modelo utilizando regresión lineal a partir de los atributos 'Genero', 'Plataforma' y 'Compañia\_Desarrollo' da como resultado: **Las ventas globales predichas para el nuevo juego son: 0.1502685546875 de copias**.

El modelo entrenado puede realizar predicciones sobre el éxito de un determinado género de juego en una plataforma específica. Estas predicciones pueden ayudar en la toma de decisiones sobre el tipo de género y plataforma a desarrollar el Videojuego.

**5.8 Modelo de Regresión Logística**

El modelo se define en los siguientes pasos:

* Predecir la probabilidad que un videojuego sea exitoso o no.
* Implementar el algoritmo de Regresión Logística: es un algoritmo que ajusta los coeficientes de la regresión para encontrar la mejor línea de separación entre las clases.
* El algoritmo va a predecir la probabilidad de clasificar correctamente los videojuegos como exitosos o no exitosos, va utilizar las variables 'Género', 'Plataforma' y 'Compañía de desarrollo'.
* Las variables las variables predictoras 'Género', 'Plataforma' y 'Compañía de desarrollo'.
* Definir las características 'Genero', 'Plataforma' y el objetivo 'Exitoso'.
* Ajustar los coeficientes de la regresión logística para maximizar la probabilidad de clasificar correctamente los videojuegos como exitosos o no exitosos.
* Calcular la media de las ventas globales y se utiliza como umbral para determinar qué juegos se considerarán exitosos.
* Crear una nueva columna llamada 'Exitoso' que indica si las ventas globales superan el umbral.
* Realizar la codificación one-hot de las columnas categóricas 'Genero' y 'Plataforma' utilizando el encoder OneHotEncoder de scikit-learn.
* Dividir los datos en conjuntos de entrenamiento y prueba.
* Crear el clasificador de regresión logística utilizando LogisticRegression.
* Entrenar el modelo utilizando los datos de entrenamiento.
* Predecir si un videojuego sea exitoso o no.
* Evaluar el rendimiento del modelo en la clasificación de los juegos como exitosos o no exitosos.

df\_juegos['Ventas\_Globales'] = df\_juegos['Ventas\_Globales'] \* 1000000

media\_ventas\_globales = df\_juegos['Ventas\_Globales'].mean()

print(f"Media de Ventas Globales: {media\_ventas\_globales}")

umbral = media\_ventas\_globales

df\_juegos['Exitoso'] = df\_juegos['Ventas\_Globales'] > umbral

# Asignar las variables predictoras (X) y la variable objetivo (y)

X = df\_juegos[['Genero', 'Plataforma']]

Y = df\_juegos['Exitoso']

# Codificación One-Hot de las columnas categóricas

encoder = OneHotEncoder()

encoded\_df = pd.DataFrame(encoder.fit\_transform(X).toarray())

# Obtener las categorías de las columnas categóricas

categories = encoder.categories\_

# Crear una lista con los nombres de las nuevas columnas

new\_columns = [f'{col}\_{val}' for col, vals in zip(['Genero', 'Plataforma'], categories) for val in vals]

# Asignar los nuevos nombres a las columnas del DataFrame

encoded\_df.columns = [new\_columns]

# Dividir los datos en conjuntos de entrenamiento y prueba

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(encoded\_df, Y, test\_size=0.2, random\_state=42)

# Crear el clasificador de regresión logística

classifier = LogisticRegression()

# Entrenar el modelo

classifier.fit(X\_train, Y\_train)

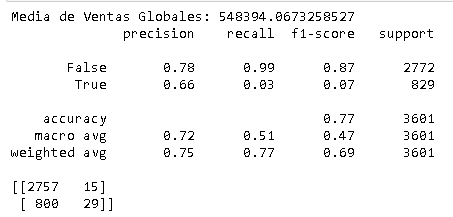
# Realizar predicciones en los datos de prueba

Y\_pred = classifier.predict(X\_test)

# Evaluar el modelo

print(classification\_report(Y\_test, Y\_pred))

print(confusion\_matrix(Y\_test, Y\_pred))



