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

Tarea Grande 2

Machine Learning

Ayudantes: Felipe García, Juan Pablo Olivares, Astrid San Martín

Profesor: Denis Parra

Anunciada: 26 de Octubre de 2020

Indicaciones

- Fecha de Entrega: 12 de Noviembre, hasta las 23:59.
- La entrega de la tarea debe realizarse en el repositorio privado GitHub asignado para esta evaluación.
- Todas las tareas se entregarán máximo a las 23:59 del plazo indicado. Puede entregar con uno o dos días de atraso, cada día de atraso significa 2 puntos de descuento a la nota máxima de la tarea, indicándolo previamente al ayudante jefe por mail (reschilling@uc.cl).
- 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.

Objetivos

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 (aprendizaje supervisado) y medir su rendimiento.

- Realizar la visualización de los resultados con una librería de visualización de python a elección (altair, matplotlib, seaborn, etc.).

Descripción de la tarea

En esta tarea tendrán la oportunidad de explorar un método de aprendizaje automático de *machine learning*, el aprendizaje supervisado.

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 formando 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 un único Jupyter notebook (`.ipynb`). Asimismo, la entrega deben realizarla en el repositorio asignado a cada pareja, **para crearlo deben usar el siguiente link**. 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 tres partes, en cada parte deberan aplicar conocimientos de programacion con las librerias ya mencionadas (`Pandas`, `scikit-learn` y, opcionalmente, `Altair`), pero ademas, deberan responder preguntas que justifiquen sus conocimientos respecto al tema, esto es igual de importante como la parte de programacion.

[1] Si luego de probar muchas con Colab tuviese problemas que no le permitieran hacer la tarea, puede hacerla localmente con Jupyter 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)

Motivación

Diversos estudios de universidades en Chile señalan que los chilenos se alimentan mal, provocando problemas de salud como obesidad, desnutrición o problemas cardíacos . Si a lo anterior se le suma la contingencia, que en los últimos meses ha significado un incremento en el sedentarismo y estrés (acuérdate de relajar tu postura y la vista), podemos comprender que este problema se ha agravado aún cuando ya era lo suficientemente grave como para liderar el ranking OCDE de sobrepeso del 2019.

Ante esta situación, las encuestas revelan que algunas de las causas del problema que se acusan son la falta de información (no saber interpretar la información nutricional del alimento que consumen), no tener tiempo para cocinar (por lo que acuden a aplicaciones como la del bigote) y dinero. Estos resultados pueden considerarse algo paradójicos, debido a que "**Vivimos** en ~~una sociedad (c¬_¬)~~ la supuesta era de la información", donde tenemos los datos en la palma de la mano, pero resulta que estamos expuestos a tanta información, que terminamos sobrepasados y sin poder interpretar lo esencial. Son necesarias la habilidades que te permitan entender estudios y ser capaz de corroborar la información. Entre estas habilidades se puede encontrar el ser capaz de manipular bases de datos y detectar patrones o información relevante en ellos.

Figura 1: Los "deliveries" han sido protagonistas en esta cuarentena.

Set de datos

Para esta tarea tendrán a su disposición el set de datos: `data.csv`

Este dataset corresponde a un estudio de monitoreo de prevalencia de obesidad en USA a través del estudio de la actividad en portales de recetas de comida [2]. En el set datos que van a trabajar podrán encontrar las propiedades nutricionales promedio de recetas de comida subidas por usuarios de un determinado condado y estado de USA al sitio de recetas de comida `www.allrecipes.com`. También podrán revisar el índice de prevalencia de obesidad y diabetes por estado y condado, junto con la cantidades de personas que son de ese lugar que son usuarios del portal de recetas *All Recipes* y, a su vez, la cantidad de *likes* que recibieron las recetas de dicho condado y estado de USA.

Las columnas que encontraremos en nuestro dataset son:

`data.csv`

- `users`: cantidad de usuarios pertenecientes a dicho condado.
- `bookmarks`: cantidad de likes que posee la receta.
- `county`: condado de determinado estado de USA al cual pertenece el usuario que subió la receta.
- `state`: estado de USA al cual pertenece el condado.
- `cal`: cantidad de calorías promedio de las recetas de dicho condado en USA.
- `cal_fat`: cantidad de calorías provenientes de la grasa promedio de las recetas de dicho condado en USA.
- `col`: cantidad de colesterol promedio de las recetas de dicho condado en USA.
- `fiber`: cantidad de fibra promedio de las recetas de dicho condado en USA.
- `sodium`: cantidad de sodio promedio de las recetas de dicho condado en USA.
- `carbs`: cantidad de hidratos de carbonos/ carbohidratos promedio de las recetas de dicho condado en USA.
- `fat`: cantidad de grasa promedio de las recetas de dicho condado en USA.
- `protein`: cantidad de proteínas promedio de las recetas de dicho condado en USA.

