Pontificia Universidad Católica de Chile
Departamento de Ciencia de la Computación
Computación: Ciencia y Tecnología del Mundo Digital

Tarea Grande 2

Machine Learning

Ayudantes: Daniela Poblete, Astrid San Martín, Ricardo Schilling

Profesor Denis Parra

Anunciada: 26 de abril de 2020

Indicaciones

- Fecha de Entrega: 12 de mayo de 2019, 20:00 hrs.
- La entrega de la tarea debe realizarse en el repositorio privado GitHub asignado para esta evaluación.
- El descuento por cada hora o fracción de atraso es 1.5 puntos de la nota final.
- Esta tarea debe realizarse de máximo 2 personas. En caso de copia la tarea será evaluada con nota 1.0 junto con las sanciones disciplinarias correspondientes.

Objetivo

Los objetivos de esta tarea son:

- Familiarizarse, analizar y procesar un set de datos.
- Analizar y seleccionar las mejores características dentro del set de datos para solucionar el problema planteado.
- Hacer uso de la librería Pandas de Python para el procesamiento de datos.
- Hacer uso de la librería `scikit-learn` de Python.
- Usar modelos de clasificación y regresión (aprendizaje supervisado) y medir su rendimiento.

- Familiarizarse con algoritmos de aprendizaje no supervisado (para esta tarea clustering). Analizar su aplicación y limitaciones.
- Realizar la visualización de los resultados mediante la librería `Altair` de Python.

Descripción de la tarea

En esta tarea tendrán la oportunidad de explorar dos métodos de aprendizaje automático de *machine learning*: aprendizaje supervisado y no supervisado.
A su vez, podrán hacer análisis exploratorio de un set de datos. En esta oportunidad deberán familiarizarse con un set de datos de ventas de videojuegos `games.csv` que podrán encontrarlo en el repositorio del curso. En este archivo encontrarán información de más de $100{,}000$ videojuegos tales como la plataforma, año de lanzamiento y ventas, entre otros datos.

Para desarrollar esta tarea deberán trabajar en grupos de a dos, para lo cual recibirán un enlace de Github Classroom donde el primer integrante del grupo deberá registrar el equipo, y el segundo deberá incorporarse al equipo, es importante que sólo el primer integrante cree el equipo y el repositorio. Deben empezar la tarea buscando a un compañero y fomando un grupo. **No se permitirán entregas individuales.** Por cierto, deben considerar que los nombres de los equipos tendrán que ser elegidos acorde a la seriedad de la instancia de evaluación de este curso, como lo es una tarea. Consecuentemente, habrá descuentos por nombres poco adecuados. Se sugiere que el nombre del grupo sean los apellidos de los integrantes.

El formato de entrega de cada una de las partes que consta esta tarea deberá ser en un único Jupyter notebook (`.ipynb`). Asimismo, la entrega deben realizarla en el repositorio asignado a cada pareja. Para trabajar con Jupyter notebook es recomendable que usen Google Colab, así evitaran tener que instalar Jupyter localmente y la instalación de todas las demás librerías a utilizar en esta tarea [1].

Esta tarea está dividida en cuatro partes, donde las tres primeras constan de trabajo de programación usando las librerías ya mencionadas (Pandas, `scikit-learn`, `Altair`). La última sección corresponde a un informe donde deberán responder preguntas sobre lo realizado en las primeras partes. Al respecto, deben tomar en cuenta que esta última parte de la tarea es la que posee mayor puntaje por lo cual, es

[1] Si luego de probar muchas con Colab tuviese problemas que no le permitieran hacer la tarea, puede hacerla localmente con jupyer notebook, pero recuerde usar Python 3 y también crear un archivo **requirements.txt** con las versiones de las bibliotecas que usó (pandas, scikit-learn, altair y otras)

importante que pongan mucha atención a las preguntas planteadas antes de empezar a trabajar, es así como prodrán focalizar de manera efectiva el desarrollo de la tarea.