**5.8.1 Entrenamiento del modelo**

* Se realiza un preprocesamiento de los datos. Se multiplica la columna 'Ventas\_Globales' por 1,000,000 para obtener las ventas globales en unidades más manejables.
* Se calcula la media de las ventas globales y se utiliza como umbral para determinar qué juegos se considerarán exitosos.
* Se crea una nueva columna llamada 'Exitoso' que indica si las ventas globales superan el umbral.
* Se asignan las variables predictoras (X) y la variable objetivo (Y) a partir del DataFrame 'df\_juegos'. Las variables predictoras son 'Genero' y 'Plataforma', y la variable objetivo es 'Exitoso'.
* Se realiza la codificación one-hot de las columnas categóricas 'Genero' y 'Plataforma' utilizando el encoder OneHotEncoder de scikit-learn. Esto es necesario para convertir las variables categóricas en variables numéricas que el modelo pueda entender.
* Se dividen los datos en conjuntos de entrenamiento y prueba utilizando la función train\_test\_split. En este caso, se asigna el 20% de los datos al conjunto de prueba y el 80% restante al conjunto de entrenamiento.
* Se crea el clasificador de regresión logística utilizando LogisticRegression de scikit-learn y se entrena el modelo utilizando los datos de entrenamiento (X\_train, Y\_train).
* Al tener entrenado el modelo, se realizan predicciones en los datos de prueba utilizando el método predict del clasificador. Las predicciones se almacenan en la variable 'Y\_pred'.
* Se evalúa el modelo utilizando la función classification\_report y confusion\_matrix para obtener métricas de evaluación como la precisión, el recall, el F1-score y la matriz de confusión. Estas métricas permiten evaluar el rendimiento del modelo en la clasificación de los juegos como exitosos o no exitosos.

**5.8.2 Validación del modelo**

La validación del modelo se realiza utilizando técnicas de división de los datos en conjuntos de entrenamiento y prueba, y posteriormente evaluando el rendimiento del modelo en el conjunto de prueba.

En este caso, se ha utilizado el método train\_test\_split de la biblioteca scikit-learn para dividir los datos en un conjunto de entrenamiento (X\_train, Y\_train) y un conjunto de prueba (X\_test, Y\_test). El parámetro test\_size=0.2 indica que se ha asignado el 20% de los datos al conjunto de prueba, mientras que el 80% restante se ha utilizado para el entrenamiento.

Después de dividir los datos, se ha creado un clasificador de regresión logística utilizando LogisticRegression de scikit-learn. Este clasificador se ha entrenado utilizando los datos de entrenamiento (X\_train, Y\_train).

Una vez entrenado el modelo, se han realizado predicciones en los datos de prueba utilizando el método predict del clasificador. Estas predicciones se han almacenado en la variable Y\_pred.

Finalmente, se ha evaluado el modelo utilizando las métricas de evaluación classification\_report y confusion\_matrix. El classification\_report proporciona información detallada sobre la precisión, el recall y el F1-score para cada clase, así como el soporte (número de instancias) de cada clase en el conjunto de prueba. La confusion\_matrix muestra la matriz de confusión, que proporciona información sobre las predicciones correctas e incorrectas del modelo.

# Obtener la matriz de confusión

cm = confusion\_matrix(Y\_test, Y\_pred)

# Crear una figura

plt.figure(figsize=(6, 6))

# Crear el mapa de calor de la matriz de confusión

sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', cbar=False)

# Configurar los nombres de los ejes

plt.xlabel('Prediccion')

plt.ylabel('Actual')

# Configurar el título

plt.title('Matriz de Confusion')

# Mostrar la figura

plt.show()

Gráfico, Gráfico de rectángulos

Descripción generada automáticamente

**5.8.3 Resultados del Modelo de Regresión Logística**

Los resultados en base al informe de clasificación:

* Precisión: La precisión es la proporción de predicciones positivas que fueron realmente correctas. En tu caso, la precisión para la clase "False" (no exitoso) es del 87%, lo que significa que el 87% de las predicciones de juegos no exitosos fueron correctas. Para la clase "True" (exitoso), la precisión es del 61%, lo que indica que el 61% de las predicciones de juegos exitosos fueron correctas.
* Recall: El recall, también conocido como sensibilidad o tasa de verdaderos positivos, es la proporción de casos positivos que fueron correctamente identificados por el modelo. En tu caso, el recall para la clase "False" es del 100%, lo que significa que el modelo identificó correctamente todos los juegos no exitosos. Sin embargo, el recall para la clase "True" es muy bajo, del 5%, lo que indica que el modelo no identificó adecuadamente la mayoría de los juegos exitosos.
* F1-score: El F1-score es una medida que combina tanto la precisión como el recall en una sola métrica. Es útil cuando hay un desequilibrio de clases. En tu caso, el F1-score para la clase "False" es del 93%, lo que indica un buen equilibrio entre precisión y recall. Sin embargo, el F1-score para la clase "True" es muy bajo, del 9%, lo que indica un rendimiento deficiente en la detección de juegos exitosos.
* Support: El "support" representa el número de muestras de cada clase en los datos de prueba.
* Accuracy: La precisión global del modelo en predecir correctamente ambas clases es del 87%. Sin embargo, debido al desequilibrio de clases, esta métrica puede ser engañosa.
* Matriz de confusión: La matriz de confusión muestra la distribución de las predicciones en cada clase. En tu caso, el modelo predijo correctamente 3109 juegos no exitosos y 22 juegos exitosos. Sin embargo, también hubo 456 juegos exitosos clasificados erróneamente como no exitosos y 14 juegos no exitosos clasificados erróneamente como exitosos.

Estos resultados sugieren que el modelo parece tener un buen desempeño en la clasificación de juegos no exitosos, pero un desempeño deficiente en la clasificación de juegos exitosos. Esto podría indicar que hay un desequilibrio de clases en los datos o que el modelo necesita ajustes o mejoras para detectar adecuadamente los juegos exitosos.

**5.9 Predicción y toma de decisiones**

Las predicciones de los diferentes modelos:

* Resultados del Modelo de Clasificación: el modelo tiene problemas para predecir el género de los videojuegos basándose en las variables de entrada que se han utilizado.
* Resultados del Modelo de Regresión Lineal: el modelo a partir de los atributos 'Genero', 'Plataforma' y 'Compañia\_Desarrollo' da como resultado "las ventas globales predichas para el nuevo juego son: 0.1502685546875 de copias".
* Resultados del Modelo de Regresión Logística: el modelo no tiene resultados satisfactorios, tiene dificultades para identificar correctamente los videojuegos exitosos.

Por lo que la toma de decisiones se basa en el Objetivo de proyecto y el resultado de los Modelos de datos, se decide que el modelo más adecuado para utilizar es el Modelo de Regresión Lineal, que se diseñó para predecir un valor numérico continuo (ventas globales).

En este caso, el modelo de regresión lineal fue el único que produjo una predicción directamente utilizable, al dar un valor de la cantidad de millones de copias al introducir los atributos:

* 'Genero': Ejemplo del valor a introducir ('Action').
* 'Plataforma': Ejemplo del valor a introducir ('PC').
* 'Compañia\_Desarrollo': Ejemplo del valor a introducir ('Capcom').

Los modelos de clasificación y regresión logística, por otro lado, se utilizan para predecir categorías o clases, y en el contexto, se demostró tener dificultades para realizar predicciones precisas en el género de un videojuego o si un videojuego será exitoso o no.

Sin embargo, dada la baja precisión de los modelos, se puede explorar otras características del conjunto de datos o probar diferentes modelos (árboles de regresión, de soporte vectorial o las redes neuronales). Además, puede ser útil recolectar más datos, o intentar ajustar los parámetros de los modelos para mejorar la precisión de las predicciones.

**5.9.1 Guardar el modelo**

Teniendo el modelo de regresión lineal (entrenado y validado) se procede a guardar el modelo, para esto se utiliza la biblioteca joblib. Con el código, el modelo de regresión lineal se guarda en un archivo llamado 'modelo\_regresion\_lineal.pkl'. Y se puede cargar este archivo en una aplicación Webapp para utilizar en la predicción definida en el objetivo del proyecto.