[2] https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0179144

- `sugar:` cantidad de azúcar/hidratos de carbonos simples promedio de las recetas de dicho condado en USA.
- `dabetis:` índice de prevalencia de diabetes en dicho condado de USA.
- `obese:` índice de prevalencia de obesidad en dicho condado de USA.

Parte 1: Procesamiento de los datos (2 ptos)

Como primer paso para esta tarea deberán realizar el preprocesamiento de los datos, con el objetivo de familiarizarse y dejarlos preparados para la siguiente parte de la tarea. Para esta parte es **obligatorio hacer uso de la librería Pandas** y queda **prohibido utilizar ciclos en esta parte**, pues se espera que el código sea eficiente.

Cargar dataset (0.1p): Para poder iniciar esta tarea, es necesario que carguen la base de datos: `data.csv` .

Análisis de las características (1p): Para realizar este paso, deberán estudiar los datasets usando Jupyter Notebook, usando funciones de pandas que entreguen estadísticas de los datos (i.e. `describe()` `de pandas`), y realizando gráficos de distribución de los datos, verificando outliers, missing data, etc., siempre justificando las decisiones que tomen. Deberán mostrar al menos un gráfico que muestre, por ejemplo: la distribución de prevalencia de obesidad y diabetes por estado.

Limpiar valores nulos (0.4p): 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 su decision**.

Normalizar y crear label (0.5p): Para este paso, procederemos a normalizar las características, llevando los datos a un rango de valores común, y crearemos un label en función de las calorías, el cual debe tener, al menos, 4 posibles valores (los sellos de las comidas son una referencia interesante).

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: Reducción de dimensionalidad (0.5 ptos)

En esta segunda parte de la tarea, deberán realizar reducción de dimensionalidad. Esto permite que un humano (investigador) pueda visualizar un *resumen*" de los datos, pero preservando la información, y, así, tener una mejor de idea de cómo estos se distribuyen y cómo se podrían trabajar.
Para ello, deberán trabajar la información mediante alguno de los métodos conocidos como PCA (*Principal Component Analysis*) o t-SNE (*T-distributed Stochastic Neighbor Embedding*), de tal manera que visualicen, bidimensional o tridimensionalmente, la tendencia de los datos y la formación de clusters de interés en función de sus features. En palabras simples, la idea es tomar todas las características del set de datos y hacer una transformación lineal[3] que nos entregue una información similar, 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.

Para esta parte deberán utilizar la librería `Scikit-learn`, ya sea usando el algoritmo de *PCA* o *t-SNE* sobre los datos. Posteriormente, deberán realizar la visualización usando librerías gráficas de Matplotlib, Seaborn o Altair[4].

Para la visualización, se espera que cada eje del gráfico obtenido corresponda a cada una de las componentes del reductor de dimensiones y que cada punto pueda ser identificado con su label/categoría respectiva (mediante colores, figuras u otros, e incluyendo la leyenda).

Parte 3: Clasificación (2 ptos)

En esta parte veremos dos tipos de clasificadores, los cuales representan ejemplos de aprendizaje supervisado. **A diferencia de la parte 1** donde está prohibido usar ciclos, en esta parte **si podrán hacer uso de estos.**

3.1 K-Nearest Neighbors (1 ptos)

Para esta parte, deberá realizar un clasificador KNN sobre la base de datos, ocupando un *training set* para el entrenamiento, un *testing set* para la evaluación, y una categoría que sirve para agrupar los datos bajo determinadas etiquetas. No está demás mencionar que este clasificador es uno de los

[3] Es real, hay Álgebra Lineal en Computación (O_O;). No temais, this is the way.

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

ejemplos de aprendizaje supervisado, cuya labor es predecir qué etiqueta debería corresponderle a los datos nuevos que llegan.

Separar datos de entrenamiento y pruebas (0.3p): 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 porcentaje 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 (0.3p): Ahora con el set de datos de entrenamiento definido, deben instanciar el modelo KNN y entrenarlo (*fit*). Para este algoritmo deben utilizar como etiqueta/-clase el label creado en la *parte 1*.

Calcular el rendimiento del clasificador (0.4p): Una vez entrenado el clasificador, corresponde testear el rendimiento de cada clase. Para poder conocer el rendimiento tendrán que utilizar las métricas: *accuracy*, *precision*, *recall* y *f-1 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 las métricas para cada clase en 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. Deben probar al menos 3 combinaciones distintas. En el informe deberán reportar y explicar las pruebas realizadas, incluyendo la toma de decisiones que se hizo.