Set de datos

Los datos para realizar la tarea se encuentran en el archivo `games_data.csv`, donde cada fila corresponde a un videojuego, con los siguientes atributos por columna:

- `ID:` Identificador del videojuego
- `Rank:` Ranking del videojuego.
- `Name:` Nombre del videojuego.
- `Platform:` Plataforma en la cual se utiliza el videojuego.
- `Year:` Año de lanzamiento del videojuego.
- `Genre:` Género del videojuego.
- `Difficulty:` Dificultad del videojuego (valores entre 1 y 10).
- `Publisher:` Editora del juego.
- `Ign_score:` Estrellas del videojuego (valores entre 1 y 10).
- `NA_Sales:` Ventas en América del Norte (en millones de dólares).
- `EU_Sales:` Ventas en Europa (en millones de dólares).
- `JP_Sales:` Ventas en Japón (en millones de dólares).
- `Other_Sales:` Ventas en el resto del mundo (en millones de dólares).

Parte 1: Procesamiento de los datos (1 pto)

En esta primera parte deberán enfocarse en leer, analizar y limpiar los datos, es decir, deben familiarizarse con la base de datos. Para ello, es obligatorio hacer uso de la librería Pandas y quedando prohibido utilizar ciclos pues se espera que el código sea eficiente.

Análisis de las características (0.3): Como se detalló, en la base de datos las características o atributos que tiene dan información sobre el origen y venta de distintos videojuegos. Por lo tanto, deben analizar la asignación de la clase ó etiqueta y poner atención si algunas de esas características tienen

o no influencia en la clase, considerando dos vectores de clase distintos: editora del juego y ventas en el resto del mundo. Es así como, después de cargar los datos deberán eliminar las columnas que consideren que no aportan información para el objetivo de la tarea (determinar la clase ó etiqueta de cada videojuego). Tengan en consideración que en el informe deberán justificar esta toma de decisión.

Limpiar valores nulos (0.4): El siguiente paso que deben realizar es buscar si el dataset contiene valores nulos. Así, usando la librería Pandas, deberán encontrar los valores nulos o `NaN` (*not a number*) dentro del DataFrame. Si encuentran estos valores nulos deberán realizar la toma de decisión sobre qué hacer con ellos (imputar un valor, eliminar el dato, etc.) y justificar debidamente en el informe.

Separar la clase y normalizar (0.3): Por último, deberán separar el dataset en una matriz de características (*features*) y un vector de clases para luego, normalizar las características. Para esta tarea, como ya se mencionó, se esperan dos separaciones: una donde el vector de clase corresponda a la editora del juego y la otra que corresponda a las ventas en el resto del mundo.

De aquí en adelante, será con estos datos ya procesados con los cuales se continuará trabajando, es decir, sin columnas innecesarias, sin datos nulos y con valores normalizados.

Parte 2: Clasificación automática y regresión (1,5 ptos)

En esta segunda parte de la tarea, deberán hacer uso de un modelo de clasificación y otro de regresión. Será necesario como primer paso entrenar el algoritmo usando un set de entrenamiento, y posteriormente evaluar el desempeño del mismo usando un set de testeo. En el caso de la predicción, se entregan los datos a clasificar sin la etiqueta de la clase, con el objeto de medir qué tan bien resulta la clasificación. Finalmente, se deberá utilizar algunas métricas ó estadísticas de evaluación para chequear cómo lo hizo el modelo entrenado.

Dentro de las herramientas disponibles en el área de *Machine Learning*, existen varios modelos de clasificación. Para esta tarea utilizarán uno de los algoritmos más sencillos e intuitivos: K Nearest Neighbors (KNN). Este modelo agrupa los datos de acuerdo a los K vecinos cercanos (K a definir), para posteriormente clasificar las instancias según la clase más popular entre sus vecinos más cercanos.
Como modelo de regresión deberán hacer uso del algoritmo Support Vector Machine (SVM). Este modelo, en palabras sencillas, busca un hiperplano de N dimensiones, con N el número de features, que encuentre la clasificación (en su versión como clasificador). En esta tarea usarán SVM en su versión

como regresor.
En esta ocasión, utilizarán la librería `scikit-learn` para trabajar en la definición y entrenamiento del modelo.