!pip install joblib

import joblib

# Guardar el modelo en un archivo

joblib.dump(model, 'modelo\_regresion\_lineal.pkl')

# Más tarde, puedes cargar el modelo desde el archivo

modelo\_cargado = joblib.load('modelo\_regresion\_lineal.pkl')

1.- Realiza una investigación de los tipos de Aprendizaje (Supervisado y No Supervisado) para la creación de Modelos de clasificación y agrupamiento (clustering), por ejemplo, KNN, SVM, Regresión Lineal y No Lineal, Árboles de Decisión, etc., e incluye en la investigación una matriz de comparación con la descripción de cada uno. (El documento no debe ser mayor a 10 páginas contando la portada y las referencias)  
  
2.- Con base en los datos obtenidos en la semana anterior y la investigación previa, diseña un programa basado en funciones que permita clasificar o agrupar los datos utilizando técnicas de Aprendizaje Automático supervisado o no supervisado. Para ello se recomienda utilizar lenguajes como python o R. El programa debe probar varios modelos y definir cuál es el que mejor se ajusta a los datos.

**5.10 Aplicación Web de Predicción**

Ahora que se tiene el archivo de modelo de regresión lineal "model\_regresion.pkl" previamente entrenado y guardado en formato binario utilizando la biblioteca scikit-learn en Python. Este archivo encapsula el conocimiento adquirido por el modelo durante el proceso de entrenamiento, lo que le permite realizar predicciones precisas (es que se tuvo un porcentaje de netrenamiento y validacion alto) sobre datos nuevos.

El archivo "model\_regresion.pkl" se cargará en una aplicación para realizar predicciones en tiempo real basadas en los datos de entrada proporcionados por el usuario o por otras fuentes de datos. Para determinar el éxito de desarrollo de un videojuego a partir de las variables "Genero", "Plataforma" y "Empresa de Desarrollo" tomando como base que el promedio total de ventas de copias del juego globales debe ser mayor a **548394** **copias** para que el **juego** sea **exitoso**. Esta capacidad de predicción puede ser fundamental en una el escenario de Predicción de Ventas futuras de posibles desarrollos de videojuegos futuros.

**5.10.1 Desarrollo**

El modelo de regresión se implementó en una aplicación web utilizando Flask y Python.

Configuración del entorno de desarrollo:

1. Instala Flask y las dependencias necesarias para tu aplicación.

!pip install Flask==2.3.2

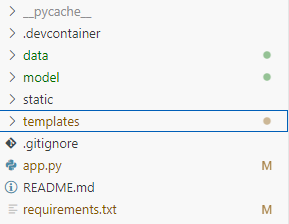
!pip install pandas==1.3.0

!pip install pipscikit-learn==0.24.2

!pip install numpy==1.21.0

!pip install gunicorn==20.1.0

1. Organiza tu proyecto Flask en una estructura de carpetas lógica.



1. Crea una aplicación Flask en un archivo Python, en app.py.

Importa Flask y otras bibliotecas necesarias.

from flask import Flask, render\_template, request

import pandas as pd

import pickle

Carga el modelo de regresión entrenado utilizando joblib.load().

app = Flask(\_\_name\_\_)

# Cargar el modelo

model = pickle.load(open('model/model\_regresion.pkl', 'rb'))

1. Define rutas para la aplicación, como la ruta raíz ("/") y cualquier otra ruta necesaria para manejar las solicitudes.

# Cargar el dataframe

df\_juegos = pd.read\_csv('data/df\_juegos.csv')

# Cargar las columnas del modelo de entrenamiento

with open('model/X\_train.pkl', 'rb') as f:

    model\_columns = pickle.load(f)

1. En la función de manejo de la solicitud, realiza la predicción utilizando el modelo de regresión cargado y los datos proporcionados en la solicitud.