3.2 Decision Tree (1 ptos)

En la sección anterior usaron el dataset completo para generar un clasificador capaz de predecir las clases en muestras o testeo, es decir no vistas por el algoritmo ("nuevas") dada la información ocupada en el entrenamiento. Ahora, se les pide realizar una nueva tarea.

En esta parte deberán utilizar un nuevo algoritmo de *clasificación*, el cual es *Decision Tree*. Tal como se realizó en el paso anterior, deberán aplicar este algoritmo sobre la base de datos conjunta, ocupando un *training set* para el entrenamiento y un *testing set* para la evaluación. Este clasificador es

otro ejemplo de aprendizaje supervisado, cuya labor, tal cual vimos con KNN, es predecir qué etiqueta debería corresponderle a los datos nuevos que llegan.

Separar datos de entrenamiento y pruebas (0.3p): 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 porcentaje 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 (0.3p): Ahora con el set de datos de entrenamiento definido, deben instanciar el modelo de Decision Tree y entrenarlo (*fit*). Para este algoritmo deben utilizar como etiqueta/clase el label creado en la *parte 1*.

Calcular el rendimiento del clasificador (0.4p): Una vez entrenado el clasificador, corresponde testear el rendimiento de cada clase. Para poder conocer el rendimiento tendrán que utilizar las métricas: *accuracy*, *precision*, *recall* y *f-1 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 las métricas para cada clase en 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. Deben probar al menos 3 combinaciones distintas. En el informe deberán reportar y explicar las pruebas realizadas, incluyendo la toma de decisiones que se hizo.

Parte 4: Preguntas (1.5 ptos)

En base a todo lo realizado hasta ahora, deberán responder algunas preguntas con el objetivo de evaluar los conceptos aplicados en esta tarea. Se busca evaluar la comprensión de conceptos vistos y desarrollados en clase.

La siguientes preguntas deberán ser incluídas en el archivo `.ipynb`, usando una celda en formato *MarkDown*. Las respuestas deberán tener una extensión máxima de 4 líneas para las preguntas y deberán

hacer uso de la letra por defecto del formato *markdown*, de no cumplir con estas restricciones existirá una penalización en el puntaje.

Preguntas "Parte 1"(0.5p)

- ¿Que columnas modificaste?, ¿Existe alguna columna en los datasets con información innecesaria? De ser así, ¿cuáles fueron y por qué?
- ¿Cómo resolviste el tema de los valores nulos?
- ¿Qué normalizaste?¿Filas o columnas? ¿Por qué? ¿Para qué sirve normalizar los datos?
- ¿Cómo determinaste los labels? Justifica tu decisión
- ¿Qué información encontraste al estudiar los atributos del dataset? ¿Encontraste algún aspecto que te llamara la atención? ¿ Por qué?

Preguntas "Parte 2"(0.5p)

- ¿Cuántas agrupaciones puede observar?¿Por qué crees que pasa esto?
- ¿Qué características de la ubicación espacial (posición en el mapa) puedes deducir de las calorías promedio de las recetas? Vale decir, si miras la proyección 2D de las muestras, ¿Qué podría decirse sobre los datos que están en la parte de más abajo, más a la derecha o más a la izquierda del mapa?

Preguntas "Parte 3.1"(0.25p)

- ¿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?
- ¿Qué hiperparámetros modificaste para probar tu clasificador? ¿Cuáles combinaciones de parámetros te dieron mejores resultados y por qué crees que es así?
- Para el clasificador, explica la diferencia entre las métricas del set de pruebas para cada clase, ¿Qué nos dice de la calidad del clasificador por cada clase?. Justifica.

Preguntas "Parte 3.2"(0.25p)

- ¿Qué hiperparámetros modificaste para probar tu clasificador? ¿Cuáles combinaciones de parámetros te dieron mejores resultados y por qué crees que es así?
- Para el clasificador, explica la diferencia entre las métricas del set de pruebas para cada clase, ¿Qué nos dice de la calidad del clasificador por cada clase?. Justifica.

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, diferente a los usados en la tarea, entrenarlo usando el 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. Además, debe incorporar una pequeña explicación de cómo resolvería este problema con un regresor, explicando lo que es, qué elementos cambiaría y qué elementos mantendría.

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 Colab, 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 a la hora indicada de entrega de la tarea. En caso de no hacerlo, se considerará sólo la entrega a la hora correspondiente, sin apelaciones.