Separar datos de entrenamiento y pruebas (0.3): En orden de preparar los datos para el entrenamiento, deberán separar el set de datos en set de entrenamiento y set de pruebas. Para definir el porcetanje de datos a usar para entrenamiento y testeo, podrán buscar en la literatura en internet las proporciones recomendadas, una vez más deberán reportar la proporción escogida justificando.

Instanciar y entrenar el clasificador (0.4): Ahora con el set de datos de entrenamiento definido, deben instanciar el modelo KNN y entrenarlo.

Instanciar y entrenar el regresor (0.4): Asimismo, deberán instanciar el modelo SVM de regresión y entrenarlo usando el set de entrenamiento.

Calcular el rendimiento de los clasificadores y regresores (0.4): Una vez entrenado cada modelo, corresponde testear el rendimiento de cada uno. Para poder conocer el rendimiento tendrán que utilizar una métrica que se conoce como *accuracy* o *score*. De este modo, deberán hacer la predicción usando el set de prueba o testeo, y comparar las predicciones con las clases reales. Se debe reportar el *accuracy score* obtenido tanto para el set de entrenamiento y el set de pruebas.

Para un correcto análisis del entrenamiento y rendimiento de los modelos, los pasos señalados anteriormente deberán ser repetidos probando distintos hiperparámetros. El detalle de los hiperparámetros a modificar pueden encontrarlos en la documentación de `scikit-learn`. Finalmente, deben escoger la combinación de hiperparámetros que muestre el mejor rendimiento, siendo este set de hiperparámetros el que deberá quedar en el código. En el informe deberán reportar y explicar las pruebas realizadas, incluyendo la toma de decisiones que se hicieron.

Parte 3: Reducción de dimensionalidad y clustering (1,5 ptos)

En la sección anterior usaron el dataset de videojuegos para generar un clasificador capaz de predecir las clases en muestras no vistas por el algoritmo ("nuevas"). Ahora, se les pide realizar dos tareas nuevas muy comúnes y útiles cuando se trabaja con un set de datos.

La primera tarea corresponde a reducción de dimensionalidad, en palabras simples, la idea es tomar todas las características del set de datos y hacer una transformación lineal que nos entregue la misma información pero con un número menor de características. Esto nos permite, por ejemplo, llevar un set de datos en tres dimensiones a dos dimensiones para poder visualizarlo de forma más sencilla.

La segunda tarea corresponde a clustering, es decir, encontrar etiquetas para datos sin clase asignada previamente, agrupandolos. La idea de este tipo algoritmos es poder separar los datos cuando NO conocemos las clases, siendo este un ejemplo de aprendizaje no supervizado.

A diferencia de la primera parte, no será necesario usar un set de entrenamiento y un set de pruebas sino que, utilizarán el conjunto de datos obtenidos en la primera parte.

Reducción de dimensionalidad (0.3): Para realizar reducción de dimensionalidad deberán usar una técnica llamada *T-distributed Stochastic Neighbor Embedding* (t-SNE), de modo que los datos queden representados con solo dos características (columnas) derivadas de las características originales.

Visualizar datos reducidos (0.3): Una vez realizada la reducción de la matriz de características a dos columnas, debrán mostrar la reducción resultante graficando los datos en dos dimensiones usando Altair [2]. Cada eje del gráfico obtenido debe corresponder a cada una de las componentes de t-SNE, y se espera que cada punto pueda ser identificado con su clase respectiva (ya sea según color o forma del marcador).

Predecir las clases usando clustering (0.6): En esta etapa, desestimarán las clases reales, y sólo usando las dos características resultantes de t-SNE, deberán aplicar dos algoritmos de clustering para dividir los datos. En concreto, deberán obtener una predicción de clase para los datos reducidos. Algo muy importante a considerar es que los algoritmos de clustering a ser utilizados deberán ser distintos, en específico, uno basado en centroides y uno basado en densidad.