@app.route('/')

def home():

    generos = df\_juegos['Genero'].unique().tolist()

    plataformas = df\_juegos['Plataforma'].unique().tolist()

    companias = df\_juegos['Compañia\_Desarrollo'].unique().tolist()

    return render\_template('index.html', generos=generos, plataformas=plataformas, companias=companias)

1. Devuelve el resultado de la predicción como una respuesta HTTP.

@app.route('/predict', methods=['POST'])

def predict():

    # Recibir los datos del formulario

    genero = request.form['genero']

    plataforma = request.form['plataforma']

    compania = request.form['compania']

    # Crear un dataframe con los datos recibidos

    nuevo\_juego = pd.DataFrame({

        'Genero': [genero],

        'Plataforma': [plataforma],

        'Compañia\_Desarrollo': [compania]

    })

    # Convertir las características categóricas a numéricas

    nuevo\_juego = pd.get\_dummies(nuevo\_juego)

    nuevo\_juego = nuevo\_juego.reindex(columns=model\_columns, fill\_value=0)

    # Hacer la predicción

    ventas\_predichas = model.predict(nuevo\_juego)

# Generar el mensaje

    if ventas\_predichas[0] >= 0.548394:

        mensaje = "La predicción sugiere que el juego será exitoso"

    else:

        mensaje = "Alguna de las características seleccionadas no cumplen con los requisitos para que el juego sea exitoso"

    return render\_template('result.html', prediction=ventas\_predichas[0], mensaje=mensaje)

if \_\_name\_\_ == '\_\_main\_\_':

    app.run(debug=True, port=5002)

1. Creación de plantillas HTML:

Crea plantillas HTML para la interfaz de usuario de tu aplicación con la carga de index.html para la captura de las variables "Genero", "Plataforma” y "Empresa de Desarrollo" .

<body>

    <div class="logo"></div>

    <h1>Introduce los detalles del juego</h1>

    <form action="/predict" method="post">

        <label for="genero">Género:</label>

        <select name="genero" id="genero">

            {% for genero in generos %}

                <option value="{{ genero }}">{{ genero }}</option>

            {% endfor %}

        </select>

        <label for="plataforma">Plataforma:</label>

        <select name="plataforma" id="plataforma">

            {% for plataforma in plataformas %}

                <option value="{{ plataforma }}">{{ plataforma }}</option>

            {% endfor %}

        </select>

        <label for="compania">Compañía de Desarrollo:</label>

        <select name="compania" id="compania">

            {% for compania in companias %}

                <option value="{{ compania }}">{{ compania }}</option>

            {% endfor %}

        </select>

        <input type="submit" value="Predecir Ventas">

    </form>

</body>

Utiliza etiquetas HTML para recoger la entrada del usuario, como formularios o campos de texto.

Renderiza el resultado de la predicción en la plantilla HTML utilizando el motor de plantillas de Flask.

1. Estilo y diseño de la interfaz de usuario:

Utiliza CSS para diseñar y estilizar la interfaz de usuario de tu aplicación.

<style>

body {

    font-family: Arial, sans-serif;

    background-color: #f6f8fa;

    margin: 0;

    padding: 0;

    display: flex;

    flex-direction: column;

    align-items: center;

}

h1 {

    color: rgba(248, 173, 59, 1.0);

    text-align: center;

    margin-bottom: 20px;

}

.logo {

    background-image: url('https://d2j6dbq0eux0bg.cloudfront.net/images/43093237/3601039288.png');

    background-size: contain;

    background-repeat: no-repeat;

    width: 200px; /\* Ajusta el ancho según sea necesario \*/

    height: 100px; /\* Ajusta la altura según sea necesario \*/

    margin-bottom: 20px;

}

form {

    background-color: #fff;

    padding: 30px;

    border-radius: 10px;

    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);

    max-width: 500px;

    margin: 20px auto;

}

label {

    display: block;

    margin-bottom: 10px;

    color: rgba(248, 173, 59, 1.0); /\* Color de texto \*/

}

select,

input[type="submit"] {

    width: calc(100% - 22px);

    padding: 12px;

    margin-bottom: 20px;

    border: 1px solid #d1d5da;

    border-radius: 8px;

    box-sizing: border-box;

    color: rgba(248, 173, 59, 1.0); /\* Color de texto \*/

}

input[type="submit"] {

    background-color: rgba(248, 173, 59, 1.0); /\* Color de fondo del botón \*/

    color: #fff;

    cursor: pointer;

    transition: background-color 0.3s ease;

}

input[type="submit"]:hover {

    background-color: rgba(218, 148, 0, 1.0); /\* Cambio de color al pasar el mouse \*/