Visualizar los clusters (0.3): Nuevamente, deberán graficar los datos en dos dimensiones usando Altair. Se espera que cada eje del gráfico sea una de las componentes de t-SNE y, cada dato pueda ser identificado con la clase que predijo (según el color o la forma del marcador).

[2] Por defecto Altair no permite hacer gráficos con más de 5000 puntos. Para deshabilitar esta restricción debes hacer lo que dice la documentación.

Parte 4: Informe (2 ptos)

En base a todo lo realizado hasta ahora, deberán responder algunas preguntas con el objetivo de evaluar los conceptos aplicados en esta tarea. Recuerden poner particular atención a esta parte de la tarea, la parte más importante de tu tarea.

La siguientes preguntas deberán ser incluídas en el archivo `.ipynb`, usando una celda en formato *markdown*. Las respuestas tienen una extensión máxima de 5 líneas para las preguntas 1, 2, 3 y 4, y 3 líneas para las preguntas 5, 6, 7 y 8. Deberán hacer uso de la letra por defecto en el formato *markdown*, de no cumplir con estas restricciones existirá una penalización en el puntaje.

1. ¿Eliminaste columnas en el dataset que no aportaban información? De ser así, ¿cuáles fueron y por qué? ¿Cómo resolviste el tema de los valores nulos? (0.3)
2. ¿Qué normalizaste, filas o columnas? ¿Por qué? ¿Para qué sirve normalizar los datos? (0.3)
3. ¿Por qué se separan los datos en set de entrenamiento y set de pruebas? ¿Qué proporción de los datos utilizaste para cada uno y por qué? (0.3)
4. ¿Qué hiperparámetros modificaste para probar tu clasificador? ¿Y el regresor? ¿Cuáles combinaciones de parámetros te dieron mejores resultados y por qué crees que es así? (0.2)
5. Explica la diferencia entre el *accuracy score* obtenido en el set de entrenamiento y el set de pruebas. Justifica. (0.3)
6. Al usar t-SNE reducimos la dimensionalidad de los datos. Nombra dos razones para hacer eso. (0.2)
7. ¿Cuantos centroides elegiste para la división del set de datos? Justifica. (0.2)
8. ¿Que observaste al graficar los puntos luego de reducir su dimensionalidad? ¿Servirá hacer clustering para este set de datos? (0.2)

Bonus (0,5 ptos)

Este bonus solo será contabilizado si la nota obtenida en la tarea es igual o superior a 4.

Para obtener el puntaje del bonus, deberán usar un clasificador y un regresor diferente al usado en la tarea, entrenarlo usando set de entrenamiento (de igual forma que en la segunda parte), y finalmente

verificar el rendimiento del (*accuracy*) en el set de pruebas. Deberán realizar una comparación sobre el rendimiento obtenido, escribiendo un breve comentario explicando esa diferencia.

Formato de entrega

La entrega de esta tarea se hará por medio del repositorio GitHub privado del grupo, creado a través de este link, donde deberán entregar un único archivo `.ipynb` con todo el desarrollo de la tarea y preguntas del informe respondidas al final de este. Si la tarea fue realizada con Google Collab, entonces desde ahí mismo pueden descargar el archivo `.ipynb` y subirlo al repositorio.

Es sumamente importante que elimines el *output* de tu Jupyter notebook. Tampoco debes subir el dataset (archivo `.csv`) a tu repositorio. No seguir estas instrucciones significará un descuento. Además, deberán ingresar el nombre de ambos integrantes al principio del archivo y citar **todo** código externo usado.

En caso de entregar la tarea atrasada, deben enviar un mail a reschilling@uc.cl con un mínimo de 1 hora de anticipación, indicando con cuantas horas de atraso entregarán la tarea y el nombre de su repositorio (por ejemplo, si entregarán la tarea a las 2:00, deben enviar un mail máximo a la 1:00 indicando dos horas de atraso). En caso de no hacerlo, se considerará sólo la entrega a la hora correspondiente (20:00 horas), sin apelaciones.