}

/\* Agregar un efecto de sombra al enfocar los campos \*/

input:focus,

select:focus {

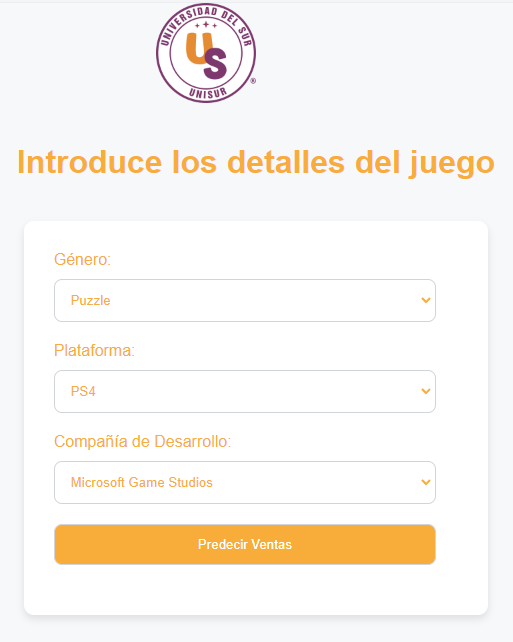
    box-shadow: 0 0 8px rgba(3, 102, 214, 0.5);

    outline: none;

}

</style>

1. Accede a tu aplicación web a través de un navegador web visitando la URL proporcionada por Flask.



1. Despliegue:

Despliega tu aplicación web en un servidor para que esté disponible públicamente a traves de un archivo html llamado result.html.

<body>

    <div class="logo"></div>

    <h2>El promedio total de ventas globales para todas las plataformas es: 548394</h2>

    <h1>{{ mensaje }}</h1>

    <h3>El resultado de la predicción es: {{ prediction\*1000000}} </h3>

    <a href="/">Volver a la página de inicio</a>

</body>

Se considera utilizar el servicio de alojamiento en la nube de github Code Spaces para el despliegue del resultado.



**8. Bibliográfia**

* Documentación oficial de Python: https://docs.python.org/3/
* Documentación oficial de Pandas: https://pandas.pydata.org/docs/
* Documentación oficial de Matplotlib: <https://matplotlib.org/stable/contents.html>
* Documentación oficial de Seaborn: <https://seaborn.pydata.org>
* Documentación oficial de Scikit-learn: https://scikit-learn.org/stable/
* Documentación oficial de Numpy: https://numpy.org/doc/stable/
* Analisis Exploratorio de Datos https://www.aprendemachinelearning.com/analisis-exploratorio-de-datos-pandas-python/
* Conceptos en Python <https://www.geeksforgeeks.org>
* Dudas <https://stackoverflow.com/questions/tagged/pandas+python>
* Analisis de datos <https://ocw.uc3m.es/course/view.php?id=230>
* Diccionarios de datos en data frame <https://github.com/nsheikh23/COVID_StockMarket_Analysis/blob/master/52_Week.ipynb>
* Procesamiento de data frames en pandas https://barcelonageeks.com/eliminar-una-o-varias-columnas-de-pyspark-dataframe/
* Data Clean <https://github.com/mramshaw/Data-Cleaning>
* Ploteo de datos <https://github.com/tomimester/python-histogram/blob/master/plot-histogram-python-pandas.ipynb>
* Data Cleaning in Python: the Ultimate Guide (2020) <https://towardsdatascience.com/data-cleaning-in-python-the-ultimate-guide-2020-c63b88bf0a0d>
* Regresión lineal en Python: un análisis detallado <https://www.cienciadedatos.net/documentos/py10-regresion-lineal-python.html>
* Aplicando Machine Learning para predecir éxitos de videojuegos   
  <https://www.saturdays.ai/projects/Videojuegos.html>
* Machine Learning para la predicción de ventas https://www.aprendemachinelearning.com/regresion-lineal-en-espanol-con-python/
* McKinney, Wes. Python for Data Analysis. O'Reilly Media, Inc, 2017.
* VanderPlas, Jake. Python Data Science Handbook: Essential Tools for Working with Data. O'Reilly Media, Inc, 2016.